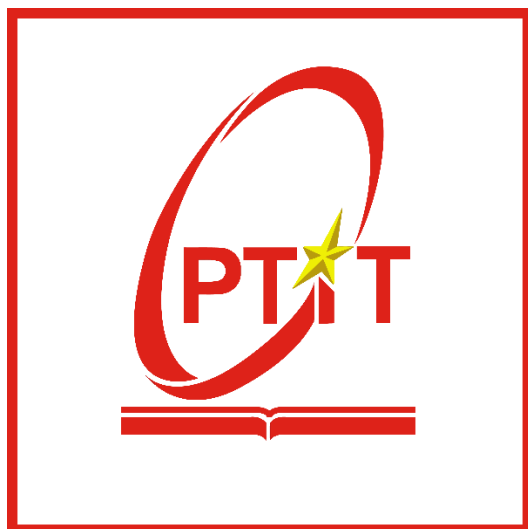


HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA AN TOÀN THÔNG TIN



Môn học: THỰC TẬP CƠ SỞ
BÁO CÁO BÀI THỰC HÀNH SỐ 15
LẬP TRÌNH CLIENT/SERVER ĐỂ TRAO ĐỔI THÔNG TIN
AN TOÀN

Sinh viên thực hiện: Lê Anh Tuấn

Mã sinh viên: B21DCAT205

Giảng viên: Ninh Thị Thu Trang

~ Hà Nội, tháng 5/2024 ~

Mục Lục

1	Mục đích	2
2	Nội dung thực hành.....	2
2.1	Lý thuyết	2
2.2	Nội dung thực hành.	5
2.2.1	Lập trình client và server với TCP socket	5
2.2.2	Trao đổi thông điệp giữa client và server và đảm bảo tính toàn vẹn của thông điệp khi trao đổi.....	6
3	Kết luận	9
4	Tài liệu tham khảo	9

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

1 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.

2 Nội dung thực hành

2.1 Lý thuyết

Socket là phương tiện hiệu quả để xây dựng các ứng dụng theo kiến trúc Client-Server. Lập trình socket TCP là một phần quan trọng của việc phát triển ứng dụng mạng trong môi trường máy tính. Sockets cung cấp một giao diện lập trình để gửi và nhận dữ liệu qua mạng.

Socket là một điểm cuối của liên kết giao tiếp hai chiều giữa hai chương trình chạy trên mạng. Nghĩa là một socket được sử dụng để cho phép 1 chương trình giao tiếp với 1 chương khác.

Các lớp Socket được sử dụng để tiến hành kết nối giữa client và server. Nó được ràng buộc với một cổng port (thể hiện là một con số cụ thể) để các tầng TCP (TCP Layer) có thể định danh ứng dụng mà dữ liệu sẽ được gửi tới.

Lập trình Socket cung cấp các xử lý cần thiết cho giao tiếp giữa các ứng dụng giữa các ứng dụng, trên hệ thống cục bộ hoặc trong môi trường mạng TCP/IP phân tán. Khi kết nối ngang hàng được thiết lập, bộ mô tả socket được sử dụng để xác định kết nối.

Một đầu của kết nối ngang hàng của ứng dụng mạng phân tán dựa trên TCP/IP được mô tả bởi ổ cắm được xác định bởi:

Địa chỉ Internet

Giao thức giao tiếp: UDP, TCP

Số cổng (Port)

Các ứng dụng Socket thường là các ứng dụng C hoặc C++ bằng cách sử dụng Socket API. Một số ngôn ngữ khác như Java, Python cũng cung cấp Socket API. Các ứng dụng Client/Server dựa trên Java, Python khai thác các dịch vụ Socket đó.

Các giai đoạn trong lập trình Socket:

Giai đoạn 1: Server tạo Socket, gán số hiệu cổng và lắng nghe yêu cầu nối kết. Server sẵn sàng phục vụ Client.socket(): Server yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển.

bind(): Server yêu cầu gán số hiệu cổng (port) cho socket.

listen(): Server lắng nghe các yêu cầu nối kết từ các client trên cổng đã được gán.

