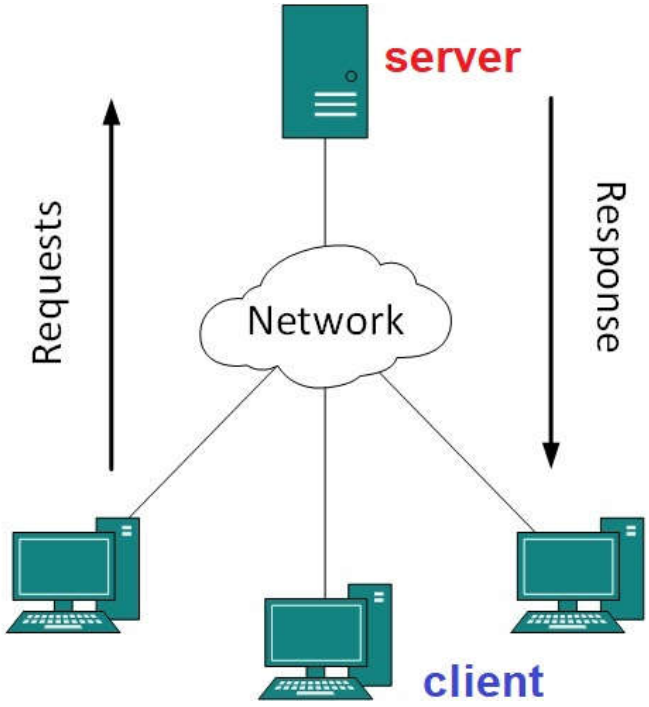
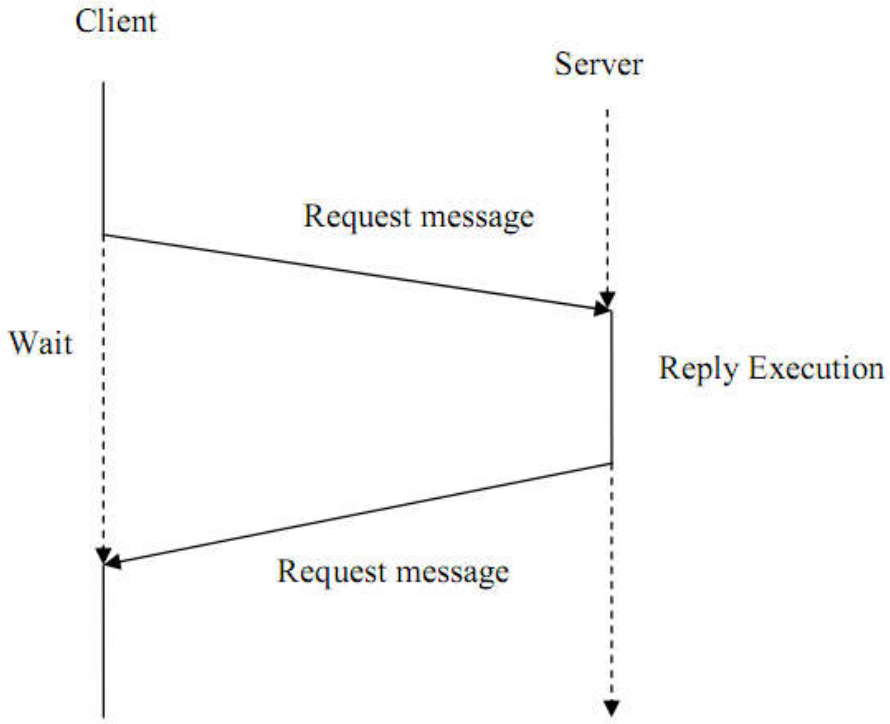
Python Sockets

