

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



**BÁO CÁO BÀI THỰC HÀNH
THỰC TẬP CƠ SỞ**

**Bài 15: Lập trình client/server để trao đổi
thông tin an toàn**

Họ và tên: Nguyễn Huy Quang

Mã sinh viên: B20DCAT144

Giảng viên: Nguyễn Hoa Cường

Hà Nội – 2023

I. Mục đích

- Sinh viên hiểu về cơ chế client/server và có thể tự lập trình client/server dựa trên socket, sau đó thực hiện ca đặt giao thức đơn giản để trao đổi thông tin an toàn.

II. Nội dung thực hành

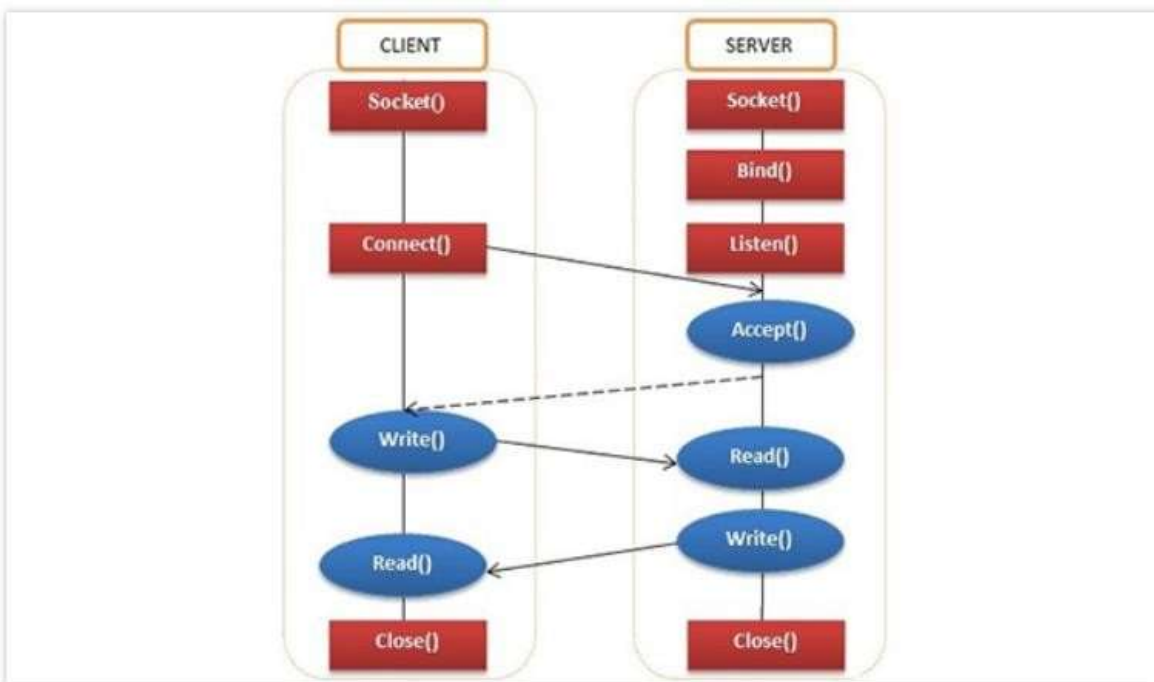
2.1. Tìm hiểu lý thuyết

2.1.1. Tìm hiểu về các khái niệm liên quan tới lập trình socket với TCP

- Socket là giao diện lập trình ứng dụng mạng được dùng để truyền và nhận dữ liệu trên internet. Giữa hai chương trình chạy trên mạng cần có một liên kết giao tiếp hai chiều, hay còn gọi là two-way communication để kết nối 2 process trò chuyện với nhau. Điểm cuối (endpoint) của liên kết này được gọi là socket.

- Một chức năng khác của socket là giúp các tầng **TCP** hoặc **TCP Layer** định danh ứng dụng mà dữ liệu sẽ được gửi tới thông qua sự ràng buộc với một cổng port (thể hiện là một con số cụ thể), từ đó tiến hành kết nối giữa client và server.