Giai đoạn 2: Client tạo Socket, yêu cầu thiết lập một nối kết với Server.

socket(): Client yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển, thông thường hệ thống tự động gán một số hiệu cổng còn rảnh cho socket của Client.

connect(): Client gửi yêu cầu nối kết đến server có địa chỉ IP và Port xác định.

accept(): Server chấp nhận nối kết của client, khi đó một kênh giao tiếp ảo được hình thành, Client và server có thể trao đổi thông tin với nhau thông qua kênh ảo này.

Giai đoạn 3: Trao đổi thông tin giữa Client và Server.

Sau khi chấp nhận yêu cầu nối kết, thông thường server thực hiện lệnh read() và ngừng cho đến khi có thông điệp yêu cầu (Request Message) từ client gửi đến.

Server phân tích và thực thi yêu cầu. Kết quả sẽ được gửi về client bằng lệnh write().

Sau khi gửi yêu cầu bằng lệnh write(), client chờ nhận thông điệp kết quả (ReplyMessage) từ server bằng lệnh read().

Giai đoạn 4: Kết thúc phiên làm việc.

Các câu lệnh read(), write() có thể được thực hiện nhiều lần (ký hiệu bằng hình ellipse).

Kênh ảo sẽ bị xóa khi Server hoặc Client đóng socket bằng lệnh close().

Ứng dụng của lập trình Socket: Lập trình socket là một phần quan trọng của việc phát triển ứng dụng mạng, cho phép các ứng dụng giao tiếp và truyền thông dữ liệu qua mạng. Dưới đây là một số ứng dụng phổ biến của lập trình socket:

Truyền thông giữa Client và Server: Socket được sử dụng để thiết lập kết nối giữa client và server trong kiến trúc client-server. Các ứng dụng web, email, game online, chat, và nhiều ứng dụng mạng khác đều sử dụng socket để gửi và nhận dữ liệu giữa các thành phần.

Real-time Applications: Các ứng dụng thời gian thực như video call, voice call, streaming media sử dụng socket để truyền dữ liệu một cách liên tục và ngay lập tức.

P2P (Peer-to-Peer) Networking: Socket cũng được sử dụng trong các ứng dụng P2P để thiết lập kết nối trực tiếp giữa các máy tính mà không thông qua một server trung gian.

Transfer Files:

Các ứng dụng chia sẻ file, FTP (File Transfer Protocol), và các dịch vụ cloud sử dụng socket để truyền tải dữ liệu giữa máy tính.

Network Protocols: Các giao thức mạng như TCP, UDP, HTTP, và SMTP đều sử dụng socket để thực hiện việc gửi và nhận dữ liệu trên mạng.

IoT (Internet of Things): Trong các hệ thống IoT, socket được sử dụng để truyền thông giữa các thiết bị thông minh để thu thập dữ liệu hoặc điều khiển các thiết bị từ xa.

Multiplayer Games: Các trò chơi trực tuyến đa người chơi thường sử dụng socket để gửi và nhận dữ liệu giữa các máy tính của người chơi.

Broadcasting và Multicasting: Socket cung cấp khả năng broadcasting và multicasting, cho phép gửi thông điệp đến nhiều người nghe (multicast) hoặc tất cả mọi người trong mạng (broadcast).

Remote Procedure Call (RPC): RPC sử dụng socket để gửi và nhận các yêu cầu giữa các máy tính trong một môi trường phân tán.

Synchronization và Communication giữa Processes: Trong hệ thống đa tiến trình, socket cũng được sử dụng để đồng bộ và giao tiếp giữa các tiến trình.

Lập trình socket cung cấp một cơ sở cho việc xây dựng nhiều loại ứng dụng mạng đa dạng, từ những ứng dụng đơn giản đến các hệ thống phức tạp.

Ưu điểm và hạn chế của lập trình socket

Ưu điểm của lập trình socket:

Đa dạng và Linh hoạt: Lập trình socket cung cấp một cơ sở linh hoạt cho việc phát triển ứng dụng mạng đa dạng từ các ứng dụng nhỏ đến các hệ thống lớn.

