**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA AN TOÀN THÔNG TIN**

# BÁO CÁO BÀI THỰC HÀNH

# HỌC PHẦN: THỰC TẬP CƠ SỞ

# MÃ HỌC PHẦN: INT13147

## BÀI THỰC HÀNH 4.1

## LẬP TRÌNH CLIENT/SERVER ĐỂ TRAO ĐỔI THÔNG TIN

Sinh viên thực hiện:

B22DCAT206 PHẠM ĐỨC NAM

Giảng viên hướng dẫn: Đỗ Xuân Chợ

**HỌC KỲ 2 NĂM HỌC 2024-202**

# Mục lục

# Danh mục hình ảnh

# 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. Cơ cở lý thuyết: Tìm hiểu về các khái niệm liên quan đến lập trình socket với TCP

### a. Socket là gì?

Socket là điểm cuối (end-point) trong liên kết truyền thông hai chiều giữa Client và Server, cho phép gửi và nhận dữ liệu qua mạng Internet.

Socket được ràng buộc với một cổng (port) để tầng TCP định danh ứng dụng mà dữ liệu sẽ được gửi tới.

Socket là giao diện lập trình ứng dụng mạng, hỗ trợ kết nối hai tiến trình (process) để giao tiếp thông qua TCP/IP hoặc UDP.


*Socket hoạt động thông qua các tầng TCP hoặc TCP Layer định danh ứng dụng, từ đó truyền dữ liệu thông qua sự ràng buộc với một cổng port*

**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.

## b. Tại sao người dùng lại cần dùng đến socket?

Socket là một công cụ quan trọng trong lập trình mạng vì nó cung cấp một cách tiêu chuẩn và hiệu quả để thiết lập và quản lý các kết nối mạng giữa các thiết bị và ứng dụng. Dưới đây là những lý do chính khiến socket trở thành một thành phần không thể thiếu trong các ứng dụng mạng:

Giao tiếp mạng: Socket cho phép các ứng dụng gửi và nhận dữ liệu qua mạng Internet hoặc mạng nội bộ. Điều này là nền tảng cho các ứng dụng như trò chuyện trực tuyến (chat), email, truyền tệp, hoặc các dịch vụ web. Ví dụ, khi bạn gửi một tin nhắn qua ứng dụng chat, socket được sử dụng để truyền dữ liệu từ thiết bị của bạn đến máy chủ và ngược lại.

Tính linh hoạt: Socket hỗ trợ nhiều giao thức mạng khác nhau, chẳng hạn như TCP (đảm bảo truyền dữ liệu đáng tin cậy, có kiểm soát lỗi) và UDP (truyền dữ liệu nhanh, không đảm bảo toàn vẹn). Điều này cho phép các nhà phát triển lựa chọn giao thức phù hợp với yêu cầu cụ thể của ứng dụng, từ các ứng dụng yêu cầu độ chính xác cao (như truyền tệp) đến các ứng dụng ưu tiên tốc độ (như streaming video).

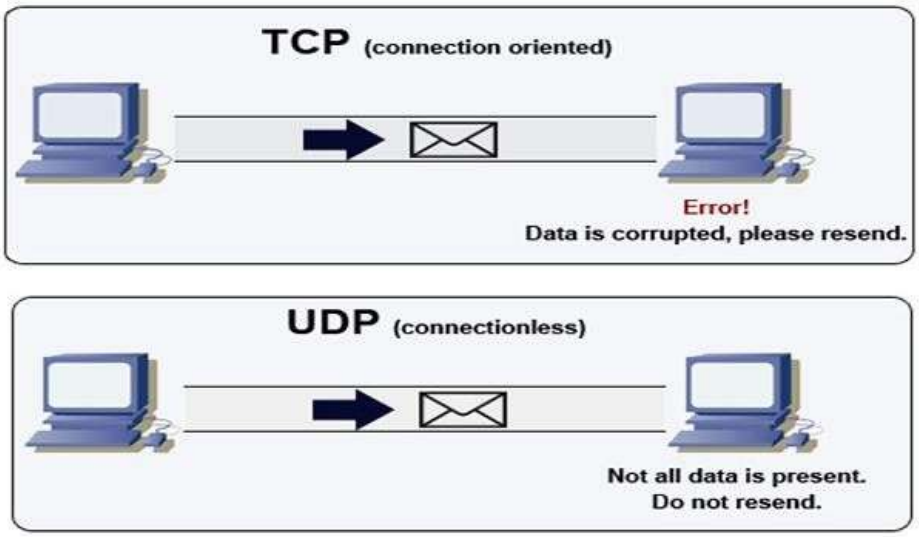
Phân phối dữ liệu: Socket cho phép dữ liệu được gửi từ một nguồn đến nhiều đích hoặc từ nhiều nguồn đến một đích. Điều này rất quan trọng trong các ứng dụng như phát trực tuyến (streaming), trò chơi đa người chơi trực tuyến, hoặc các hệ thống phân tán. Ví dụ, trong một ứng dụng phát video trực tiếp, socket giúp phân phối luồng dữ liệu từ máy chủ đến nhiều người xem đồng thời.

Tương thích đa nền tảng: Socket được hỗ trợ trên nhiều hệ điều hành (Windows, Linux, macOS) và các thiết bị (máy tính, điện thoại di động, thiết bị nhúng). Điều này giúp các nhà phát triển dễ dàng xây dựng các ứng dụng mạng hoạt động trên nhiều nền tảng mà không cần thay đổi mã nguồn cơ bản.