- Dựa trên giao thức TCP(Transmission Control Protocol), stream socket thiết lập giao tiếp 2 chiều theo mô hình client và server. Được gọi là socket hướng kết nối.

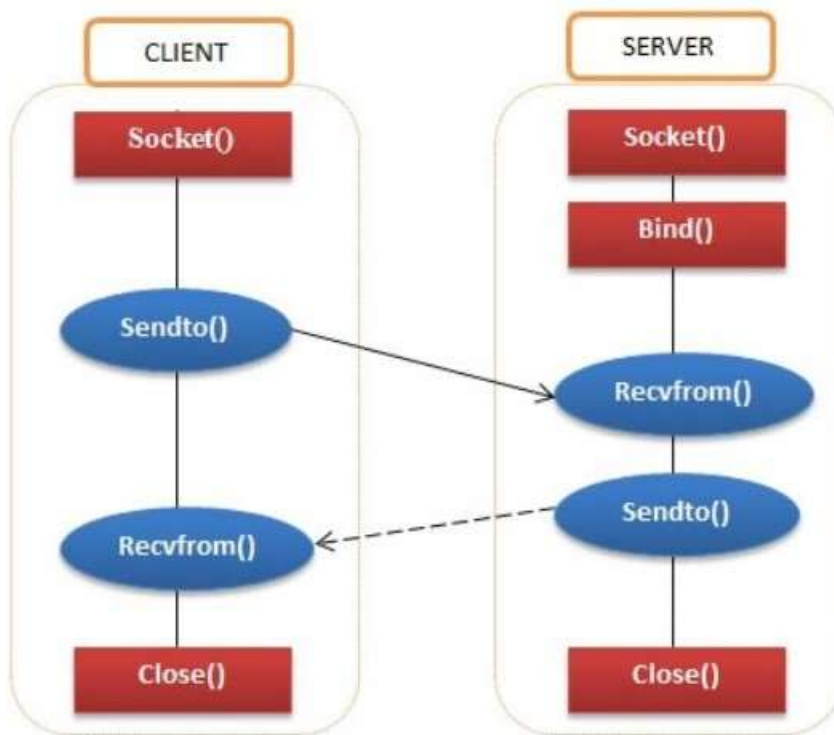


Stream Socket chỉ hoạt động khi server và client đã kết nối với nhau

- **Stream Socket** hay còn gọi là socket hướng kết nối, là socket hoạt động thông qua **giao thức TCP (Transmission Control Protocol)**. Stream Socket chỉ hoạt động khi server và

client đã kết nối với nhau. Giao thức này đảm bảo dữ liệu được truyền đến nơi nhận một cách đáng tin cậy, đúng tuần tự nhờ vào cơ chế quản lý luồng lưu thông trên mạng và cơ chế chống tắc nghẽn.

- Dựa trên giao thức UDP(User Datagram Protocol) việc truyền dữ liệu không yêu cầu có sự thiết lập kết nối giữa 2 process. Tức là nó cung cấp connection-less point cho việc gửi và nhận packets. Gọi là socket không hướng kết nối.



-Do không yêu cầu thiết lập kết nối, không phải có những cơ chế phức tạp. Nên tốc độ giao thức khá nhanh, thuận tiện cho các ứng dụng truyền dữ liệu nhanh như chat, game online...