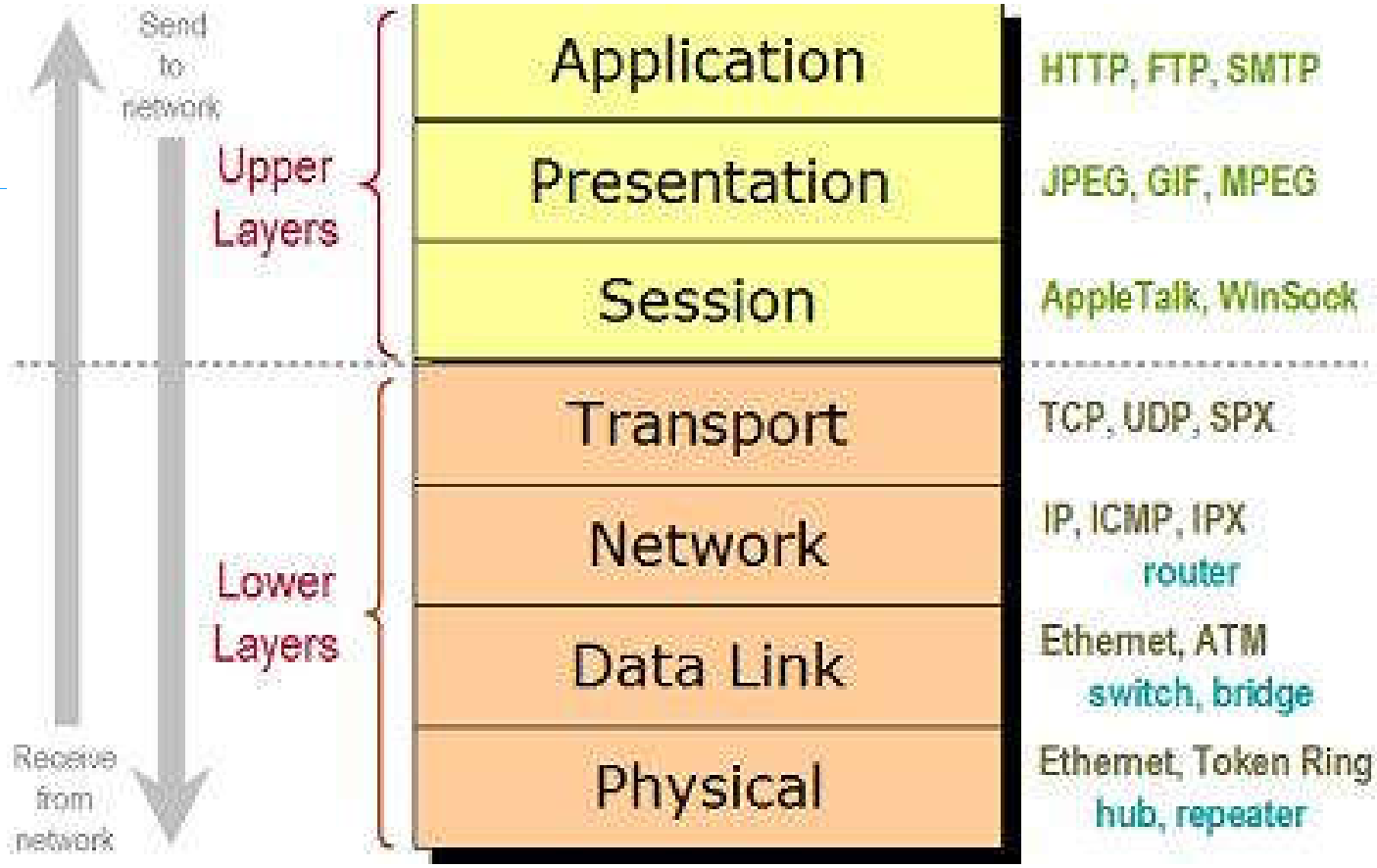
Nội dung

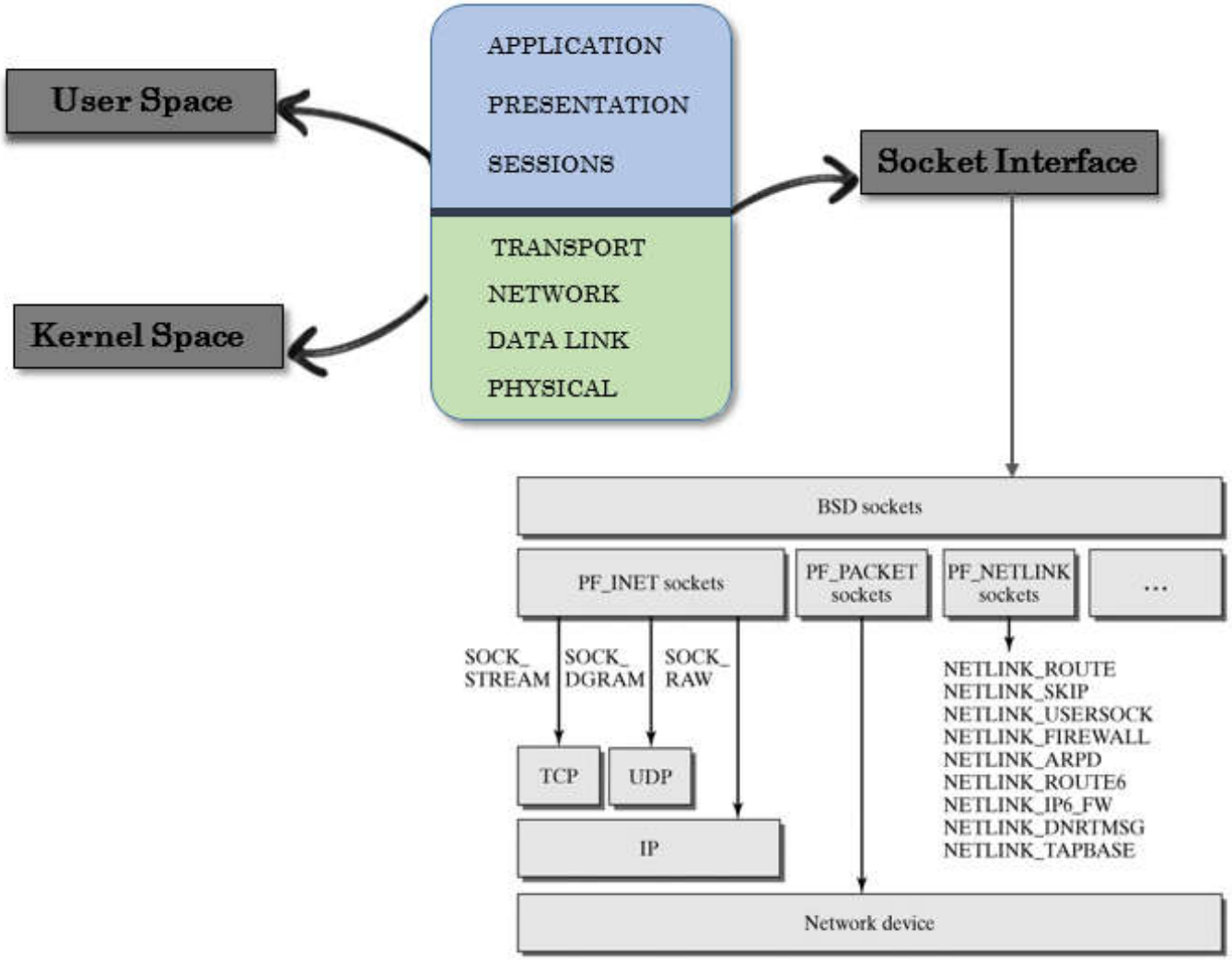
1. Mô hình Client - Server
2. Khái niệm sockets
3. Server Socket Methods
4. Client Socket Methods
5. Bài tập thực hành

Client – Server






Send to network
Receive from network
Upper Layers
Lower Layers
Application
Presentation
Session
Transport
Network
Data Link
Physical
HTTP, FTP, SMTP
JPEG, GIF, MPEG
AppleTalk, WinSock
TCP, UDP, SPX
IP, ICMP, IPX
router
Ethernet, ATM
switch, bridge
Ethernet, Token Ring
hub, repeater


User Space
Kernel Space
APPLICATION
PRESENTATION
SESSIONS
TRANSPORT
NETWORK
DATA LINK
PHYSICAL
Socket Interface
BSD sockets
PF_INET sockets
PF_PACKET sockets
PF_NETLINK sockets
...
SOCK_ STREAM
SOCK_ DGRAM
SOCK_ RAW
TCP
UDP
IP
NETLINK_ROUTE
NETLINK_SKIP
NETLINK_USERSOCK
NETLINK_FIREWALL
NETLINK_ARPD
NETLINK_ROUTE6
NETLINK_IP6_FW
NETLINK_DNRTMSG
NETLINK_TAPBASE
Network device

Python networking

- Python có 2 level để truy cập đến các dịch vụ mạng:
  - Ở low-level, có thể truy cập đến Socket của Hệ điều hành, nơi cho phép tác động Client - Server bằng cả 2 giao thức Connection-oriented (kết nối định tuyến) và Connectionless (kết nối bất định tuyến).
  - Ngoài ra, Python còn cung cấp 1 số thư viện để truy cập đến các giao thức High-level như HTTP, FTP, ….