Kiểm soát độ trễ: Socket cho phép các nhà phát triển tối ưu hóa việc truyền và nhận dữ liệu để giảm độ trễ, điều này đặc biệt quan trọng trong các ứng dụng thời gian thực như trò chơi trực tuyến, gọi video, hoặc hệ thống điều khiển từ xa. Bằng cách điều chỉnh các tham số socket, nhà phát triển có thể cân bằng giữa tốc độ và độ tin cậy của dữ liệu.

Tóm lại, socket là một công cụ không thể thiếu trong việc phát triển các ứng dụng mạng, cung cấp khả năng giao tiếp linh hoạt, hiệu quả và đáng tin cậy giữa các thiết bị trên Internet.

Điều kiện để Socket hoạt động?




Chức năng của socket là kết nối giữa client và server thông qua TCP/IP và UDP để truyền và nhận giữ liệu qua Internet

Socket hoạt động dựa trên các giao thức mạng như TCP/IP hoặc UDP, và để socket có thể hoạt động, cần đáp ứng các điều kiện sau:

Thông tin kết nối: Socket yêu cầu thông tin về địa chỉ IP và số hiệu cổng (port) của cả hai ứng dụng (Client và Server) để thiết lập kết nối. Địa chỉ IP xác định thiết bị trên mạng, trong khi số hiệu cổng xác định ứng dụng cụ thể trên thiết bị đó.

Vị trí ứng dụng: Hai ứng dụng có thể chạy trên cùng một máy hoặc hai máy khác nhau. Nếu chạy trên cùng một máy, số hiệu cổng của hai ứng dụng phải khác nhau để tránh xung đột.

Giao thức mạng: Socket hoạt động dựa trên giao thức TCP hoặc UDP. Với TCP, kết nối được thiết lập thông qua quá trình bắt tay ba bước (three-way handshake), đảm bảo kết nối ổn định và dữ liệu được truyền chính xác. Với UDP, không cần thiết lập kết nối trước, nhưng dữ liệu có thể bị mất hoặc không đảm bảo thứ tự.

Socket chỉ hoạt động khi các điều kiện trên được đáp ứng, và các ứng dụng phải được cấu hình đúng để gửi và nhận dữ liệu qua mạng.

## c. Phân loại Socket?

Socket được phân loại dựa trên giao thức và cách sử dụng. Dưới đây là các loại socket chính:

- Theo giao thức:

Socket TCP (Transmission Control Protocol): Sử dụng giao thức TCP để thiết lập kết nối đáng tin cậy, đảm bảo dữ liệu được truyền đến đích chính xác và theo đúng thứ tự. TCP kiểm soát lỗi và đảm bảo tính toàn vẹn của dữ liệu, phù hợp với các ứng dụng như truyền tệp, email, hoặc các trang web.

Socket UDP (User Datagram Protocol): Sử dụng giao thức UDP, không thiết lập kết nối trước

và không đảm bảo dữ liệu đến đích chính xác. UDP phù hợp cho các ứng dụng cần tốc độ cao, như streaming video, trò chơi trực tuyến, hoặc truyền dữ liệu không yêu cầu độ tin cậy tuyệt đối.

- Theo cách sử dụng:

Socket Server: Được thiết kế để lắng nghe (listen) các kết nối đến từ Client và chấp nhận (accept) chúng. Socket Server thường được sử dụng trong các ứng dụng như máy chủ web, máy chủ chat, hoặc máy chủ trò chơi.

Socket Client: Được sử dụng để khởi tạo kết nối đến Socket Server và gửi/nhận dữ liệu. Socket Client thường được triển khai trong các ứng dụng như trình duyệt web, ứng dụng chat, hoặc các ứng dụng cần giao tiếp với máy chủ.

Việc lựa chọn loại socket phù hợp phụ thuộc vào yêu cầu cụ thể của ứng dụng, chẳng hạn như độ tin cậy, tốc độ, hoặc tính phức tạp của giao tiếp.

## d. Lập trình socket với TCP/IP

**Quy trình hoạt động**:

Máy chủ (Server) phải chạy trước, tạo socket để chấp nhận kết nối từ Client.

Client tạo socket TCP, chỉ định địa chỉ IP và cổng của Server.

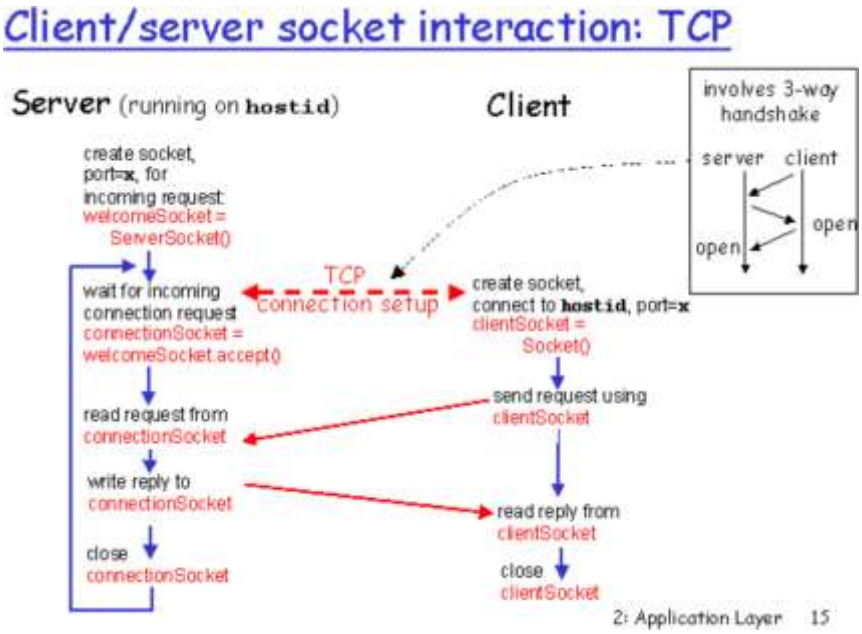
TCP thực hiện bắt tay 3 bước (three-way handshake) để thiết lập kết nối.