2.2. Thực hành

1. Lập trình client và server với TCP socket

- Lập trình Client

```
1 import socket
2
3 HOST = '127.0.0.1'
4 PORT = 8080
5 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
6 s.connect((HOST, PORT))
7 s.send(b"Hello, I am B20DCAT144 Client")
8 server_msg = s.recv(1024)
9 print("Receive from server: " + server_msg.decode())
10 s.close()
```

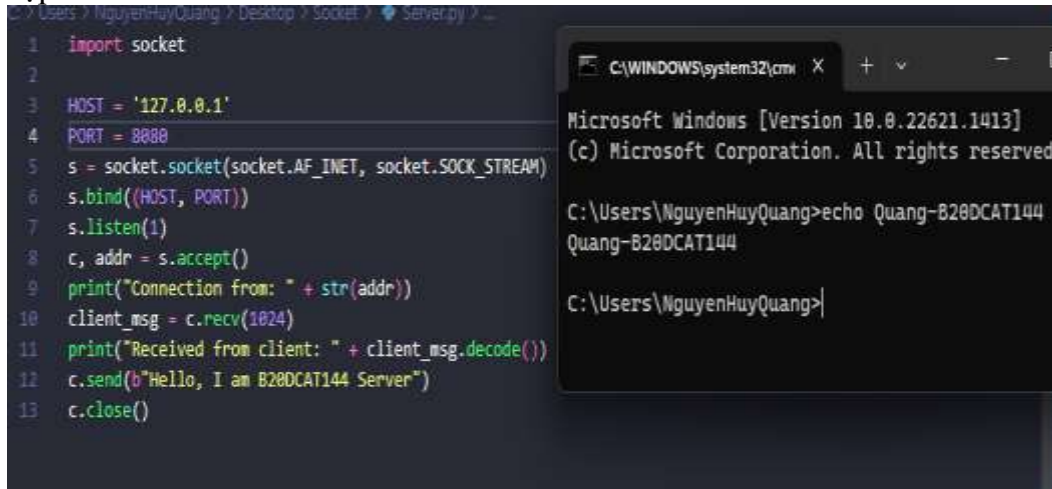
The screenshot shows a Windows command prompt window with the following commands and output:

```
C:\WINDOWS\system32\cmd
Microsoft Windows [Version 10.0.22621.1413]
(c) Microsoft Corporation. All rights reserved.

C:\Users\NguyenHuyQuang>echo Quang-B20DCAT144
Quang-B20DCAT144

C:\Users\NguyenHuyQuang>
```

- Lập trình Server

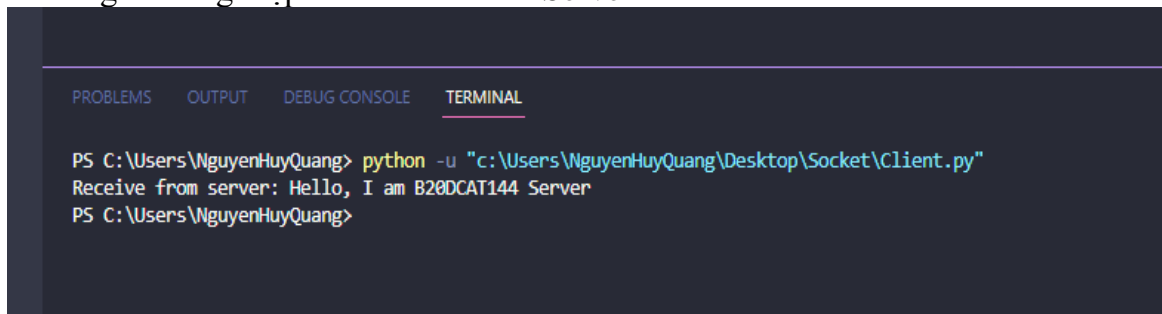


The screenshot shows a code editor with a Python script for a server. The script is as follows:

```
1 import socket
2
3 HOST = '127.0.0.1'
4 PORT = 8888
5 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
6 s.bind((HOST, PORT))
7 s.listen(1)
8 c, addr = s.accept()
9 print("Connection from: " + str(addr))
10 client_msg = c.recv(1024)
11 print("Received from client: " + client_msg.decode())
12 c.send(b"Hello, I am B20DCAT144 Server")
13 c.close()
```

Overlaid on the code editor is a Windows command prompt window. It shows the command `echo Quang-B20DCAT144` being executed, which outputs `Quang-B20DCAT144`. The window title is `C:\WINDOWS\system32\cmd`.

