**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 AN TOÀN**

Sinh viên thực hiện: Ngô Quang Thắng – B22DCAT287

Giảng viên hướng dẫn: TS. Quản Trọng Thế

**HỌC KỲ 2 NĂM HỌC 2024-2025**

# **MỤC LỤC**

[**MỤC LỤC** 2](#_Toc198399279)

[**DANH MỤC HÌNH ẢNH** 3](#_Toc198399280)

[CHƯƠNG 1. GIỚI THIỆU CHUNG VỀ BÀI THỰC HÀNH 4](#_Toc198399281)

[1.1 Mục đích 4](#_Toc198399282)

[1.2 Tìm hiểu lý thuyết 4](#_Toc198399283)

[1.2.1 Socket là gì? 4](#_Toc198399284)

[1.2.2. Tại sao cần sử dụng socket? 4](#_Toc198399285)

[1.2.3. Điều kiện để Socket hoạt động? 5](#_Toc198399286)

[1.2.4. Phân loại socket 6](#_Toc198399287)

[1.2.5. Lập trình socket với TCP/IP 7](#_Toc198399288)

[CHƯƠNG 2. NỘI DUNG THỰC HÀNH 9](#_Toc198399289)

[2.1 Chuẩn bị môi trường 9](#_Toc198399290)

[2.2 Các bước thực hiện 9](#_Toc198399291)

[2.2.1.Lập trình client và server với TCP socket 9](#_Toc198399292)

[2.2.2.Trao đổi thông điệp giữa client và server và đảm bảo tính toàn vẹn của thông điệp khi trao đổi 11](#_Toc198399293)

[2.3. Kết luận 17](#_Toc198399294)

[TÀI LIỆU THAM KHẢO 18](#_Toc198399295)

# **DANH MỤC HÌNH ẢNH**

[Hình 1: Socket là gì 4](#_Toc198398871)

[Hình 2: Socket hoat động 5](#_Toc198398872)

[Hình 3: Quá trình socket hoạt động 8](#_Toc198398873)

[Hình 4: Lập trình bên client 9](#_Toc198398874)

[Hình 5: Lập trình bên server 10](#_Toc198398875)

[Hình 6: Chạy bên server 10](#_Toc198398876)

[Hình 7: Client gửi và nhận được tin 10](#_Toc198398877)

[Hình 8: Server nhận được từ client và gửi lại 11](#_Toc198398878)

[Hình 9: Đọc gói tin trên ip server 11](#_Toc198398879)

[Hình 10: Đọc gói tin trên ip client 11](#_Toc198398880)

[Hình 11: Cải tiến code server 12](#_Toc198398881)

[Hình 12: Cải tiến code server 12](#_Toc198398882)

[Hình 13: Cải tiến code server 12](#_Toc198398883)

[Hình 14: Cải tiến code Client 13](#_Toc198398884)

[Hình 15: Cải tiến code Client 13](#_Toc198398885)

[Hình 16: Client gửi thông điệp tới server và được băm kiểm tra tính toàn vẹn 14](#_Toc198398886)

[Hình 17: Server nhận được thông tin và kiểm tra đúng đắn tính toàn vẹn 14](#_Toc198398887)

[Hình 18: Gói tin của client và hàm băm 15](#_Toc198398888)

[Hình 19: Gói tin bắt được từ server và hàm băm của gói tin 15](#_Toc198398889)

[Hình 20: Giá trị key của client bị thay đổi, không đảm bảo toàn vẹn 16](#_Toc198398890)

[Hình 21: Server nhận được gói tin bị thay đổi không đảm bảo tính toàn vẹn 16](#_Toc198398891)

[Hình 22: Hàm băm tại gói tin bị thay đổi 17](#_Toc198398892)

[Hình 24: Gói tin được bắt của server 17](#_Toc198398893)

1. GIỚI THIỆU CHUNG VỀ BÀI THỰC HÀNH
   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 cài đặt giao thức đơn giản để trao đổi thông tin an toàn.

* 1. Tìm hiểu lý thuyết

**1.2.1 Socket là gì?**



Hình 1: Socket là gì

Socket là điểm cuối (endpoint) trong một kết nối truyền thông hai chiều giữa hai tiến trình (process) trên mạng. Nó đóng vai trò là giao diện lập trình ứng dụng (API – Application Programming Interface) để các chương trình có thể gửi và nhận dữ liệu thông qua mạng máy tính, thường sử dụng giao thức TCP/IP.

Mỗi socket được ràng buộc với một địa chỉ IP và một cổng (port) cụ thể, giúp định danh ứng dụng mà dữ liệu cần được truyền tới. Trong mô hình Client-Server, một socket thường đại diện cho phía máy khách hoặc máy chủ, giúp thiết lập và duy trì kết nối truyền thông.

Khi một ứng dụng muốn giao tiếp qua mạng, nó sẽ tạo một socket, liên kết nó với địa chỉ IP và cổng cụ thể, sau đó sử dụng socket để gửi hoặc nhận dữ liệu qua kết nối mạng. Với TCP, socket cung cấp kết nối **đáng tin cậy** và **có thứ tự**, phù hợp với các ứng dụng như trình duyệt web, phần mềm chat, truyền tệp,...