Server tạo socket mới dành riêng cho Client khi chấp nhận kết nối (phương thức accept()).

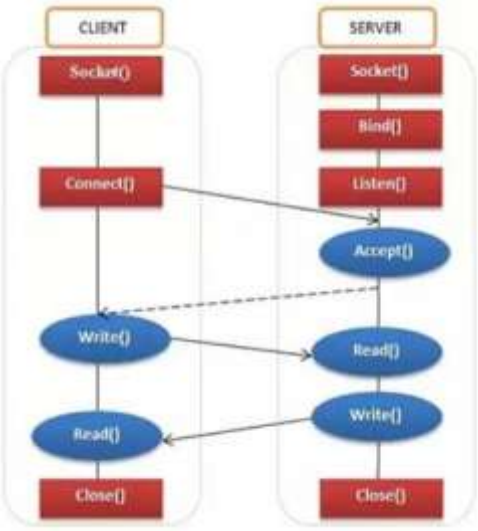
Khi tiến trình chủ đã chạy, lúc này tiến trình khách sẽ tạo ra 1 socket TCP để có thể  kết nối đến máy chủ. Trong khi máy khách đang tạo TCP socket, nó sẽ đặc tả địa chỉ IP, số cổng của tiến trình chủ.

Khi socket của tiến trình khách vừa được tạo, TCP trên máy khách sẽ tiến hành thực hiện quá trình bắt tay 3 bước và thiết lập kết nối TCP tới máy chủ.

Trong quá trình bắt tay 3 bước, khi tiến trình chủ nhận thấy tiến trình khách, nó sẽ tự tạo ra 1 socket mới chỉ dành riêng cho tiến trình khách đó. Khi được máy khách gõ cửa, chương trình kích hoạt với phương thức accept(). Cuối quá trình bắt tay 3 bước, một kết nối TCP tồn tại giữa socket của máy khách và socket của máy chủ.




Mô tả quá trình:




1. Trước tiên chúng ta sẽ tạo ra một máy chủ bằng cách mở một socket - Socket()

2. Sau đó chúng ta sẽ liên kết nó với một host hoặc một máy và một port - Bind()
3. Tiếp theo server sẽ bắt đầu lắng nghe trên port đó - Listen()
4. Yêu cầu kết nối từ client được gửi tới server - Connect()
5. Server sẽ accept yêu cầu từ client và sau đó kết nối được thiết lập - Accept()
6. Bây giờ cả hai đều có thể gửi và nhận tin tại thời điểm đó - Read() / Write()
7. Và cuối cùng khi hoàn thành chúng có thể đóng kết nối - Close()