Sockets

- Socket là **một cổng logic** mà một chương trình sử dụng để kết nối với một chương trình khác chạy trên cùng một máy tính hoặc trên một máy tính khác thông qua giao thức mạng.
- Chương trình mạng có thể sử dụng nhiều Socket cùng một lúc, nhờ đó nhiều chương trình có thể sử dụng Internet cùng một lúc.
- Có 2 loại socket thường thấy đó là dùng giao thức **TCP** hoặc **UDP**.

Stream Sockets

- Stream Socket – Dựa trên giao thức TCP (Tranmission Control Protocol) việc truyền dữ liệu chỉ thực hiện giữa 2 quá trình đã thiết lập kết nối. Giao thức này đảm bảo dữ liệu được truyền đến nơi nhận một cách đáng tin cậy, đúng thứ tự nhờ vào cơ chế quản lý luồng lưu thông trên mạng và cơ chế chống tắc nghẽn.

Datagram Socket

- Datagram Socket – Dựa trên giao thức UDP (User Datagram Protocol) việc truyền dữ liệu không yêu cầu có sự thiết lập kết nối giữa 2 quá trình.
- Ngược lại với giao thức TCP thì dữ liệu được truyền theo giao thức UDP không được tin cậy, có thế không đúng trình tự và lặp lại. Tuy nhiên vì nó không yêu cầu thiết lập kết nối không phải có những cơ chế phức tạp nên tốc độ nhanh…ứng dụng cho các ứng dụng truyền dữ liệu nhanh như chat, game…..