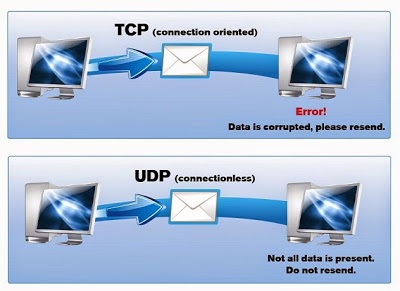
**1.2.2. Tại sao cần sử dụng socket?**

Người dùng cần đến socket vì nó là một cách tiêu biểu để thiết lập và quản lý kết nối mạng giữa các thiết bị và ứng dụng trên Internet. Dưới đây là một số lý do chính:

* **Giao tiếp mạng**: Cho phép các ứng dụng giao tiếp với nhau thông qua mạng cục bộ (LAN) hoặc Internet. Điều này là cần thiết trong nhiều ứng dụng như trò chơi trực tuyến, chat, truyền tệp, và nhiều ứng dụng mạng khác. Ví dụ: trò chuyện trực tuyến, game online, truyền file,...
* **Tính linh hoạt**: Socket hỗ trợ cả TCP (đảm bảo độ tin cậy) và UDP (nhanh, không cần xác nhận), tùy theo yêu cầu của ứng dụng, cung cấp sự linh hoạt trong cách ứng dụng giao tiếp với nhau.
* **Phân phối dữ liệu**: Hỗ trợ gửi dữ liệu từ một nguồn tới nhiều đích hoặc từ nhiều nguồn tới một đích, tạo nền tảng cho các hệ thống phân tán hoặc ứng dụng nhiều người dùng như streaming video, trò chơi trực tuyến, và các ứng dụng đa người dùng
* **Tương thích đa nền tảng**: Hầu hết hệ điều hành hiện đại đều hỗ trợ socket, giúp các ứng dụng có thể hoạt động trên nhiều nền tảng khác nhau như Windows, Linux, macOS,...
* **Kiểm soát độ trễ**: Lập trình viên có thể chủ động tối ưu tốc độ truyền nhận dữ liệu, đặc biệt quan trọng với các ứng dụng thời gian thực như game, VoIP, livestream,...

Tóm lại, socket là một công cụ quan trọng trong việc phát triển các ứng dụng mạng, cho phép giao tiếp hiệu quả và đa dạng giữa các thiết bị và ứng dụng trên Internet.

**1.2.3. Điều kiện để Socket hoạt động?**



Hình 2: Socket hoat động

Như đã đề cập trước đó, 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. Giao diện lập trình ứng dụng mạng này chỉ có thể hoạt động khi đã có thông tin về thông số IP và số hiệu cổng của 2 ứng dụng cần trao đổi dữ liệu cho nhau.

Để socket có thể hoạt động và thực hiện truyền thông hiệu quả giữa hai tiến trình, cần đảm bảo các điều kiện sau:

* Hai tiến trình cần có thông tin địa chỉ IP và cổng (port) để xác định nơi gửi và nhận dữ liệu.
* Hai tiến trình có thể chạy trên **cùng một máy** hoặc **khác máy**:
  + Nếu chạy **trên cùng một máy**, mỗi socket phải sử dụng một cổng riêng biệt để tránh xung đột.
  + Nếu chạy **trên các máy khác nhau**, cả hai phải có khả năng truy cập mạng và giao tiếp qua giao thức TCP/IP hoặc UDP.
* Phía **server** phải luôn hoạt động để lắng nghe và chấp nhận kết nối từ client.

**1.2.4. Phân loại socket**

Socket có thể được phân loại chủ yếu dựa trên hai tiêu chí: giao thức và cách sử dụng. Dưới đây là một số phân loại phổ biến của socket:

1. **Theo giao thức sử dụng:**
   * + **Socket TCP**: Sử dụng giao thức TCP (Transmission Control Protocol). Đây là loại socket phổ biến nhất, cung cấp kết nối tin cậy, tuần tự và có kiểm soát lỗi, bảo đảm dữ liệu đến được đích một cách chính xác. Phù hợp với truyền file, HTTP, email,...
     + **Socket UDP**: Sử dụng giao thức UDP (User Datagram Protocol). Không đảm bảo tính tin cậy nhưng có tốc độ truyền nhanh mà không cần quá nhiều kiểm soát., thích hợp cho các ứng dụng như video streaming, game thời gian thực,...
2. **Theo vai trò trong mô hình client-server:**
   * + **Socket server**: Được sử dụng ở phía máy chủ để lắng nghe và chấp nhận kết nối từ các client. Sử dụng các hàm *bind(), listen(), accept().*
     + **Socket client**: Được sử dụng ở phía máy khách để thiết lập kết nối tới server và thực hiện trao đổi dữ liệu. Sử dụng hàm *connect().*

Mỗi loại socket có mục đích và ứng dụng khác nhau trong lập trình mạng. Sử dụng đúng loại socket sẽ giúp đảm bảo tính ổn định và hiệu suất của ứng dụng mạng.

**1.2.5. Lập trình socket với TCP/IP**

Các tiến trình mà muốn truyền thông với nhau thì sẽ gửi thông điệp thông qua các socket. Socket là cánh cửa của tiến trình ứng dụng và giao thức tầng transport (ở đây là TCP). Để có thể tương tác với nhau và máy chủ có thể nhận liên lạc từ máy khách thì máy chủ phải luôn sẵn sàng.

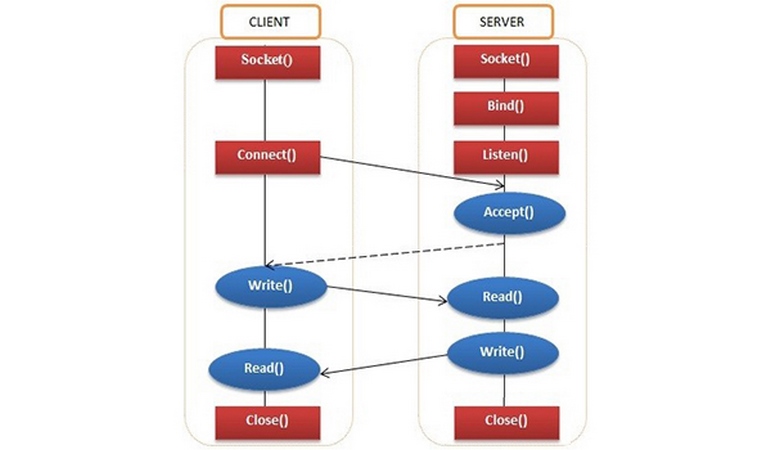
Điều này có 2 nghĩa, thứ nhất, giống như trường hợp của UDP, một tiến trình của máy chủ phải được chạy trước khi máy khách khởi tạo liên lạc đến. Thứ hai, chương trình chủ phải tạo ra 1 socket để sẵn sàng chấp nhận kết nối từ tiến trình khách.

