

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

**KHOA CÔNG NGHỆ THÔNG TIN 1**

---



**BÀI THỰC HÀNH 15**  
**THỰC TẬP CƠ SỞ**

**Họ và tên : Đinh Quang Hiếu**

**Mã sinh viên: B19DCAT065**

**Giảng viên giảng dạy: Hoàng Xuân Dật**

**HÀ NỘI, THÁNG 5/2022**

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

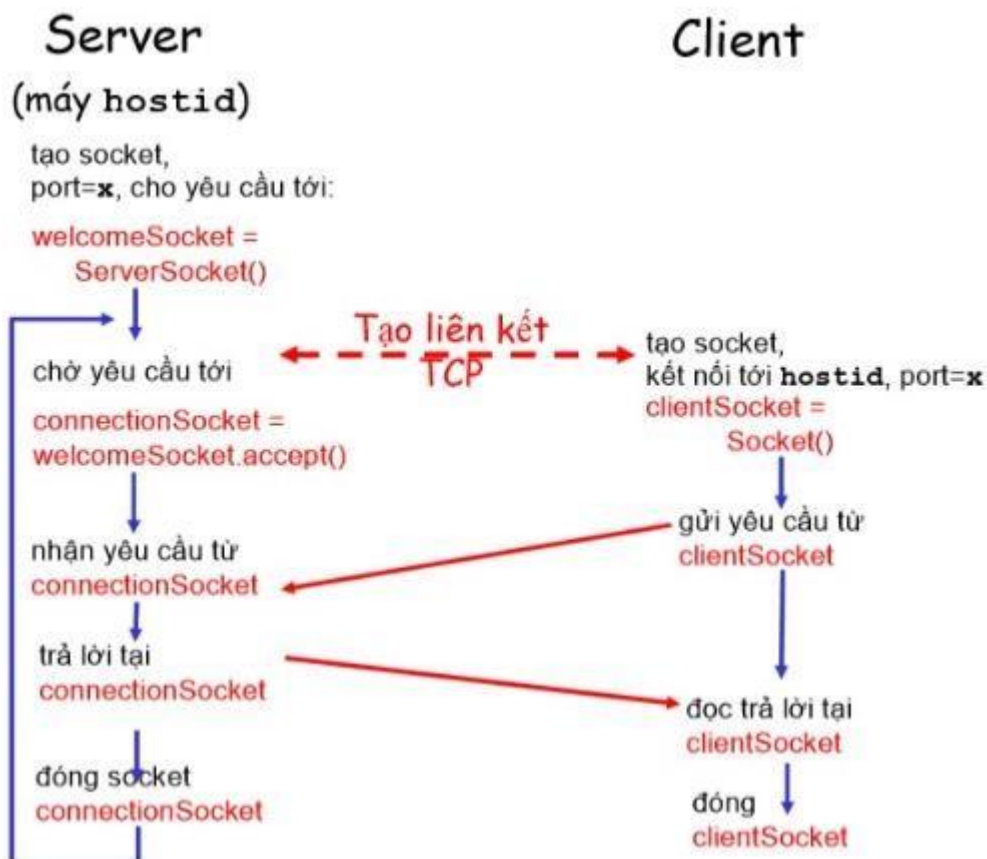
## I. Giới thiệu chung

### 1. Mục đích

- Tìm hiểu về cơ chế client/server, lập trình client/server dựa trên socket, cài đặt giao thức đơn giản để trao đổi thông tin an toàn.

### 2. Yêu cầu

- Tìm hiểu lý thuyết
  - Socket
    - ✦ Là điểm cuối (end-point) trong liên kết truyền thông hai chiều (two-way communication) biểu diễn kết nối giữa Client – Server
    - ✦ Phân loại Socket:
      - Stream Socket: Dựa trên giao thức TCP, thiết lập giao tiếp 2 chiều; đảm bảo dữ liệu được truyền đến nơi nhận một cách đáng tin cậy, đúng tuần tự.
      - Datagram Socket: Dựa trên giao thức UDP, không yêu cầu có sự thiết lập kết nối giữa 2 process; ưu điểm là tốc độ giao thức nhanh.
    - Lập trình socket với TCP