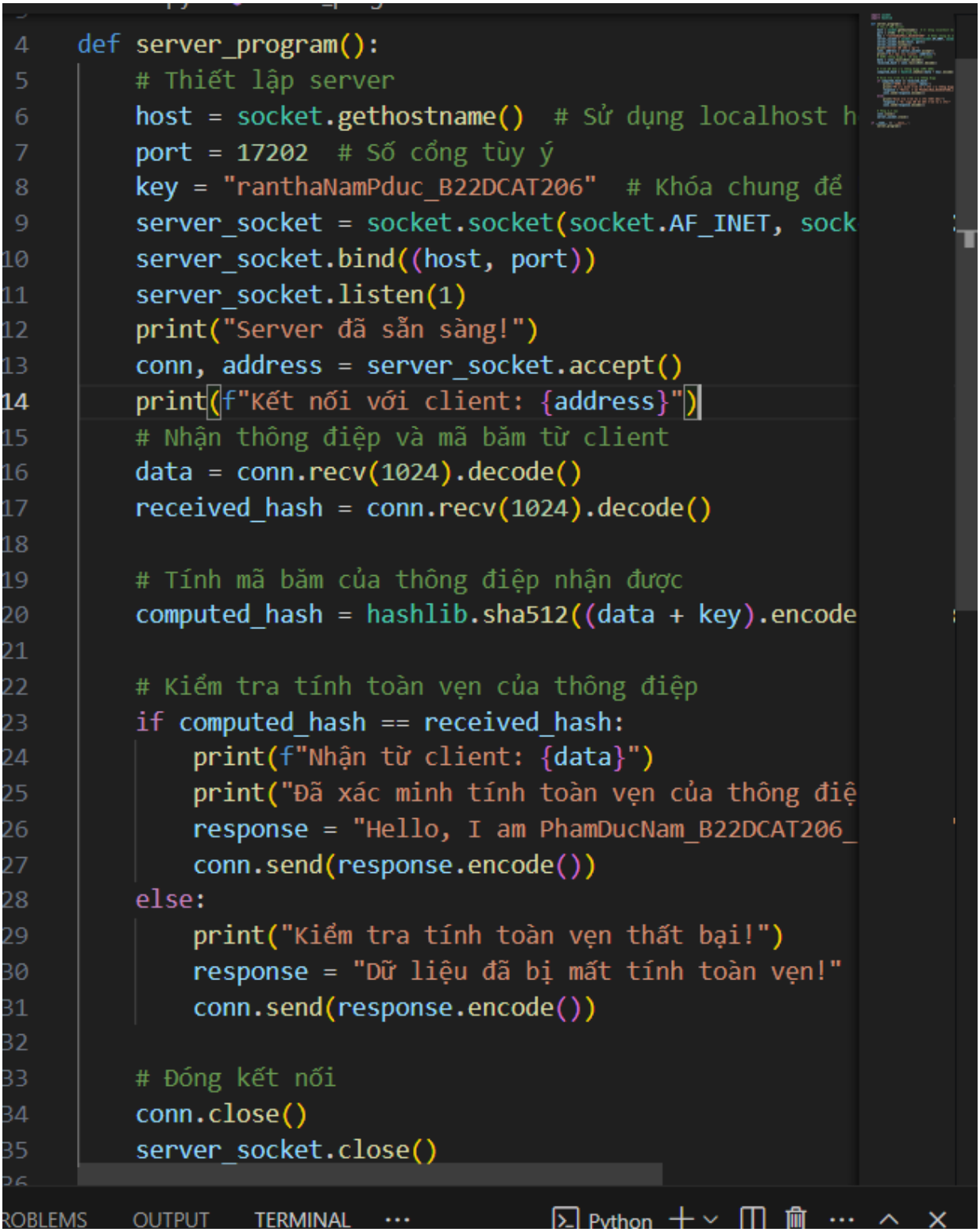
## 2.2. Các bước thực hiện

### 2.2.1. Chuẩn bị môi trường

- Môi trường Python hoặc Java để chạy được ứng dụng client/server đã lập trình.
- Phần mềm Wireshark

### 2.2.2. Lập trình client và server với TCP socket

- Lập trình client
- Lập trình server
- Chạy server sau đó chạy client
- Client gửi thông điệp cá nhận hóa cho server: “Hello, I am TruongAnhTuan_206_client.”
- Server nhận được hiển thị thông điệp nhận được và gửi lại client thông điệp: server gửi lại “Hello, I am TruongAnhTuan_B22DCAT206_server”



```python
def server_program():
    # Thiết lập server
    host = socket.gethostname()  # Sử dụng localhost h
    port = 17202  # Số cổng tùy ý
    key = "ranthaNamPduc_B22DCAT206"  # Khóa chung để
    server_socket = socket.socket(socket.AF_INET, sock
    server_socket.bind((host, port))
    server_socket.listen(1)
    print("Server đã sẵn sàng!")
    conn, address = server_socket.accept()
    print(f"Kết nối với client: {address}")
    # Nhận thông điệp và mã băm từ client
    data = conn.recv(1024).decode()
    received_hash = conn.recv(1024).decode()

    # Tính mã băm của thông điệp nhận được
    computed_hash = hashlib.sha512((data + key).encode

    # Kiểm tra tính toàn vẹn của thông điệp
    if computed_hash == received_hash:
        print(f"Nhận từ client: {data}")
        print("Đã xác minh tính toàn vẹn của thông điệ
        response = "Hello, I am PhamDucNam_B22DCAT206_
        conn.send(response.encode())
    else:
        print("Kiểm tra tính toàn vẹn thất bại!")
        response = "Dữ liệu đã bị mất tính toàn vẹn!"
        conn.send(response.encode())

    # Đóng kết nối
    conn.close()
    server_socket.close()
```