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.

Trong lập trình mạng theo mô hình Client-Server sử dụng TCP/IP, quá trình thiết lập kết nối giữa hai tiến trình trải qua các bước sau:



Hình 3: Quá trình socket hoạt động

1. **Khởi tạo socket**: Phía server tạo một socket thông qua lời gọi socket().
2. **Liên kết socket với địa chỉ IP và cổng**: Server sử dụng *bind()* để liên kết socket với một địa chỉ IP và cổng cụ thể.
3. **Bắt đầu lắng nghe kết nối**: Server sử dụng *listen()* để chờ đợi các yêu cầu kết nối từ client.
4. **Kết nối từ client**: Client tạo socket và gọi *connect()* để kết nối đến server thông qua IP và cổng đã biết.
5. **Chấp nhận kết nối**: Server sử dụng *accept()* để chấp nhận kết nối từ client. Sau khi chấp nhận, một socket mới được tạo để phục vụ riêng cho client đó.
6. **Trao đổi dữ liệu**: Sau khi kết nối được thiết lập thành công, hai bên có thể sử dụng *send()* và *recv()* (hoặc *read()/write()*) để gửi và nhận dữ liệu.
7. **Đóng kết nối**: Sau khi truyền dữ liệu xong, cả hai bên gọi *close()* để giải phóng tài nguyên và kết thúc kết nối.

Đặc biệt, kết nối TCP được thiết lập thông qua quá trình **bắt tay 3 bước (three-way handshake)** để đảm bảo sự tin cậy trước khi truyền dữ liệu.

1. NỘI DUNG THỰC HÀNH
   1. Chuẩn bị môi trường

(Mô tả về các công việc chuẩn bị công cụ, môi trường cho bài thực hành)

* Môi trường Python hoặc Java để chạy được ứng dụng client/server đã lập trình
* Phần mềm Wireshark
  1. Các bước thực hiện

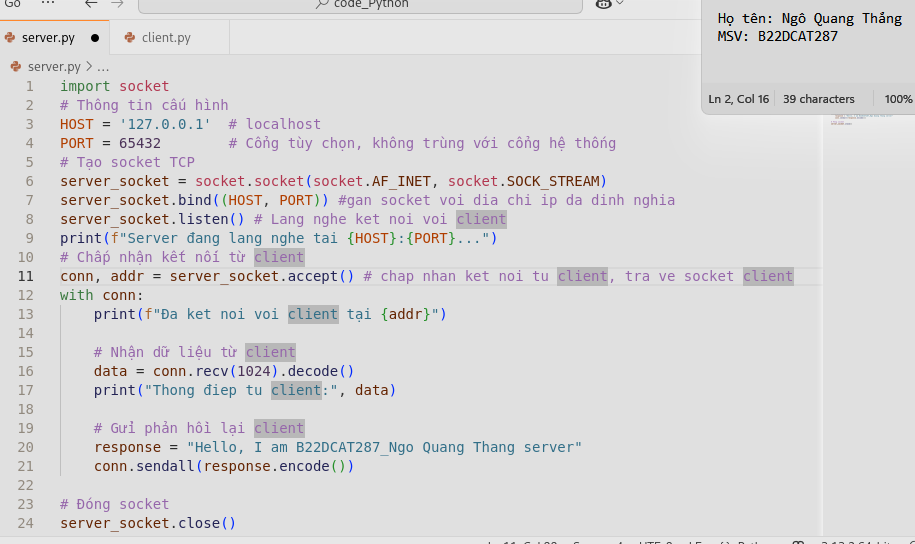
### **2.2.1.Lập trình client và server với TCP socket**

o Lập trình client



Hình 4: Lập trình bên client

o Lập trình server



Hình 5: Lập trình bên server

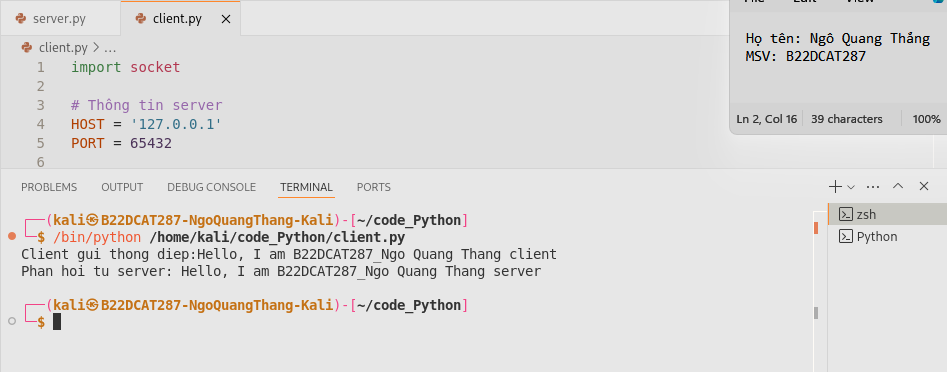
o Chạy server



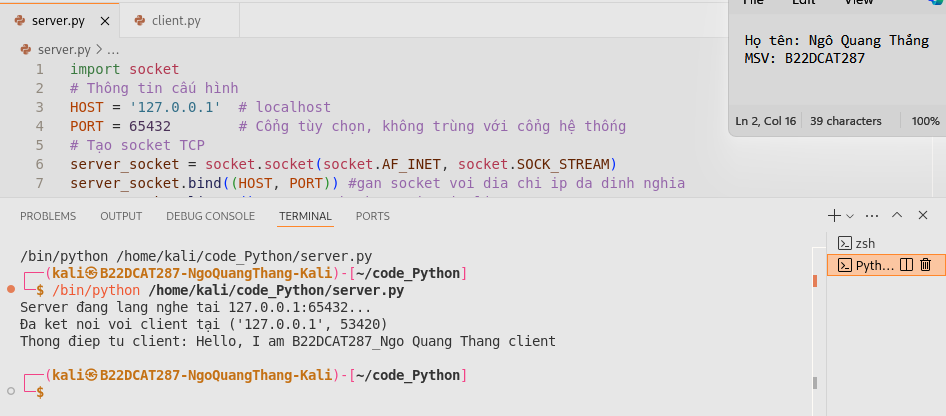
Hình 6: Chạy bên server

o Chạy Client, Client gửi thông điệp cá nhận hóa cho server: “Hello, I am client.”