- Chuẩn bị
  - Phần mềm Wireshark
  - Môi trường Python

### 3. Các bước thực hiện

#### 3.1 Lập trình client và server với TCP socket

- a) Các bước thực hiện
- Lập trình client

```
copy /> send_msg
import socket
import threading
import time

# IP Address and Port
HOST = '127.0.0.1'
PORT = 8080

# Create socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_address = (HOST, PORT)
print('connecting to %s port ' %str(server_address))
s.connect(server_address)

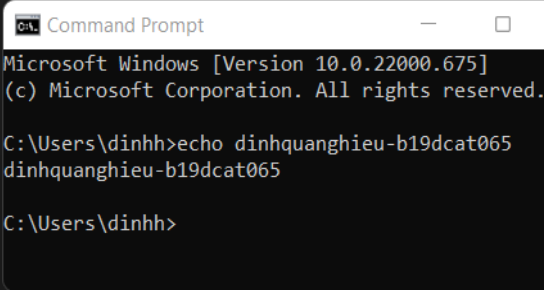
# Function sent message to server
def sent_msg():
    while True:
        msg = input('\nSent to server: ')
        if len(msg) == 0:
            continue

        time.sleep(0.2)

        # Send data with TCP
        s.sendall(bytes(msg, "utf8"))

# Function receive data to server
def rev_msg():
    while True:
        # Receive data
        data = s.recv(1024)
        msg_rev = data.decode("utf8")
        print('\nReceive from server: ', msg_rev)

        time.sleep(0.2)
```



```
client.py M X
client.py > sent_msg
37
38 try:
39     # Thread sent message
40     th1 = threading.Thread(target=sent_msg, name='t1')
41
42     # Thread receive data
43     th2 = threading.Thread(target=rev_msg, name='t2')
44
45     # Set two thread is Deamon
46     th1.daemon = True
47     th2.daemon = True
48
49     # Start thread
50     th1.start()
51     th2.start()
52
53     time.sleep(1)
54
55     # End thread
56     th1.join()
57     th2.join()
58
59 finally:
60     s.close()
```

```
Command Prompt
Microsoft Windows [Version 10.0.22000.675]
(c) Microsoft Corporation. All rights reserved.

C:\Users\dinh>echo dinhquanghieu-b19dcat065
dinhquanghieu-b19dcat065

C:\Users\dinh>
```

- Lập trình server

```
server.py M X
server.py > ...
1  import socket
2  import threading
3  import time
4
5  # IP Address and Port
6  HOST = '127.0.0.1'
7  PORT = 8080
8
9
10 # Create socket
11 # Listen max 2 socket
12 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
13 s.bind((HOST, PORT))
14 s.listen(2)
15
16 # Function sent message to client
17 def sent_msg(client):
18     while True:
19         msg = input('\nSent to client: ')
20         if len(msg) == 0:
21             continue
22
23         time.sleep(0.2)
24         # Send data with TCP
25         client.sendall(bytes(msg, "utf8"))
26
27
28 # Function receive data to client
29 def rev_msg(client):
30     while True:
31         # Receive data
32         data = client.recv(1024)
33         msg_rev= data.decode("utf8")
34         print("\nReceive from client: " + msg_rev)
35         time.sleep(0.2)
36
37
```

```
Command Prompt
Microsoft Windows [Version 10.0.22000.675]
(c) Microsoft Corporation. All rights reserved.

C:\Users\dinh>echo dinhquanghieu-b19dcat065
dinhquanghieu-b19dcat065

C:\Users\dinh>
```

```
server.py M X
server.py > ...
35 time.sleep(0.2)
36
37
38 while True:
39     # Accept client connect to server
40     # Create client socket
41     client, addr = s.accept()
42
43     try:
44         print('Connected by', addr)
45
46         # Thread sent message
47         th1 = threading.Thread(target=sent_msg, args=(client,), name='t1')
48
49         # Thread receive data
50         th2 = threading.Thread(target=rev_msg, args=(client,), name='t2')
51
52         # Set two thread is Deamon
53         th1.daemon = True
54         th2.daemon = True
55
56         # Start thread
57         th1.start()
58         th2.start()
59
60         time.sleep(1)
61
62         # End thread
63         th1.join()
64         th2.join()
65
66     finally:
67         client.close()
68
69 s.close()
```

```
Command Prompt
Microsoft Windows [Version 10.0.22000.675]
(c) Microsoft Corporation. All rights reserved.

C:\Users\dinh>echo dinhquanghieu-b19dcat065
dinhquanghieu-b19dcat065

C:\Users\dinh>
```