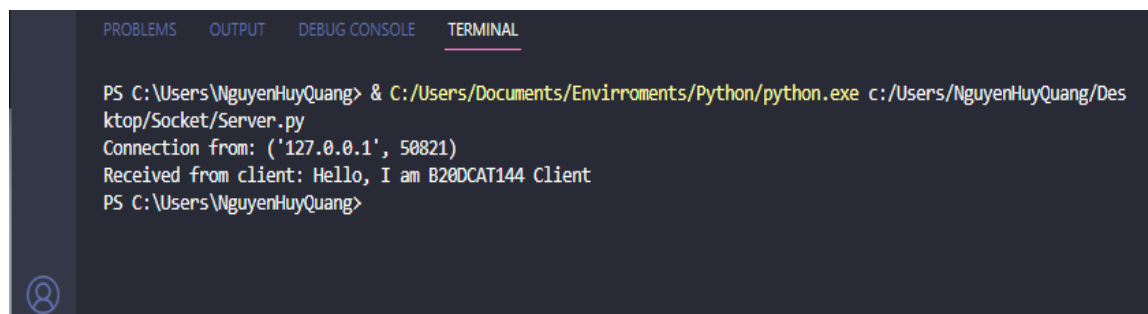
- Chạy Server sau đó chạy Client
- Client gửi thông điệp cá nhân hóa cho Server



The screenshot shows a terminal window with the following text:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
PS C:\Users\NguyenHuyQuang> python -u "c:\Users\NguyenHuyQuang\Desktop\Socket\Client.py"
Receive from server: Hello, I am B20DCAT144 Server
PS C:\Users\NguyenHuyQuang>
```

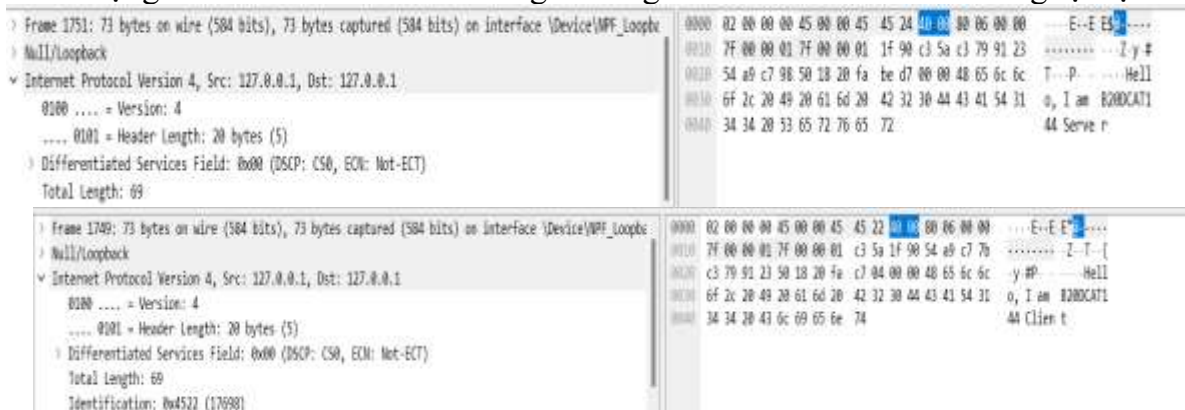
- Server nhận và gửi lại thông điệp cho Client



The screenshot shows a terminal window with the following text:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
PS C:\Users\NguyenHuyQuang> & C:/Users/Documents/Enviroments/Python/python.exe c:/Users/NguyenHuyQuang/Desktop/Socket/Server.py
Connection from: ('127.0.0.1', 50821)
Received from client: Hello, I am B20DCAT144 Client
PS C:\Users\NguyenHuyQuang>
```

- Sử dụng Wireshark để bắt các thông tin đã gửi từ client đến server và ngược lại