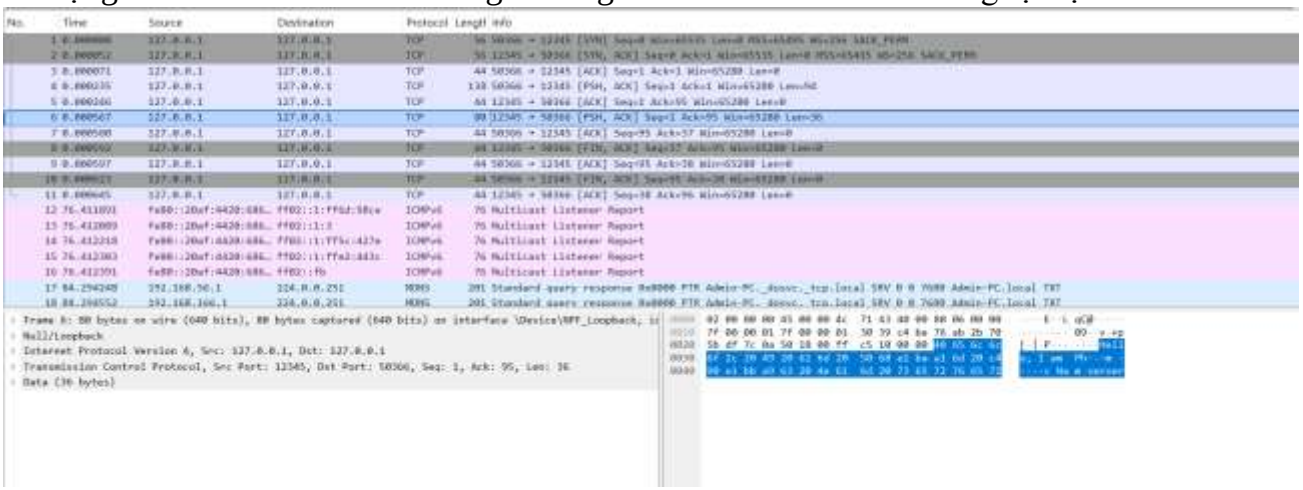


*Lập trình và chạy Server*

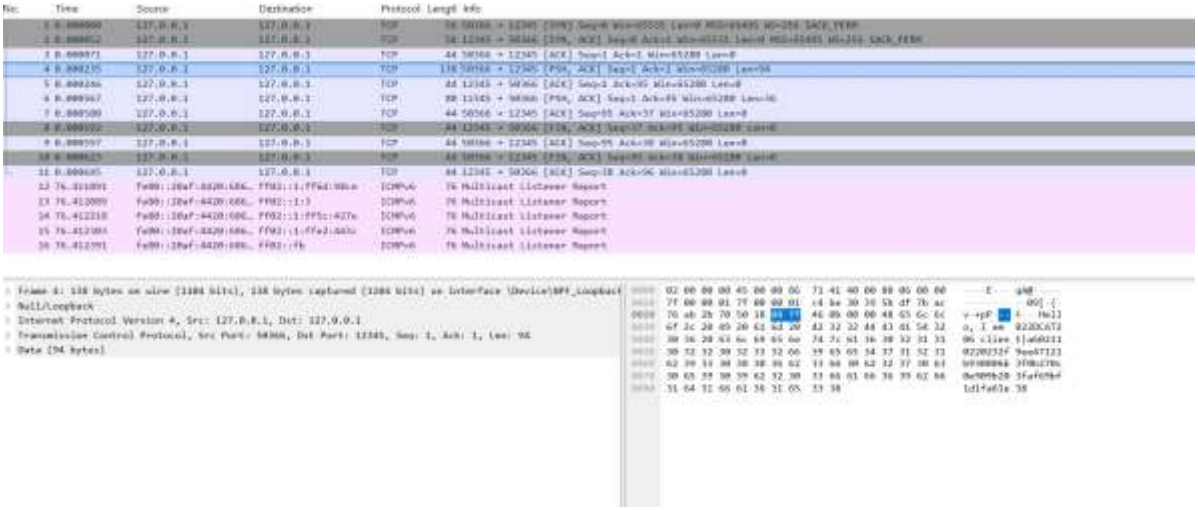
```python
import socket
import hashlib

# Thay bằng mã sinh viên và tên của bạn
STUDENT_ID = "12345678"
STUDENT_NAME = "Nguyen Van A"
SECRET_KEY = "mysecretkey"

def calculate_hash(message):
    # Tính giá trị băm của (thông điệp + key)
    return hashlib.sha256((message + SECRET_KEY).encode()).hexdigest()

def start_client():
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect(('localhost', 12345))
    print("Connected to server")

    # Gửi thông điệp kèm giá trị băm
    message = f"Hello, I am {STUDENT_ID} client"
    message_hash = calculate_hash(message)
    data_to_send = f"{message}|{message_hash}"
    client_socket.send(data_to_send.encode())
    print(f"Sent to server: {message}")
    print(f"Sent hash: {message_hash}")

    # Nhận phản hồi từ server
    response = client_socket.recv(1024).decode()
    print(f"Received from server: {response}")

    # Đóng kết nối
    client_socket.close()

if __name__ == "__main__":
    start_client()
```

*Lập trình và chạy Client*

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



*Bắt thông tin được gửi từ Server đến Client*



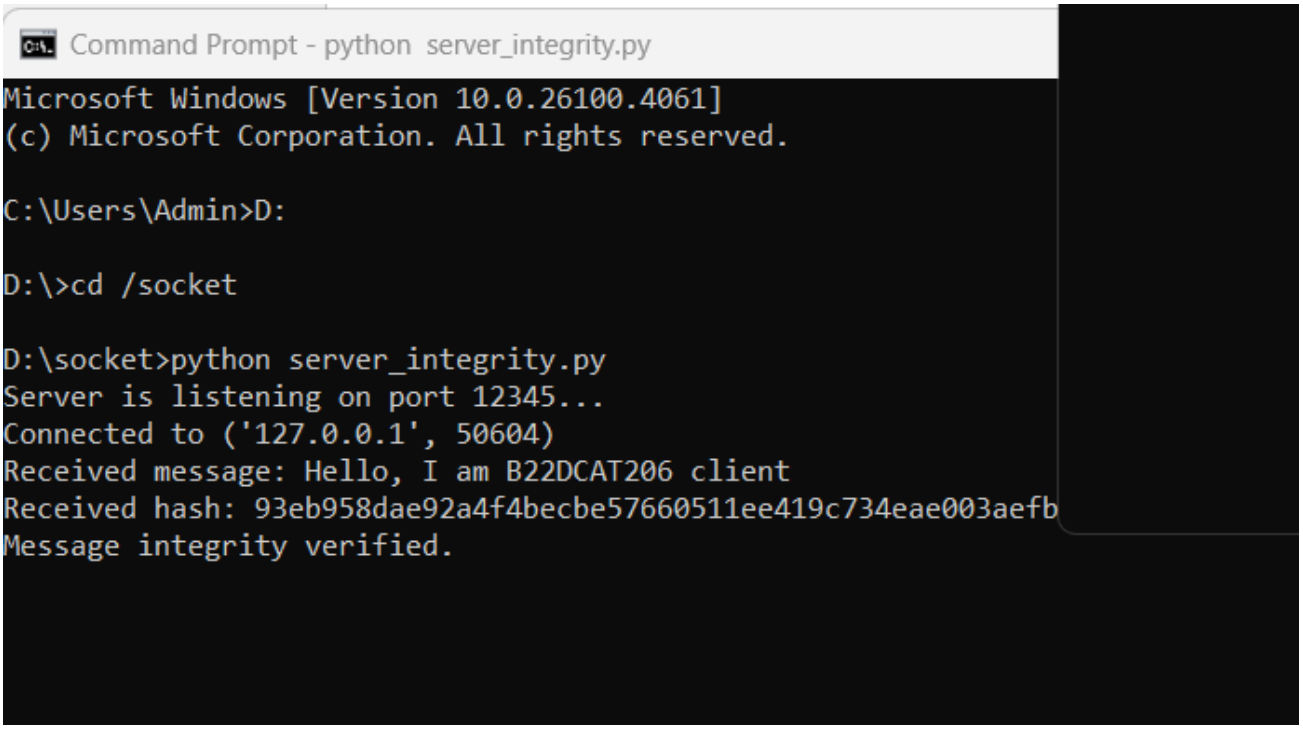
*Bắt thông tin được gửi từ Client đến Server*