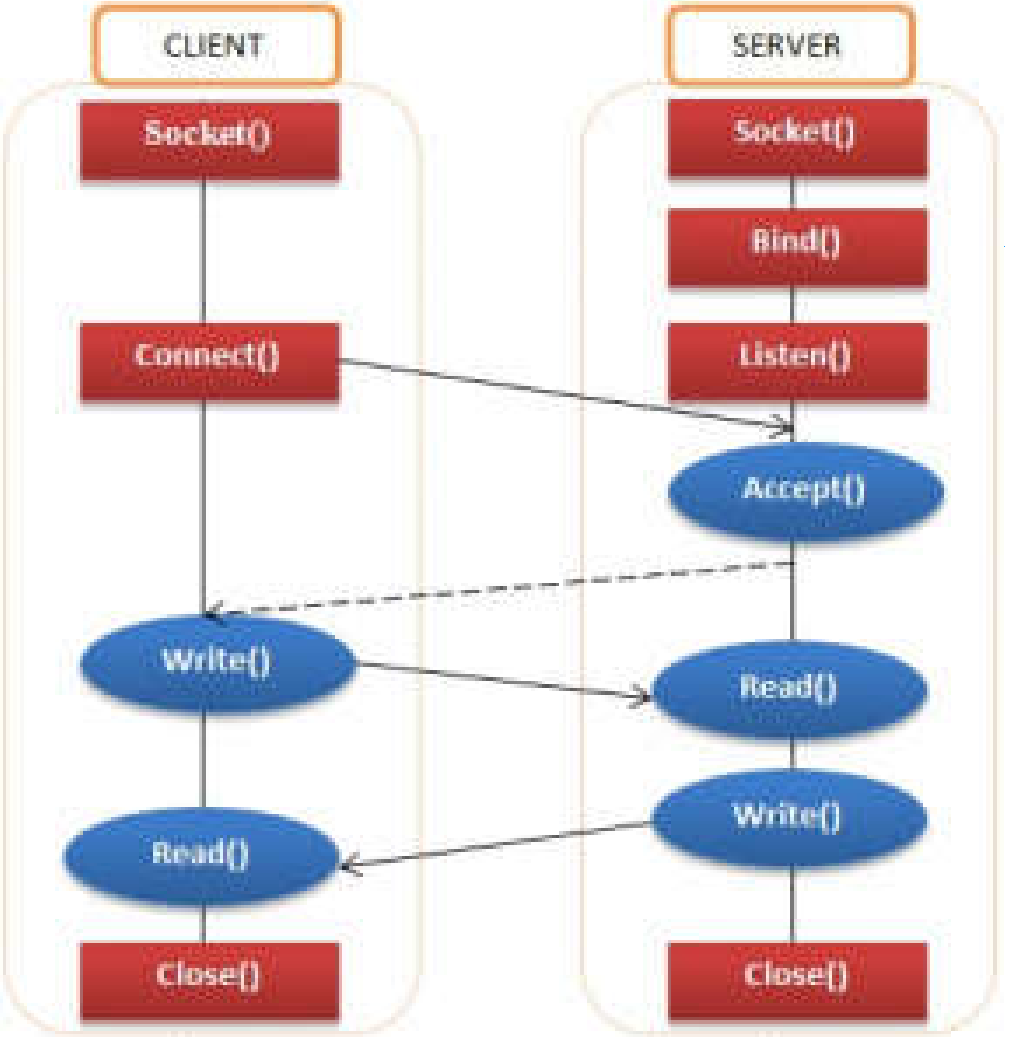
Port

- Port xách định duy nhất một tiến trình (process) trên một máy trong mạng. Hay nói cách khác là cách mà phân biệt giữa các ứng dụng chạy trên máy tính.
- Một TCP/IP Socket gồm một địa chỉ IP kết hợp với một port xác định duy nhất một tiến trình (process) trên mạng. Hay nói cách khác luồng thông tin trên mạng dựa vào IP là để xác định một máy còn port xác định 1 tiến trình trên 1 máy.