Hiệu suất cao: Sử dụng socket giúp tối ưu hóa hiệu suất bởi vì nó cho phép truyền dữ liệu trực tiếp giữa các thiết bị mà không cần nhiều lớp trung gian.

Real-time Communication: Lập trình socket rất hiệu quả cho việc xây dựng các ứng dụng thời gian thực như video call, streaming, và trò chơi trực tuyến.

Độ ổn định và Tin cậy: Các giao thức như TCP (Transmission Control Protocol) sử dụng lập trình socket để cung cấp truyền dữ liệu đáng tin cậy, đảm bảo dữ liệu được gửi và nhận đúng định dạng và không bị mất.

Khả năng Mở rộng: Lập trình socket cho phép mở rộng dễ dàng khi cần thêm các chức năng hoặc tăng cường khả năng xử lý.

Hạn chế của lập trình socket:

Phức tạp trong Quản lý Kết nối: Quản lý kết nối và xử lý lỗi có thể trở nên phức tạp, đặc biệt là trong các ứng dụng có số lượng lớn các client và kết nối.

Bảo mật: Lập trình socket có thể đối mặt với các vấn đề bảo mật như nguy cơ tấn công từ xa và đánh cắp dữ liệu nếu không được triển khai một cách chặt chẽ.

Khả năng Mất Kết nối: Trong môi trường mạng không ổn định, kết nối có thể mất mát, và việc xử lý mất kết nối có thể là một thách thức.

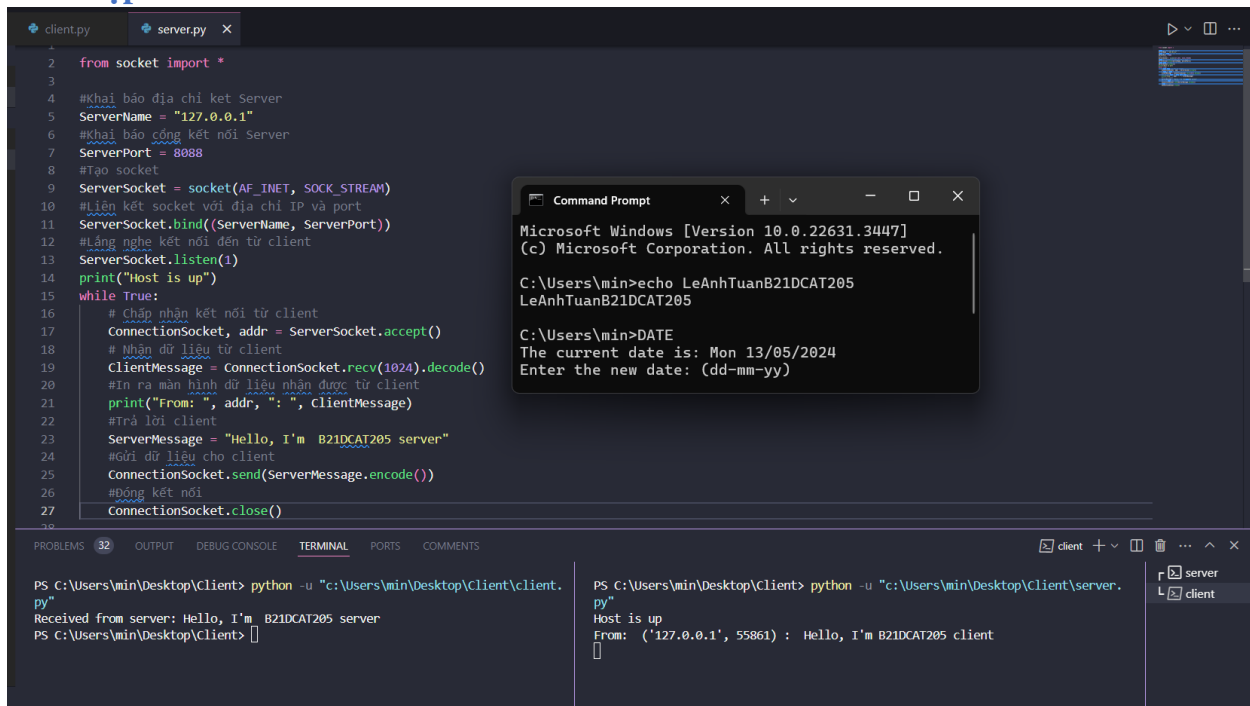
Hiệu năng của UDP: Mặc dù UDP (User Datagram Protocol) nhanh hơn TCP, nhưng nó không đảm bảo truyền dữ liệu đúng định dạng và không mất mát, điều này có thể là một hạn chế trong những ứng dụng đòi hỏi độ tin cậy cao.