- Chạy server sau đó chạy client
- Client gửi thông điệp cá nhân hóa cho server: “Hello, I am B19DCAT205 client.”
- Server nhận được hiển thị thông điệp nhận được và gửi lại client thông điệp: server gửi lại “Hello, I am B19DCAT205 server”

```

PS D:\Socket-Programing> py .\server.py
Connected by ('127.0.0.1', 51763)

Sent to client: hello i am b19dcat065

Sent to client:
Receive from client: dinh quang hieu - b19dcat065

```

```

PS D:\Socket-Programing> py .\client.py
connecting to ('127.0.0.1', 8080) port

Sent to server:
Receive from server: hello i am b19dcat065
dinh quang hieu - b19dcat065

Sent to server:

```

```

C:\Users\dinh>echo dinhquanghieu-b19dcat065
dinhquanghieu-b19dcat065

C:\Users\dinh>

```

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

Time	Source	Destination	Protocol	Length	Info
10.1.904114	127.0.0.1	127.0.0.1	TCP	56	51790 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
11.1.904183	127.0.0.1	127.0.0.1	TCP	56	8080 → 51790 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
12.1.904211	127.0.0.1	127.0.0.1	TCP	44	51790 → 8080 [ACK] Seq=1 Ack=1 Win=2161152 Len=0
14.8.779351	127.0.0.1	127.0.0.1	TCP	72	8080 → 51790 [PSH, ACK] Seq=1 Ack=1 Win=2161152 Len=28 [TCP segment of a reassembled PDU]
15.8.779394	127.0.0.1	127.0.0.1	TCP	44	51790 → 8080 [ACK] Seq=1 Ack=29 Win=2161152 Len=0

Wireshark · Packet 14 · Adapter for loopback traffic capture

```

> Frame 14: 72 bytes on wire (576 bits), 72 bytes captured (576 bits) on interface \Device\NPF_{...}, id 0
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 8080, Dst Port: 51790, Seq: 1, Ack: 1, Len: 28

```

```

C:\Users\dinh>echo dinhquanghieu-b19dcat065
dinhquanghieu-b19dcat065

C:\Users\dinh>

```

```

0000 02 00 00 00 45 00 00 44 fd 7c 40 00 80 06 00 00 .....E..D..|@....
0010 7f 00 00 01 7f 00 00 01 1f 90 ca 4e 36 f0 7c 89 .....N6..|..
0020 12 ca 57 f1 50 18 20 fa dc 29 00 00 64 69 6e 68 ..W.P..)..dinh
0030 20 71 75 61 6e 67 20 68 69 65 75 20 2d 20 62 31 quang hieu - b1
0040 39 64 63 61 74 30 36 35 9dcat065

```

## b) Kết quả cần đạt được

- Chạy thành công client và server theo mục tiêu ban đầu
- Bắt được các bản tin trao đổi giữa client và server trong Wireshark

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

- a) Các bước thực hiện ○ 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.