### 2.2.3. 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.

```python
import socket
import hashlib

STUDENT_ID = "Phạm Đức Nam"
STUDENT_NAME = "B22DCAT206"
SECRET_KEY = "B22DCAT206"
def verify_integrity(message, received_hash):
    # Tính giá trị băm của (thông điệp + key)
    expected_hash = hashlib.sha256((message + SECRET_KEY).encode()).hexdigest()
    return expected_hash == received_hash
def start_server():
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind(('localhost', 12345))
    server_socket.listen(1)
    print("Server is listening on port 12345...")
    while True:
        client_socket, addr = server_socket.accept()
        print(f"Connected to {addr}")
        # Nhận dữ liệu từ client
        data = client_socket.recv(1024).decode()
        if not data:
            client_socket.close()
            continue
        # Tách thông điệp và giá trị băm
        try:
            message, received_hash = data.rsplit('|', 1)
            print(f"Received message: {message}")
            print(f"Received hash: {received_hash}")
```

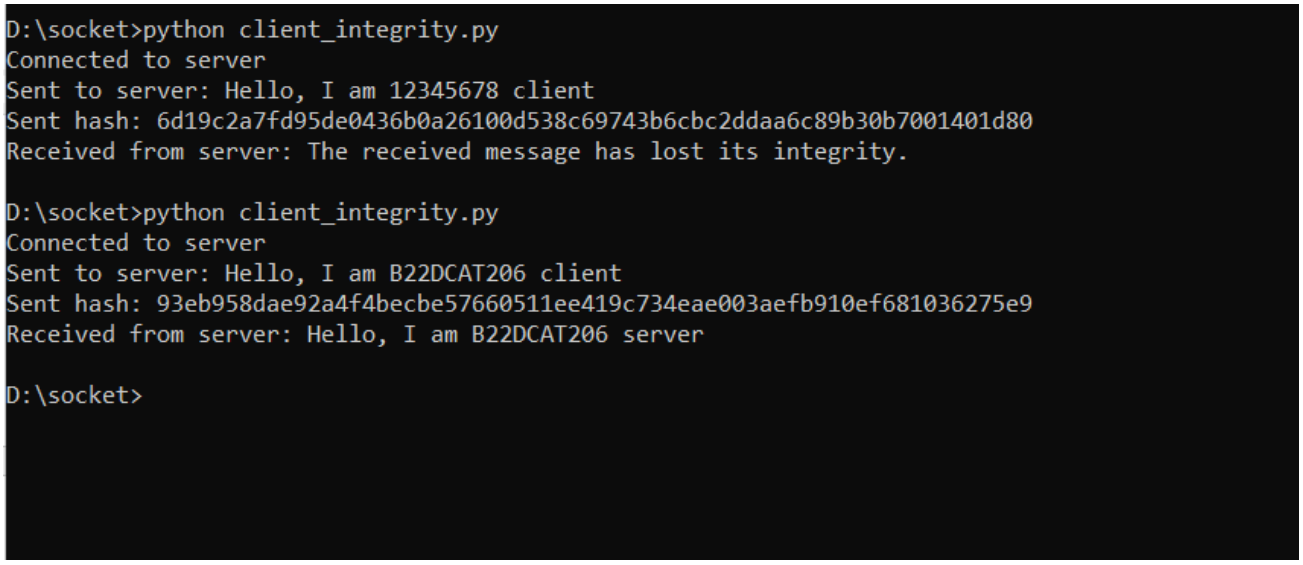
*Code của Server*




*Kết quả khi chạy*

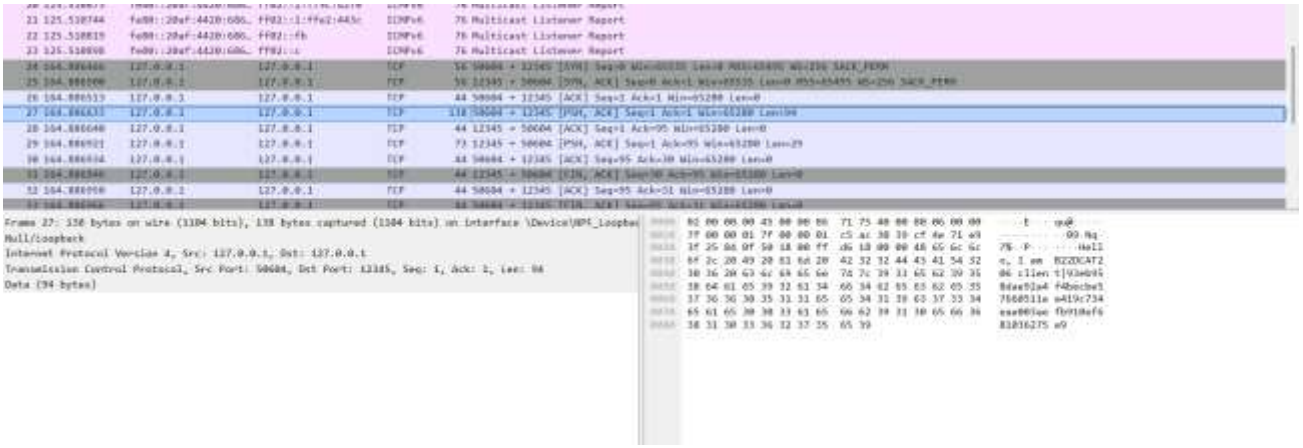
```python
def calculate_hash(message):
    # Tính giá trị băm của (thông điệp + key)
    return hashlib.sha256((message + SECRET_KEY).encode()).hexdigest()

def start_client():
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect(('localhost', 12345))
    print("Connected to server")

    # Gửi thông điệp kèm giá trị băm
    message = f"Hello, I am {STUDENT_ID} client"
    message_hash = calculate_hash(message)
    data_to_send = f"{message}|{message_hash}"
    client_socket.send(data_to_send.encode())
    print(f"Sent to server: {message}")
    print(f"Sent hash: {message_hash}")

    # Nhận phản hồi từ server
    response = client_socket.recv(1024).decode()
    print(f"Received from server: {response}")

    # Đóng kết nối
    client_socket.close()

if __name__ == "__main__":
    start_client()
```