Khó khăn trong Debug và Testing: Xử lý lỗi và debugging có thể khá phức tạp trong lập trình socket, đặc biệt là trong các ứng dụng lớn và phức tạp.

Tuy nhiên, nếu được triển khai đúng cách và với sự quản lý kết nối tốt, lập trình socket vẫn là một công cụ mạnh mẽ để xây dựng các ứng dụng mạng đa dạng và hiệu quả.

2.2 Nội dung thực hành.

2.2.1 Lập trình client và server với TCP socket



```
client.py
server.py

from socket import *
#Khai báo địa chỉ kết Server
ServerName = "127.0.0.1"
#Khai báo cổng kết nối Server
ServerPort = 8088
#Tạo socket
ServerSocket = socket(AF_INET, SOCK_STREAM)
#liên kết socket với địa chỉ IP và port
ServerSocket.bind((ServerName, ServerPort))
#lắng nghe kết nối đến từ client
ServerSocket.listen(1)
print("Host is up")
while True:
    # Chấp nhận kết nối từ client
    ConnectionSocket, addr = ServerSocket.accept()
    # Nhận dữ liệu từ client
    ClientMessage = ConnectionSocket.recv(1024).decode()
    #In ra màn hình dữ liệu nhận được từ client
    print("From: ", addr, ": ", ClientMessage)
    #Trả lời client
    ServerMessage = "Hello, I'm B21DCAT205 server"
    #Gửi dữ liệu cho client
    ConnectionSocket.send(ServerMessage.encode())
    #Đóng kết nối
    ConnectionSocket.close()

C:\Users\min>echo LeAnhTuanB21DCAT205
LeAnhTuanB21DCAT205

C:\Users\min>DATE
The current date is: Mon 13/05/2024
Enter the new date: (dd-mm-yy)
```

```
PS C:\Users\min\Desktop\Client> python -u "c:\Users\min\Desktop\Client\client.py"
Received from server: Hello, I'm B21DCAT205 server
PS C:\Users\min\Desktop\Client>

PS C:\Users\min\Desktop\Client> python -u "c:\Users\min\Desktop\Client\server.py"
Host is up
From: ('127.0.0.1', 55861) : Hello, I'm B21DCAT205 client
```

Hình 1: Mã nguồn của server và chạy thành công server


```
1 from socket import *
2 from hashlib import *
3 # Khai báo địa chỉ và cổng kết nối Server
4 ServerName = "127.0.0.1"
5 ServerPort = 8088
6 # Tạo ra một key để kiểm tra tính toàn vẹn của dữ liệu
7 key = "B21DCAT205"
8 # Tạo socket
9 ServerSocket = socket(AF_INET, SOCK_STREAM)
10 ServerSocket.bind((ServerName, ServerPort))
11 ServerSocket.listen(1)
12 print("Host is up")
13 while True:
14     # Chấp nhận kết nối từ client
15     ConnectionSocket, addr = ServerSocket.accept()
16     # Nhận dữ liệu từ client
17     ClientMessage = ConnectionSocket.recv(1024).decode()
18     ClientHash = ConnectionSocket.recv(1024).decode()
19     # In ra màn hình dữ liệu nhận được từ client
20     print("From: ", addr, ":", ClientMessage)
21     # Trả lời client
22     ServerMessage = "Hello, I'm B21DCAT205 server"
23
24     if ( ClientHash != sha512(ClientMessage.encode("utf-8") + key.encode("utf-8")).hexdigest() ):
25         ServerMessageError = "The received message has lost integrity"
26         ConnectionSocket.send(ServerMessageError.encode())
27     else:
28         ConnectionSocket.send(ServerMessage.encode())
29     ConnectionSocket.close()
30
```