Tạo hash module trong Python:

```
hash.py X
hash.py > ...
1 import hmac
2 import hashlib
3 import binascii
4
5 def get_hash_code(msg, key):
6     # Convert hex to bin
7     # Encode string with utf8
8     # Hash msg with SHA256, convert to hex
9     key = binascii.unhexlify(key)
10    msg = msg.encode()
11    return hmac.new(key, msg, hashlib.sha256).hexdigest().upper()
12
13 def check_integrity_msg(msg, key, code_rev):
14     # Hash msg with key and compare
15     if code_rev == get_hash_code(msg, key):
16         return True
17     else:
18         return False
19
```

```
Command Prompt
Microsoft Windows [Version 10.0.22000.675]
(c) Microsoft Corporation. All rights reserved.

C:\Users\dinh>echo dinhquanghieu-b19dcat065
dinhquanghieu-b19dcat065

C:\Users\dinh>
```

```
# Function sent message to client
def sent_msg(client):
    while True:
        msg = input('\nSent to client: ')
        if len(msg) == 0:
            continue

        time.sleep(0.2)

        # Hash msg and send msg + hash code
        hash_code = hash.get_hash_code(msg, SECRET_KEY)
        data_sent = msg + '|' + hash_code

        # Send data with TCP
        client.sendall(bytes(data_sent, "utf8"))
```

```
Command Prompt
Microsoft Windows [Version 10.0.22000.675]
(c) Microsoft Corporation. All rights reserved.

C:\Users\dinh>echo dinhquanghieu-b19dcat065
dinhquanghieu-b19dcat065

C:\Users\dinh>
```



```
# Function receive data to client
def rev_msg(client):
    while True:
        # Receive data
        data = client.recv(1024)
        msg_rev, hash_code_rev = data.decode("utf8").split('|')

        # Check integrity of message
        if hash.check_integrity_msg(msg_rev, SECRET_KEY, hash_code_rev):
            print("\nReceive from client: " + msg_rev)
        else:
            print("\nThe received message has lost its integrity")

        time.sleep(0.2)
```

```
Command Prompt
Microsoft Windows [Version 10.0.22000.675]
(c) Microsoft Corporation. All rights reserved.

C:\Users\dinh>echo dinhquanghieu-b19dcat065
dinhquanghieu-b19dcat065

C:\Users\dinh>
```

- 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.”

The screenshot shows a code editor with two files: `serverhash.py` and `clienthash.py`. The `serverhash.py` file contains the following code:

```
1 import socket
2 import sys
3 import threading
4 import time
5 import hash
6
7 # IP Address and Port
8 HOST = '127.0.0.1'
9 PORT = 8080
10
11 # Key to ensure integrity msg
12 SECRET_KEY = "5D2E44719232EA78CD2B50"
```

The `clienthash.py` file contains the following code:

```
1 import socket
2 import threading
3 import time
4 import hash
5
6 # IP Address and Port
7 HOST = '127.0.0.1'
8 PORT = 8080
9
10 # Key to ensure integrity msg
11 SECRET_KEY = "5D2E44719232EA78CD2B32"
```

A Command Prompt window is open in the foreground, showing the following commands and output:

```
Microsoft Windows [Version 10.0.22000.675]
(c) Microsoft Corporation. All rights reserved.

C:\Users\dinh>echo dinhquanghieu-b19dcat065
dinhquanghieu-b19dcat065

C:\Users\dinh>
```

The screenshot shows a terminal window with the following output:

```
PS D:\Socket-Programing> py .\serverhash.py
Connected by ('127.0.0.1', 64320)

Sent to client: dinh quang hieu - b19dcat065
Sent to client: []
```

The `clienthash.py` file is also shown, with the following output:

```
PS D:\Socket-Programing> py .\clienthash.py
connecting to ('127.0.0.1', 8080) port

Sent to server:
The received message has lost its integrity
[]
```

A Command Prompt window is open in the foreground, showing the following commands and output:

```
Microsoft Windows [Version 10.0.22000.675]
(c) Microsoft Corporation. All rights reserved.

C:\Users\dinh>echo dinhquanghieu-b19dcat065
dinhquanghieu-b19dcat065

C:\Users\dinh>
```

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

The image shows a Wireshark packet capture of a TCP SYN packet. The packet list on the left shows five packets, with packet 4 selected. The packet details pane shows the structure of the selected packet: Frame 4: 137 bytes on wire (1096 bits), 137 bytes captured (1096 bits) on interface \Device\NPF\_{...}, id 0. The packet is a Null/Loopback packet. The Internet Protocol Version 4 section shows Source: 127.0.0.1, Destination: 127.0.0.1. The Transmission Control Protocol section shows Source Port: 8080, Destination Port: 64340, Seq: 1, Ack: 1, Len: 93. The packet bytes pane shows the raw data of the packet, including the Ethernet II header, Internet Protocol Version 4 header, and the TCP header.

Command Prompt output:

```

C:\Users\dinh>echo dinhquanghieu-b19dcat065
dinhquanghieu-b19dcat065
C:\Users\dinh>

```

## b) Kết quả cần đạt được

- Chạy thành công client và server theo mục tiêu ban đầu, bắt được các bản tin trao đổi giữa client và server trong Wireshark