o Server nhận được hiển thị thông điệp nhận được và gửi lại client thông điệp: server gửi lại “Hello, I am server”

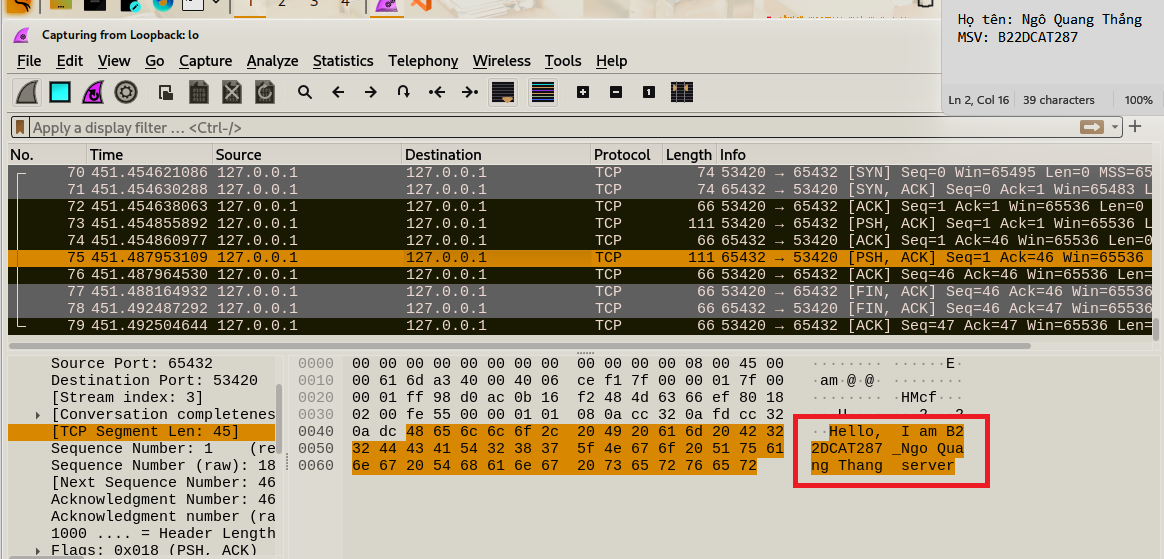


Hình 7: Client gửi và nhận được tin

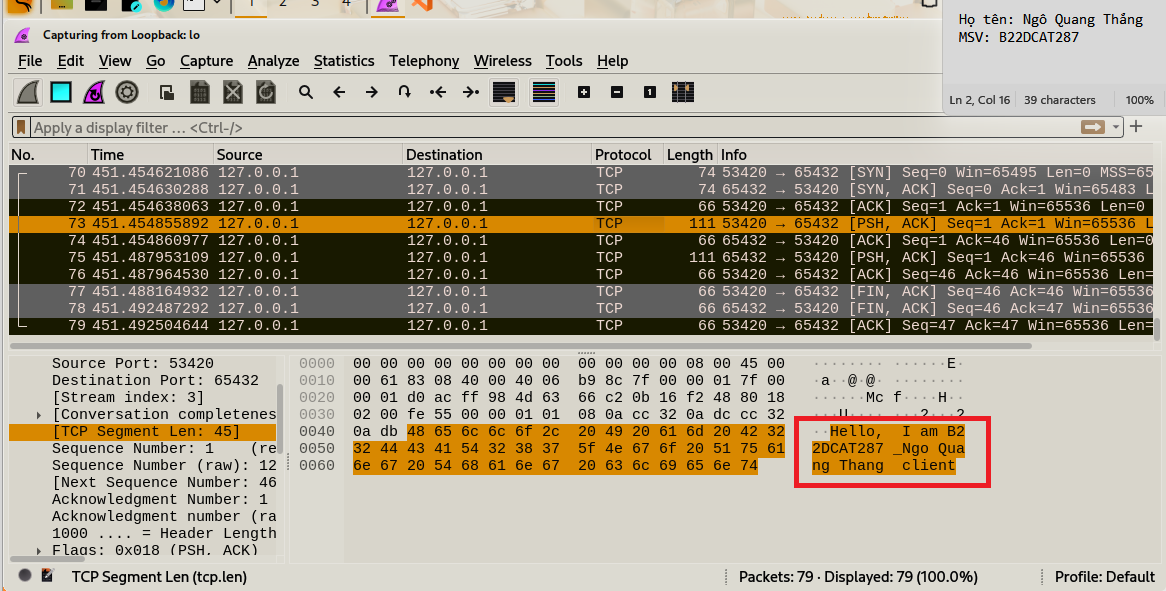


Hình 8: Server nhận được từ client và gửi lại

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