*Code Client*

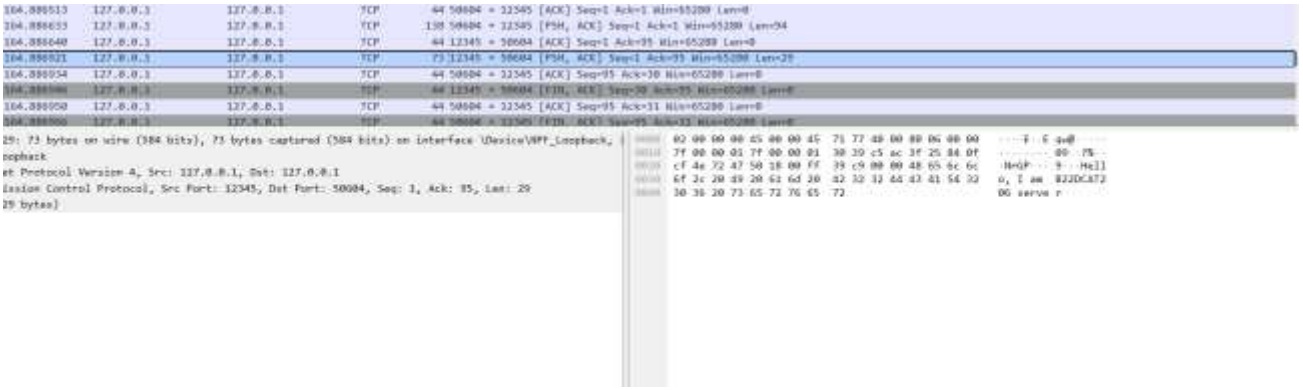



*Kết quả khi chạy Client*

– Bắt các bản tin trao đổi giữa Client và Server trong Wireshark trong trường hợp giống key.



*Client gửi thông điệp + key đã được mã hóa SHA512*



*Server gửi phản hồi đến Client sau khi nhận thông điệp + key không thay đổi*

– Thay đổi giá trị key tại client và thực hiện gửi lại, nếu không đáp ứng tính toàn vẹn cần thông báo: “The received message has lost its integrity.”

```python
import hashlib

# Thay bằng mã sinh viên và tên của bạn
STUDENT_ID = "B22DCAT206"
STUDENT_NAME = "Phạm Đức Nam"
SECRET_KEY = "WrongKey"

def calculate_hash(message):
    return hashlib.sha256((message + SECRET_KEY).encode()).hexdigest()

def start_client():
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect(('localhost', 12345))  # Thay bằng IP nếu cần
    print("Connected to server")
```

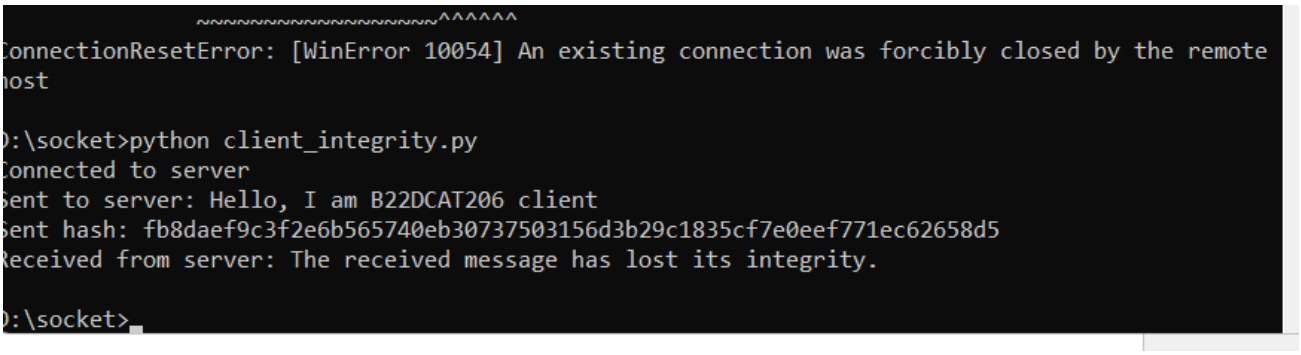
*Thay đổi key ở Client → Thông điệp mã hash thay đổi*

```
def calculate_hash(message):
    # Tính giá trị băm của (thông điệp + key)
    return hashlib.sha256((message + SECRET_KEY).encode()).hexdigest()

def start_client():
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect(('localhost', 12345))
    print("Connected to server")

    # Gửi thông điệp kèm giá trị băm
    message = f"Hello, I am {STUDENT_ID} client"
    message_hash = calculate_hash(message)
    data_to_send = f"{message}|{message_hash}"
    client_socket.send(data_to_send.encode())
    print(f"Sent to server: {message}")
    print(f"Sent hash: {message_hash}")

    # Nhận phản hồi từ server
    response = client_socket.recv(1024).decode()
    print(f"Received from server: {response}")

    # Đóng kết nối
    client_socket.close()

if __name__ == "__main__":
    start_client()
```