Terminal output:

```
y"
<built-in method connect of socket object at 0x0000024B069C27A0>
Send message to server Hello, I'm B21DCAT205 client
Send hashed to server 8489faed8becbf7ae3fd464f0d05cafa9b745879e315ece788a221f8
3beef4f3946b3e4b3712ab9e34617463008742a16ae5c6801f069dd9a9ce4c4bce198d2
Received from server: Hello, I'm B21DCAT205 server
PS C:\Users\min\Desktop\Client>
```

Command Prompt output:

```
Microsoft Windows [Version 10.0.22631.3447]
(c) Microsoft Corporation. All rights reserved.

C:\Users\min>echo LeAnhTuanB21DCAT205
LeAnhTuanB21DCAT205

C:\Users\min>DATE
The current date is: Mon 13/05/2024
Enter the new date: (dd-mm-yy)
```

Hình 4: Mã nguồn của server và chạy thành công server

```
1 from socket import *
2 from hashlib import *
3 # Khai báo địa chỉ kết nối Server
4 ServerName = "127.0.0.1"
5 # Khai báo cổng kết nối Server
6 ServerPort = 8088
7 Password = "B21DCAT205"
8
9 # Tạo socket
10 ClientSocket = socket(AF_INET, SOCK_STREAM)
11 # Kết nối đến server
12 ClientSocket.connect((ServerName, ServerPort))
13 print(ClientSocket.connect)
14 # Nhập dữ liệu từ bàn phím
15 ClientMessage = "Hello, I'm B21DCAT205 client"
16 hashed = sha512(ClientMessage.encode("utf-8") + Password.encode("utf-8")).hexdigest()
17 print("Send message to server", ClientMessage)
18 ClientSocket.send(ClientMessage.encode())
19 print("Send hashed to server", hashed)
20 ClientSocket.send(hashed.encode())
21 # Nhận dữ liệu từ server
22 ServerMessage = ClientSocket.recv(1024).decode()
23 # In ra màn hình dữ liệu nhận được từ server
24 print("Received from server:", ServerMessage)
25 # Đóng kết nối
26 ClientSocket.close()
27
28
```

Terminal output:

```
y"
<built-in method connect of socket object at 0x0000024B069C27A0>
Send message to server Hello, I'm B21DCAT205 client
Send hashed to server 8489faed8becbf7ae3fd464f0d05cafa9b745879e315ece788a221f8
3beef4f3946b3e4b3712ab9e34617463008742a16ae5c6801f069dd9a9ce4c4bce198d2
Received from server: Hello, I'm B21DCAT205 server
PS C:\Users\min\Desktop\Client>
```