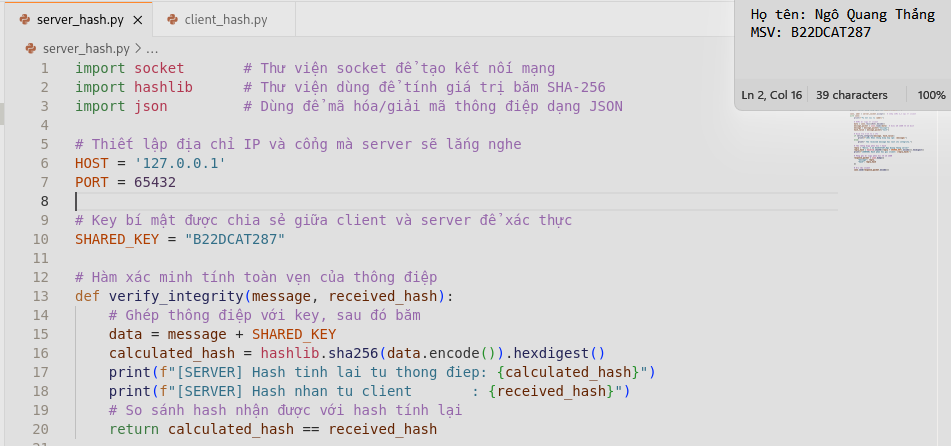
Hình 9: Đọc gói tin trên ip server



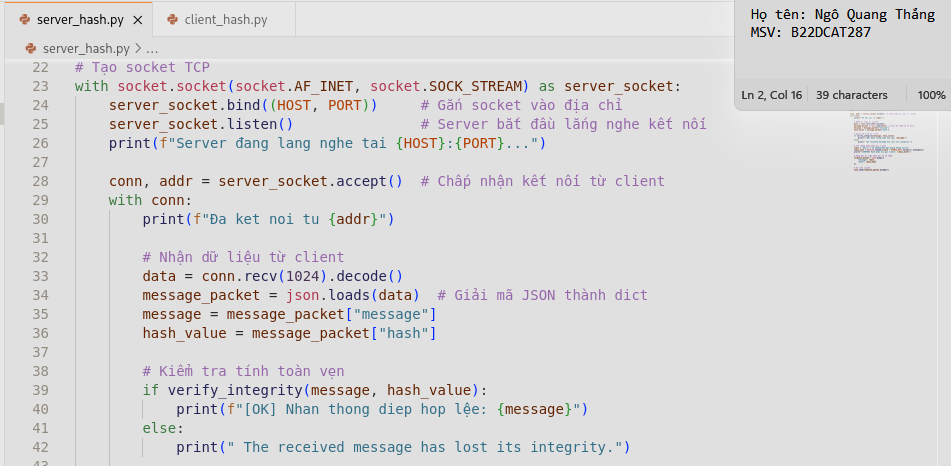
Hình 10: Đọc gói tin trên ip client

### **2.2.2.Trao đổi thông điệp giữa client và server và đảm bảo tính toàn vẹn của thông điệp khi trao đổi**

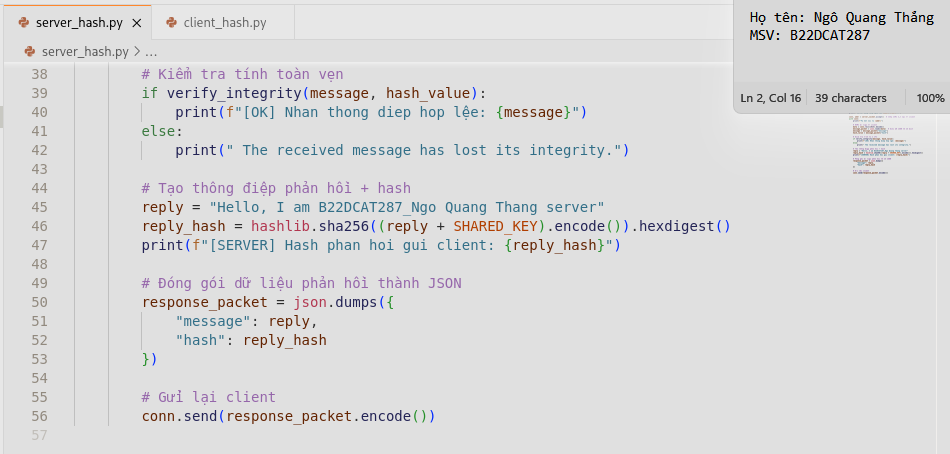
* Cải tiến code Server



Hình 11: Cải tiến code server