2. Trao đổi thông điệp giữa Client và Server và đảm bảo tính toàn vẹn của thông điệp khi trao đổi

- Từ client và server, sửa đổi để sao cho: khi gửi thông điệp sẽ gửi kèm theo giá trị băm của thông điệp + key để phía bên kia kiểm tra xác minh tính toàn vẹn. hai bên có thể thống nhất một giá trị key trước đó
 - Sửa code client thống nhất key giữa server và client là “Quang”

```

Clientkey.py
1 import socket
2 import hashlib
3
4 HOST = '127.0.0.1'
5 PORT = 5000
6 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
7 s.connect((HOST, PORT))
8 while True:
9     print("Client: ", end='')
10    mes = input()
11    key = "Quang"
12    hashmes = hashlib.md5((mes + key).encode()).hexdigest()
13    s.sendall(bytes(mes, "utf8"))
14    s.sendall(bytes(hashmes, "utf8"))
15    data = s.recv(1024)
16    hashdata = s.recv(1024)
17    str_data = data.decode("utf8")
18    str_hashdata = hashdata.decode("utf8")
19    print('Server Response: ', data.decode("utf8"))
20    checkhash = hashlib.md5((str_data + key).encode()).hexdigest()
21    if str_hashdata == checkhash:
22        print("message from server is OK")
23    else:
24        print("the received message from server has lost its integrity")
25    if not data:
26        break

```

```

C:\WINDOWS\system32\cmd
Microsoft Windows [Version 10.0.22621.1413]
(c) Microsoft Corporation. All rights reserved.

C:\Users\NguyenHuyQuang>echo Quang-B20DCAT144
Quang-B20DCAT144

C:\Users\NguyenHuyQuang>

```


- code server để kiểm tra tính toàn vẹn của thông điệp

The screenshot shows a Visual Studio Code editor with a file named `Serverkey.py` open. The code is a Python script that uses the `socket` and `hashlib` modules to create a server. It binds to `127.0.0.1` on port `5000` and listens for connections. When a connection is accepted, it prints the address and enters a loop where it receives data from the client. The data is decoded and hashed using MD5 with a key "Quang". It then compares the received hash with the calculated hash. If they match, it prints "message from client is OK". Otherwise, it prints "the received message from client has lost its integrity". The server also prompts for a message from the client, hashes it with the same key, and sends the hash back to the client. The loop breaks if the client sends "quit".

Overlaid on the editor is a Windows command prompt window. It shows the command `echo Quang-B20DCAT144` being executed, which outputs `Quang-B20DCAT144`. The prompt is currently at `C:\Users\NguyenHuyQuang>`.

- Trao đổi thông điệp khi chưa đổi key ở client

The screenshot shows a Windows command prompt window with the following text:

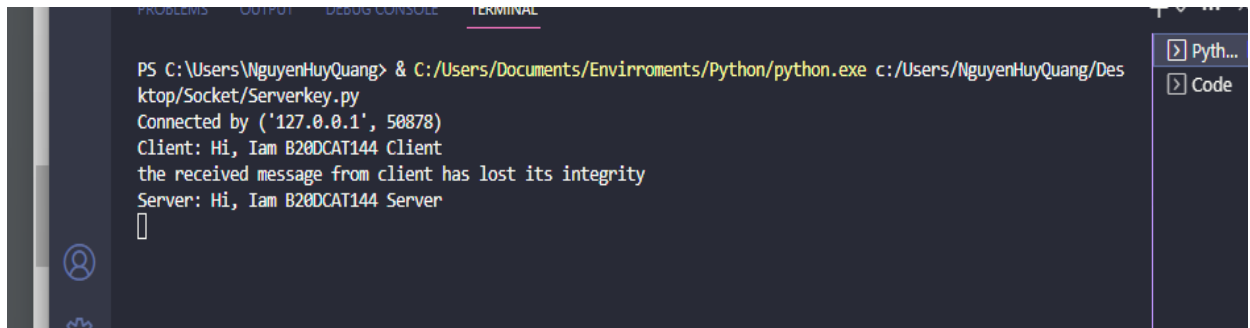
```
PS C:\Users\NguyenHuyQuang> python -u "c:\Users\NguyenHuyQuang\Desktop\Socket\Clientkey.py"
Client: Hi, I am B20DCAT144 Client
Server Response: Hi, I am B20DCAT144 Server
message from server is OK
Client: 
```

```
PS C:\Users\NguyenHuyQuang> & C:/Users/Documents/Enviroments/Python/python.exe c:/Users/NguyenHuyQuang/Desktop/Socket/Serverkey.py
Connected by ('127.0.0.1', 50864)
Client: Hi, I am B20DCAT144 Client
message from client is OK
Server: Hi, I am B20DCAT144 Server
[]
```