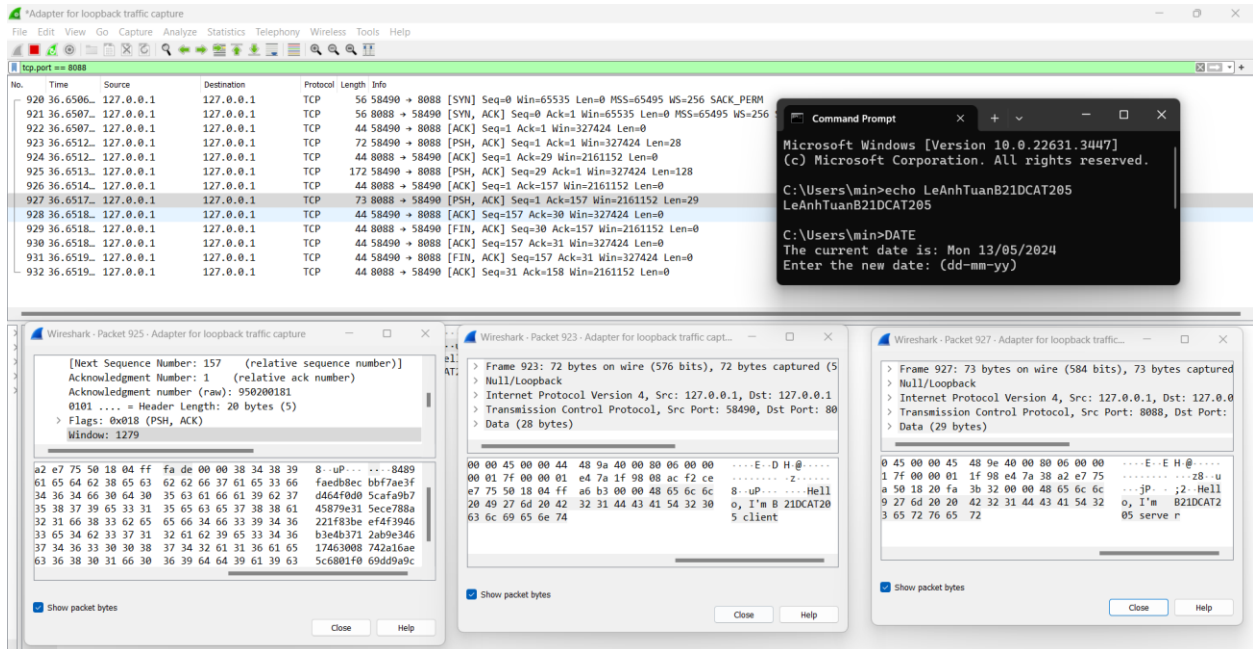
Command Prompt output:

```
Microsoft Windows [Version 10.0.22631.3447]
(c) Microsoft Corporation. All rights reserved.

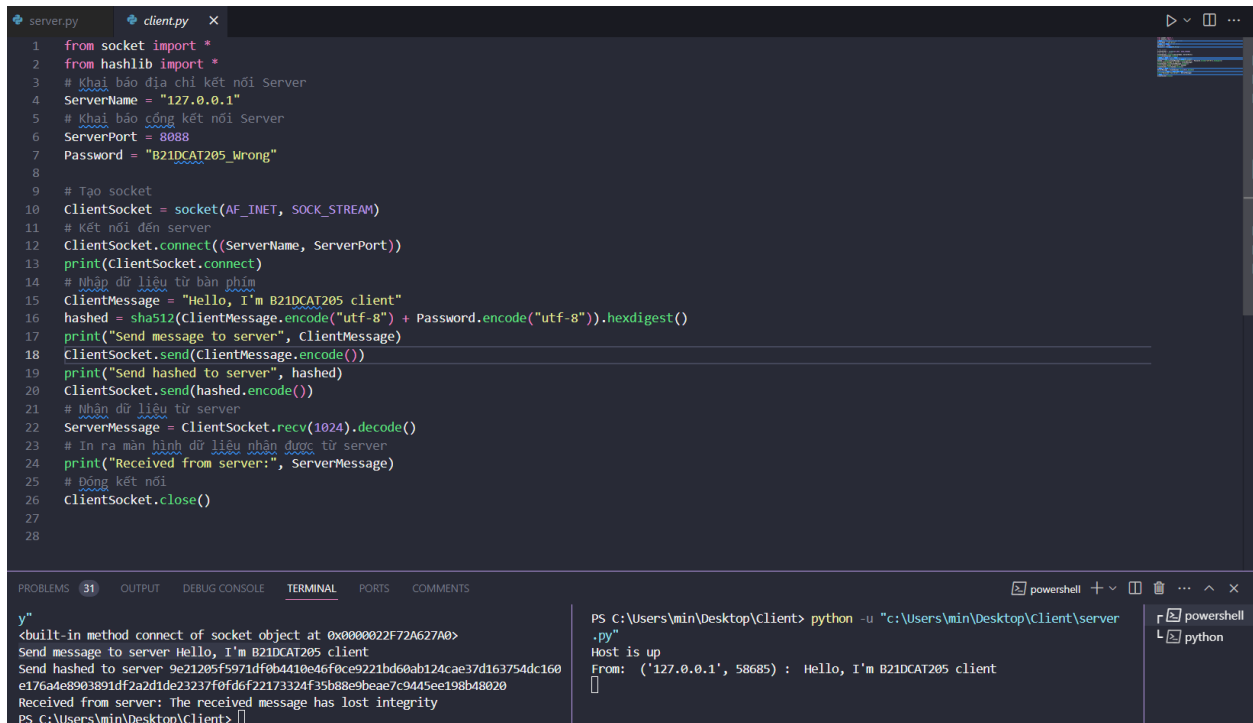
C:\Users\min>echo LeAnhTuanB21DCAT205
LeAnhTuanB21DCAT205

C:\Users\min>DATE
The current date is: Mon 13/05/2024
Enter the new date: (dd-mm-yy)
```