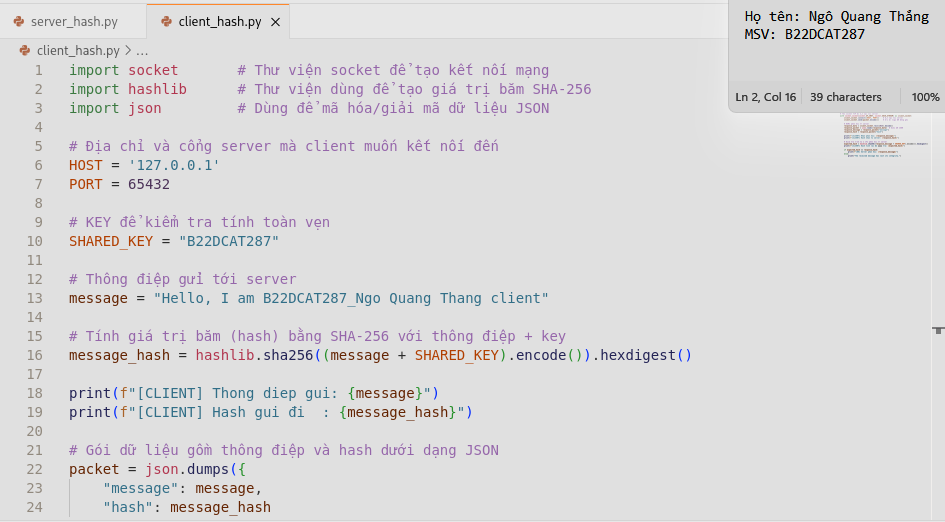


Hình 12: Cải tiến code server

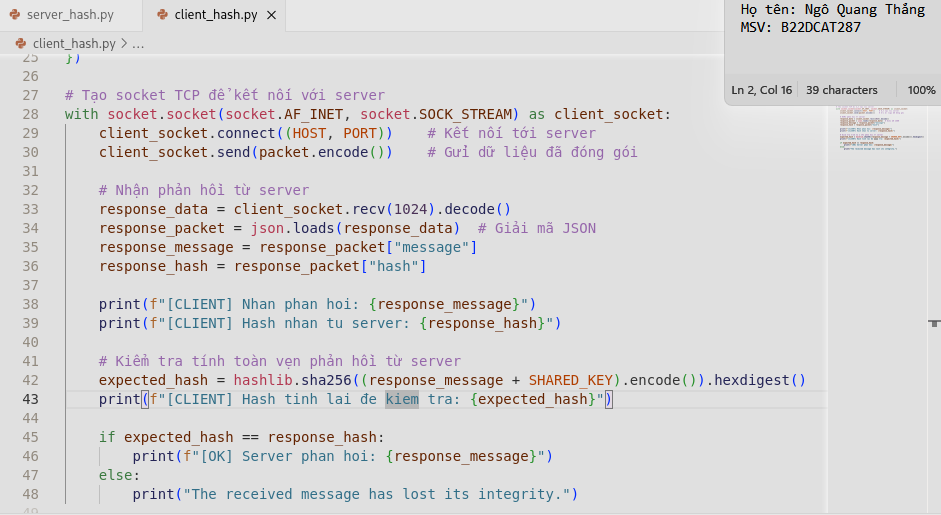


Hình 13: Cải tiến code server

* Cải tiến codeClient

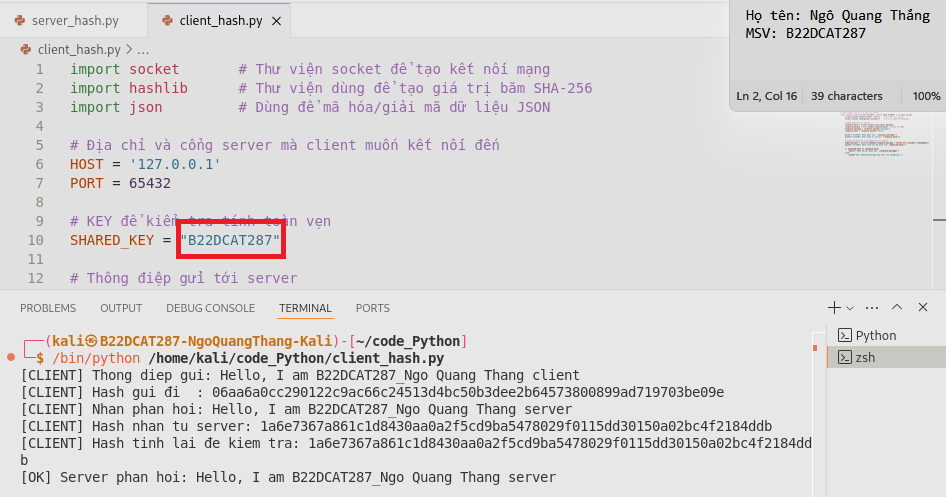


Hình 14: Cải tiến code Client

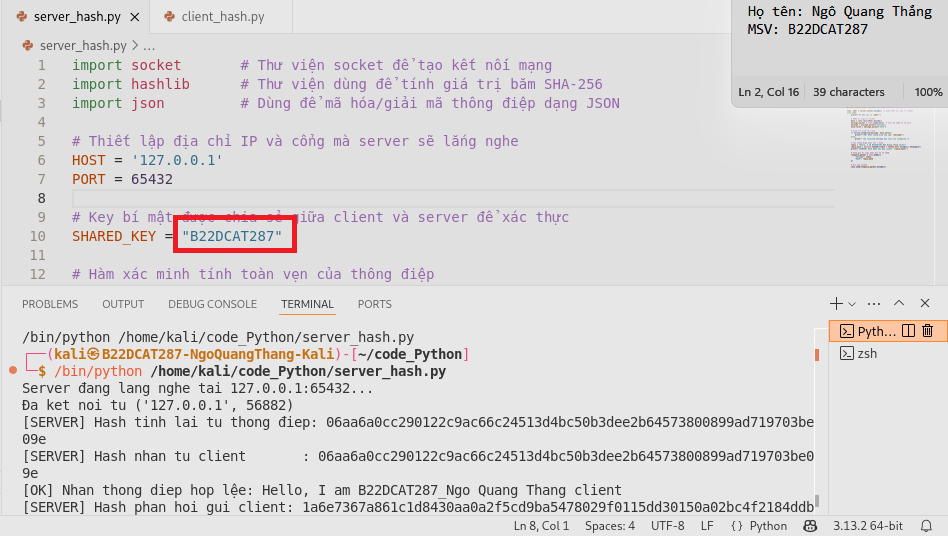


Hình 15: Cải tiến code Client

* 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.