*Bên Server giữ nguyên*

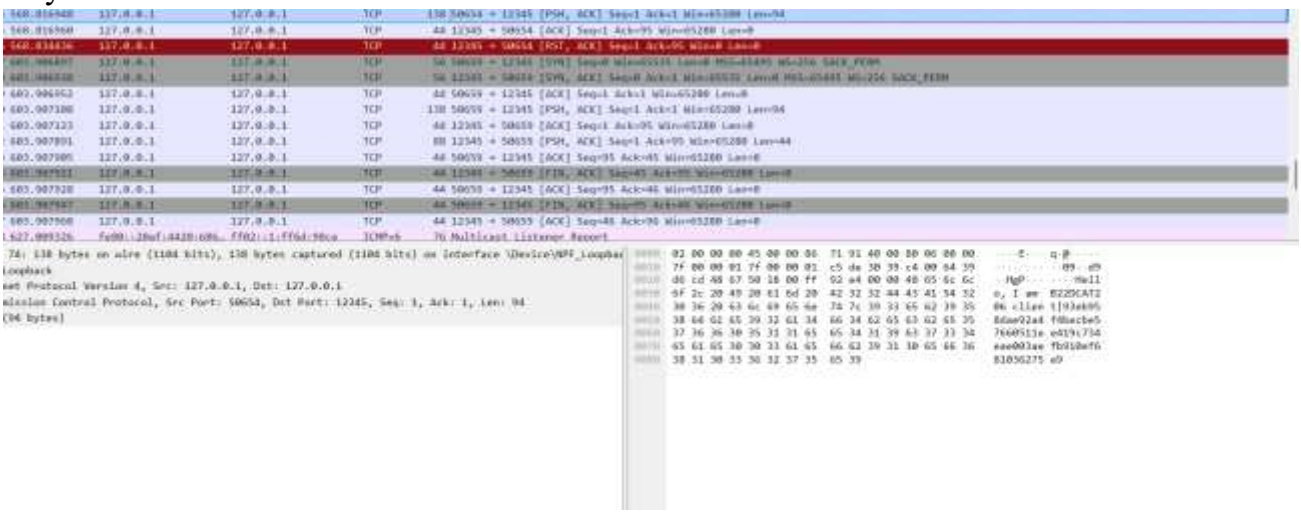
```
                ~~~~~~~~~~~~~~~^^
KeyboardInterrupt
^C
D:\socket>python server_integrity.py
Server is listening on port 12345...
Connected to ('127.0.0.1', 50659)
Received message: Hello, I am B22DCAT206 client
Received hash: fb8daef9c3f2e6b565740eb30737503156d3b29c1835cf7e0eef771ec62658d5
Message integrity check failed.
```

*Bên Server: mã hash của thông điệp đã bị thay đổi → Không toàn vẹn dữ liệu*

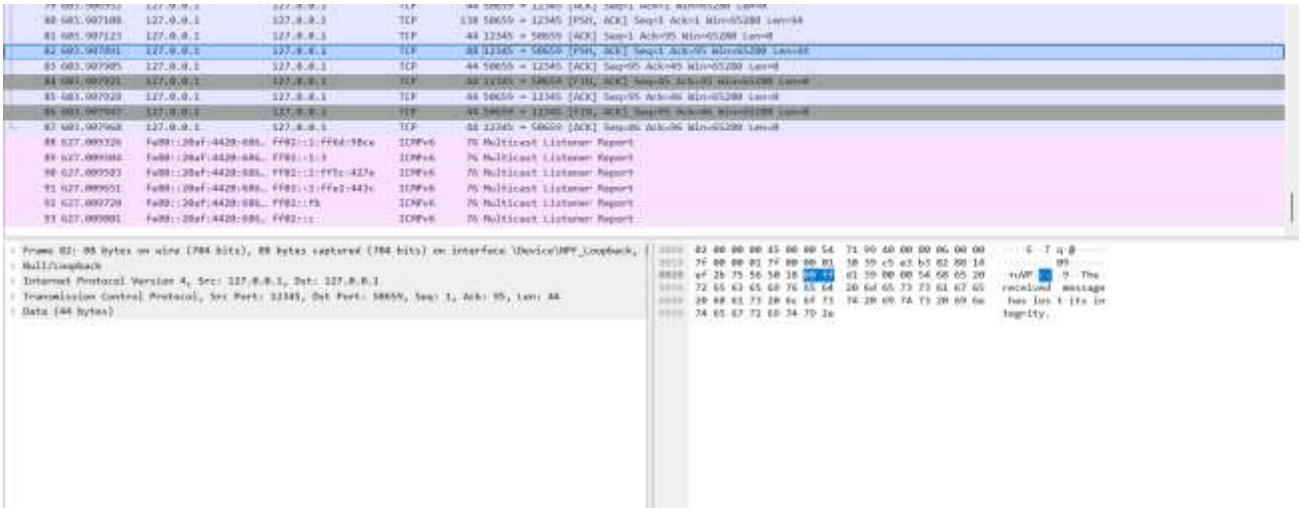



*Kết quả, bên Client nhận được thông báo rằng dữ liệu đã bị mất mác*

– Bắt được các bản tin trao đổi giữa client và server trong Wireshark trong trương hợp key bị thay đổi



*Khi truyền từ Client → Server, Mã hash đã bị thay đổi*



*Server phản hồi Client: dữ liệu đã mất mác*

## 3. Kết luận

- Lập trình và chạy thành công TCP Socket Client/Server
- Trao đổi thông điệp giữa client và server và đảm bảo tính toàn vẹn của thông điệp khi trao đổi
- Bắt được các bản tin trao đổi giữa client và server trong Wireshark

## 4. Tài liệu tham khảo

[1]. Tham khảo tài liệu: Chapter 2: Application Layer V8.1 (9/2020) tại địa chỉ http://gaia.cs.umass.edu/kurose_ross/ppt.php (chú ý ví dụ từ trang 105).

[2]. Hướng dẫn code Socket, xem ở đây