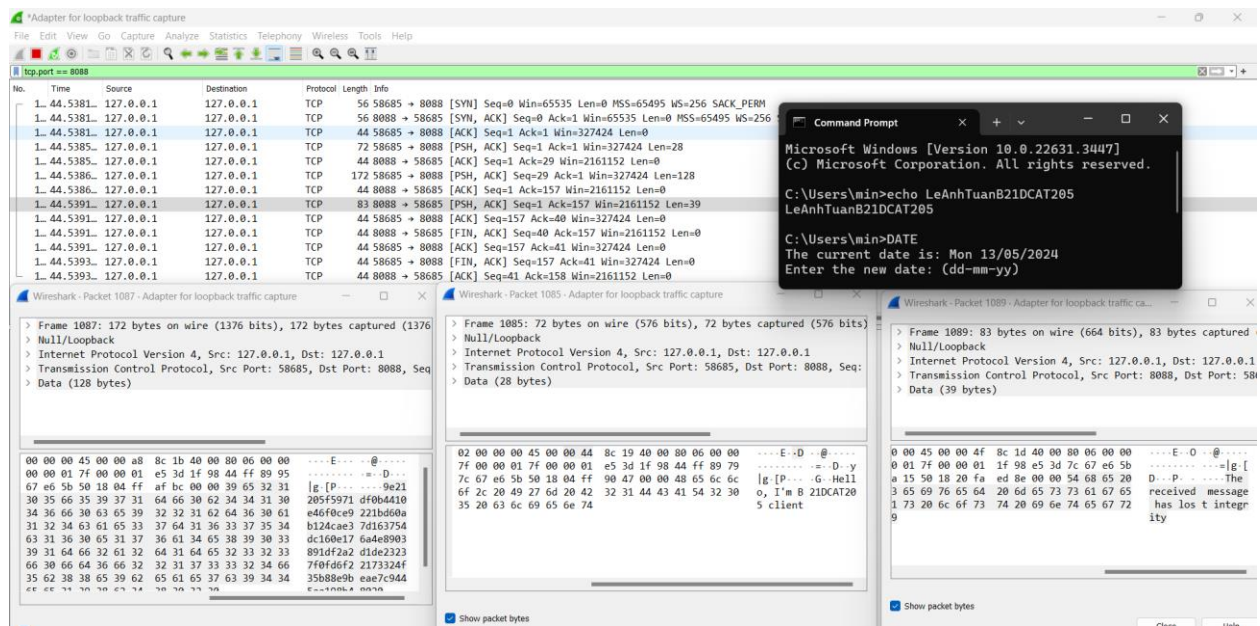
Hình 5: Mã nguồn của client và chạy thành công client



Hình 6: Bắt các gói tin bằng Wireshark: (gói tin chứa hàm băm của client, gói tin client gửi tới và gói tin server trả lời)



Hình 7: Thay đổi key của client, chạy lại client, và nhận được thông báo “The received message has lost its integrity.”:



Hình 8: Bắt các gói tin bằng Wireshark: (gói tin hàm băm của client gửi tới, gói tin client gửi đến, gói tin thông báo “The received message has lost its integrity” từ server):

3 Kết luận

- Qua bài báo cáo này, ta đã cùng nhau tìm hiểu về cách lập trình socket cũng như cách hoạt động của nó. Kết quả đạt được sau bài báo cáo này đó là ta đã bước đầu cơ bản hiểu biết về lập trình Socket.

4 Tài liệu tham khảo

- Chapter 2: Application Layer V8.1 (9/2020)
http://gaia.cs.umass.edu/kurose_ross/ppt.php