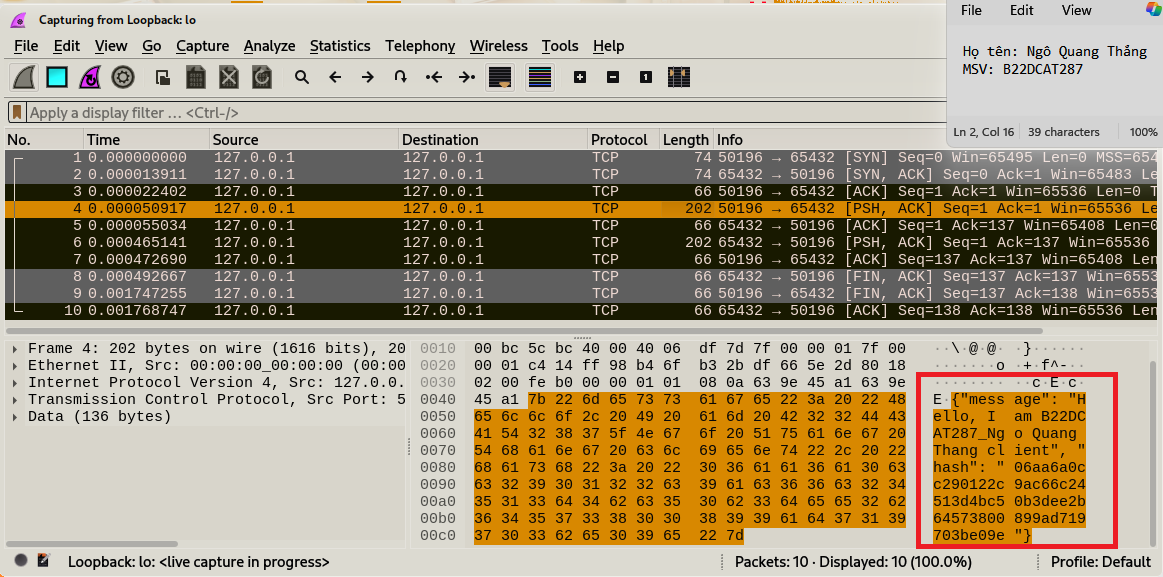


Hình 16: Client gửi thông điệp tới server và được băm kiểm tra tính toàn vẹn

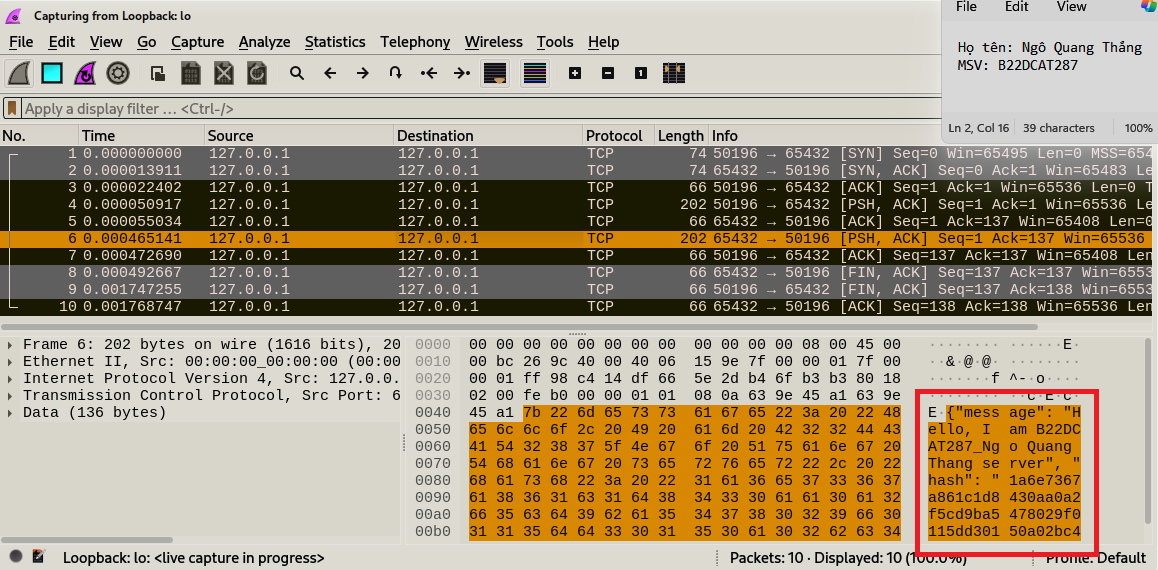


Hình 17: Server nhận được thông tin và kiểm tra đúng đắn tính toàn vẹn

* Bắt được các bản tin trao đổi giữa client và server trong Wireshark

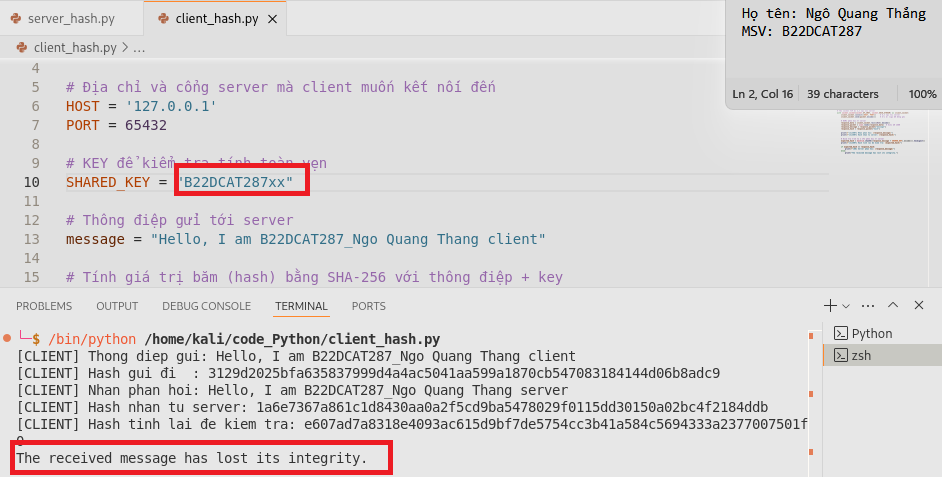


Hình 18: Gói tin của client và hàm băm

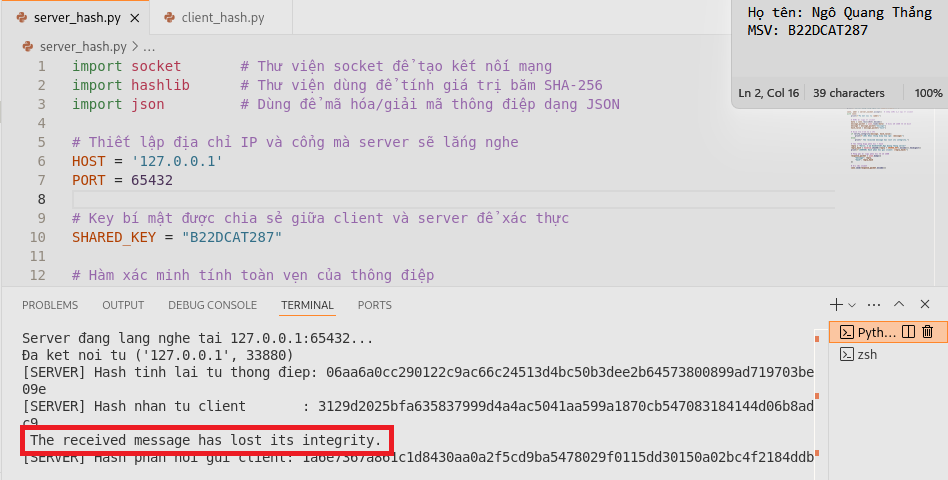


Hình 19: Gói tin bắt được từ server và hàm băm của gói tin

* 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.”