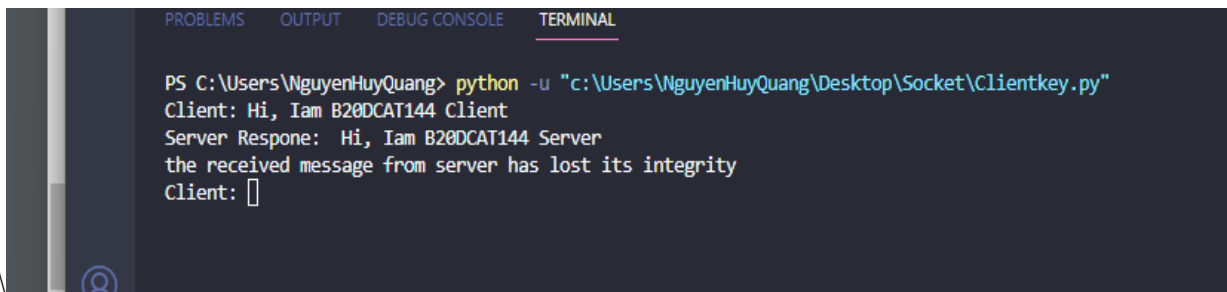
- Thay đổi key ở Client thành “B20DCAT144”

```
Clientkey.py X Serverkey.py Client.py Server.py
C:\Users\NguyenHuyQuang\Desktop\Socket> Clientkey.py
1 import socket
2 import hashlib
3
4 HOST = '127.0.0.1'
5 PORT = 5000
6 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
7 s.connect((HOST, PORT))
8 while True:
9     print("Client: ", end='')
10    mes = input()
11    key = "B20DCAT144"
12    hashmes = hashlib.md5((mes + key).encode()).hexdigest()
13    s.sendall(bytes(mes, "utf8"))
14    s.sendall(bytes(hashmes, "utf8"))
15    data = s.recv(1024)
16    hashdata = s.recv(1024)
17    str_data = data.decode("utf8")
18    str_hashdata = hashdata.decode("utf8")
19    print('Server Response: ', data.decode("utf8"))
20    checkhash = hashlib.md5((str_data + key).encode()).hexdigest()
21    if str_hashdata == checkhash:
22        print("message from server is OK")
23    else:
24        print("the received message from server has lost its integrity")
25    if not data:
26        break
```

- Trao đổi thông tin



```
PS C:\Users\NguyenHuyQuang> & C:/Users/Documents/Environments/Python/python.exe c:/Users/NguyenHuyQuang/Desktop/Socket/Serverkey.py
Connected by ('127.0.0.1', 50878)
Client: Hi, Iam B20DCAT144 Client
the received message from client has lost its integrity
Server: Hi, Iam B20DCAT144 Server
█
```



```
PS C:\Users\NguyenHuyQuang> python -u "c:/Users/NguyenHuyQuang/Desktop/Socket/Clientkey.py"
Client: Hi, Iam B20DCAT144 Client
Server Response: Hi, Iam B20DCAT144 Server
the received message from server has lost its integrity
Client: █
```