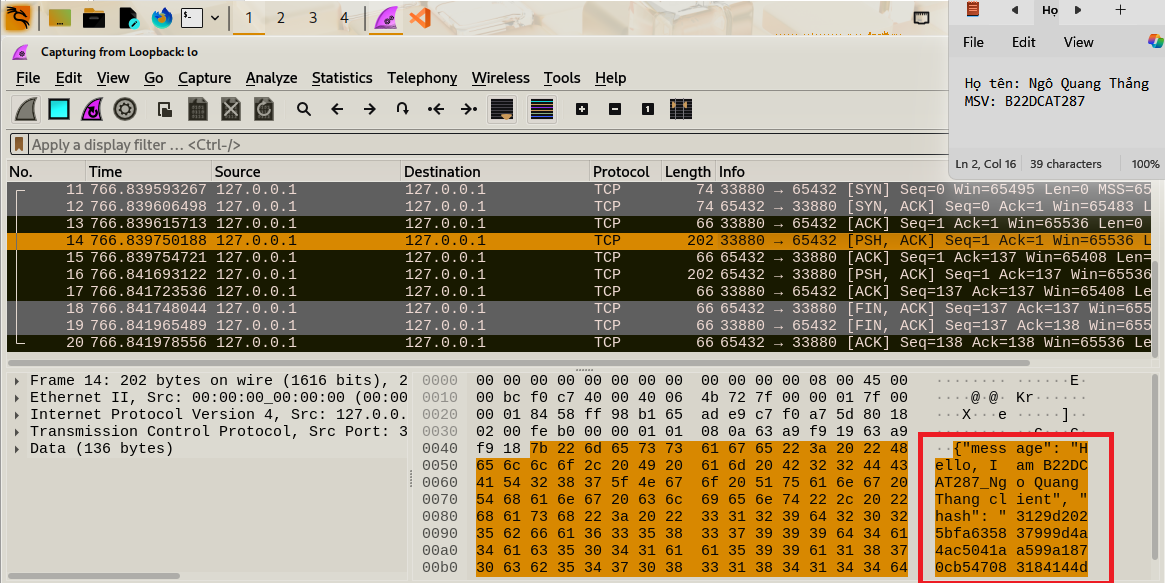


Hình 20: Giá trị key của client bị thay đổi, không đảm bảo toàn vẹn

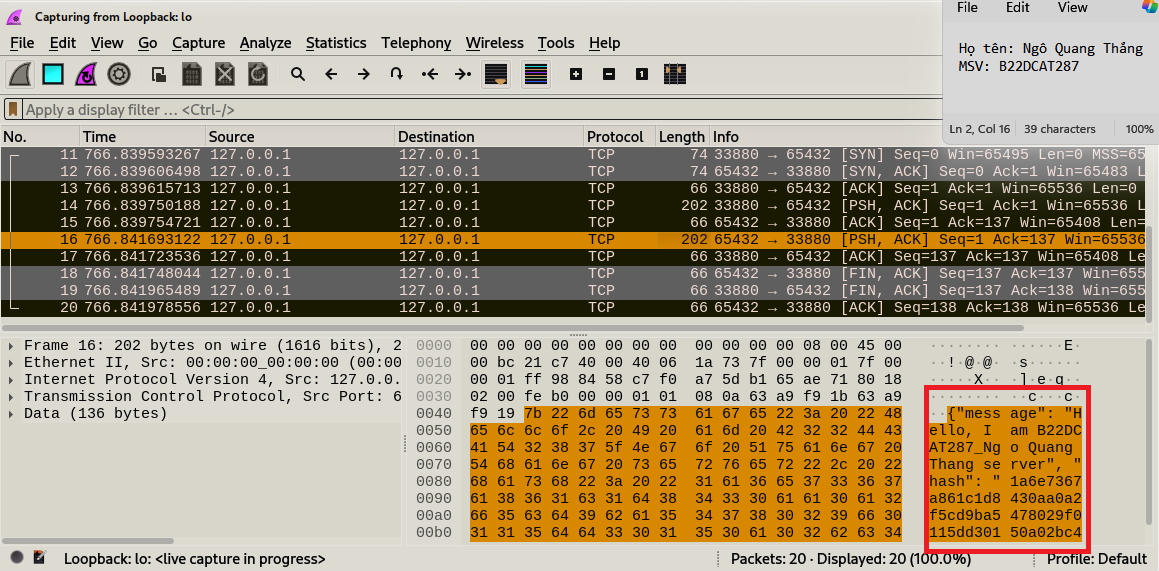


Hình 21: Server nhận được gói tin bị thay đổi không đảm bảo tính toàn vẹn

* Bắt được các bản tin trao đổi giữa client và server trong Wireshark



Hình 22: Hàm băm tại gói tin bị thay đổi



Hình 24: Gói tin được bắt của server

## **2.3. Kết luận**

- Hiểu được các khái niệm về socket, TCP

- Lập trình được Client/Server và hoạt động thành công

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. Nguồn tham khảo code socket : [Lập Trình Socket Với TCP/IP Trong Python | CodeLearn](https://codelearn.io/sharing/lap-trinh-socket-voi-tcpip-trong-python)
3. Nguồn code : <https://github.com/ThangNQ-Closer/TTCS_4.1>