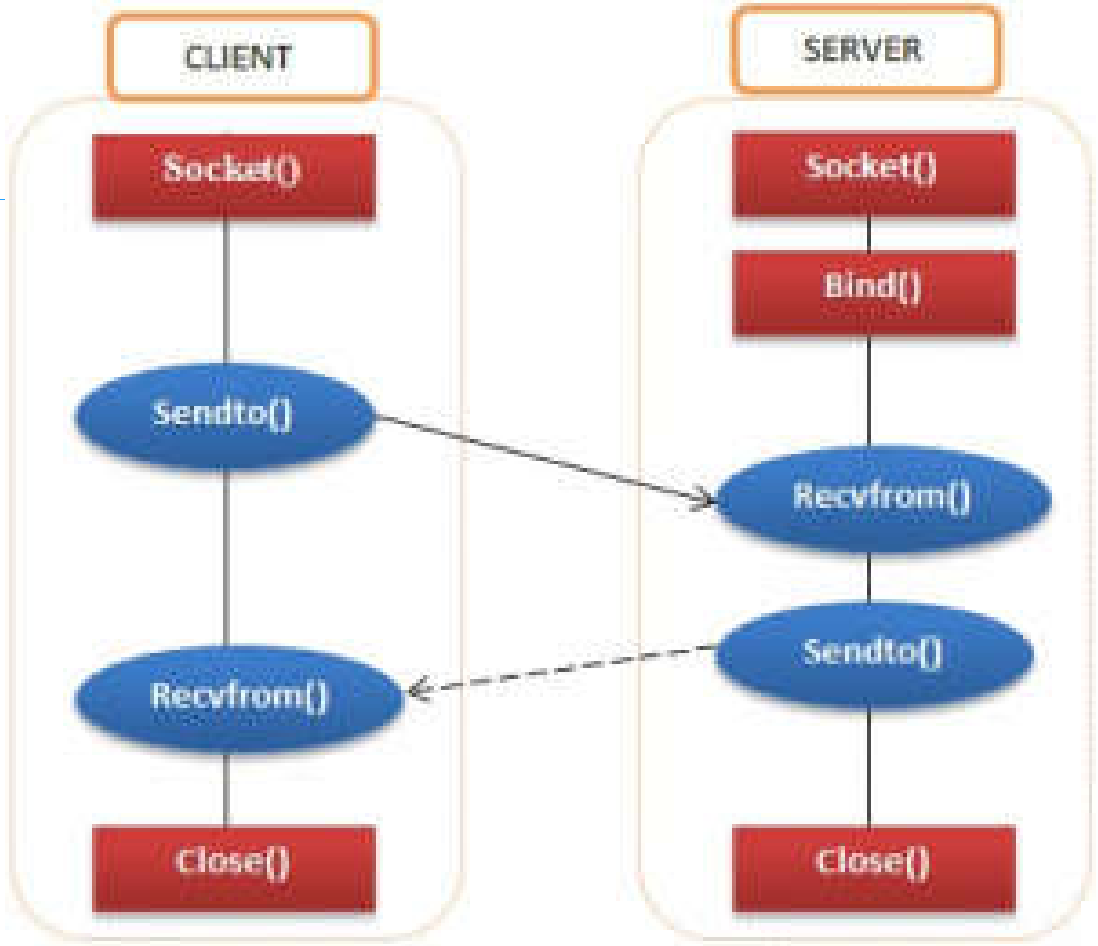
Client – Server

- Quy trình hoạt động của ứng dụng Server – Client như sau: Server có nhiệm vụ của là lắng nghe, chờ đợi kết nối từ Client trên địa chỉ IP của mình với PORT được quy định sẵn. Khi client gửi dữ liệu tới Server thì nó phải giải quyết một công việc là nhận dữ liệu đó -> xử lý -> trả kết quả lại cho Client.
- Client là ứng dụng được phục vụ, nó chỉ gửi truy vấn và chờ đợi kết quả từ Server

Sử dụng TCP


CLIENT
SERVER
Socket()
Socket()
Bind()
Connect()
Listen()
Accept()
Write()
Read()
Read()
Write()
Close()
Close()

Sử dụng UDP


CLIENT
SERVER
Socket()
Socket()
Bind()
Sendto()
Recvfrom()
Sendto()
Recvfrom()
Close()
Close()

❖ Để tạo 1 socket ta làm như sau:

s = socket.socket (socket_family, socket_type)

❖Với socket_family, socket_type đã được giới thiệu bên trên (Domain, type)

❖Sau khi khởi tạo được 1 đối tượng socket, tiếp đến ta có thể khởi tạo chương trình cho Client hoặc Server

Term	Description
domain	Họ của các giao thức dùng để truyền dữ liệu: AF_INET, PF_INET, PF_UNIX, PF_X25, …. Ở đây chúng ta thường dùng AF_INET (IPv4)
type	Kiểu của socket: SOCK_STREAM (TCP), SOCK_DGRAM (UDP)
protocol	Dùng để xác định các biến thể của các giao thức, mặc định là 0
hostname	Dùng để xác định hostname, có thể là: •Chuỗi •Địa chỉ IPv4 nếu dùng AF_INET •Địa chỉ IPv6 nếu dùng AF_INET6
port	Xác định port của socket

Server Socket Methods

Method	Description
s.bind()	Kết nối (hostname, port number) đến socket.
s.listen()	Cho Server lắng nghe các Client
s.accept()	Thiết lập kết nối giữa Client-Server

Client Socket Methods

Method	Description
s.connect()	Kết nối đến Server

General Socket Methods

Method	Description
s.recv()	Nhận dữ liệu trong giao thức TCP
s.send()	Truyền dữ liệu trong giao thức TCP
s.recvfrom()	Nhận dữ liệu trong giao thức UDP
s.sendto()	Truyền dữ liệu trong giao thức UDP
s.close()	Đóng kết nối
socket.gethostname()	Trả về tên của host

Server site

```
import socket            # Import socket module
s = socket.socket()      # Tạo đối tượng socket (mặc định là TCP trên nền IPv4)
host = socket.gethostname() # Xác định tên localhost
port = 12345             # Xác định port muốn dùng
s.bind((host, port))     # Kết nối đến socket
s.listen(5)              # Server lắng nghe 5 Client cùng 1 lúc
while True:
  c, addr = s.accept()   # Thiết lập kết nối
  print 'Got connection from', addr
  c.send('Thank you for connecting') #Gửi dữ liệu đến Client
  c.close()              # Đóng kết nối
```

Client site

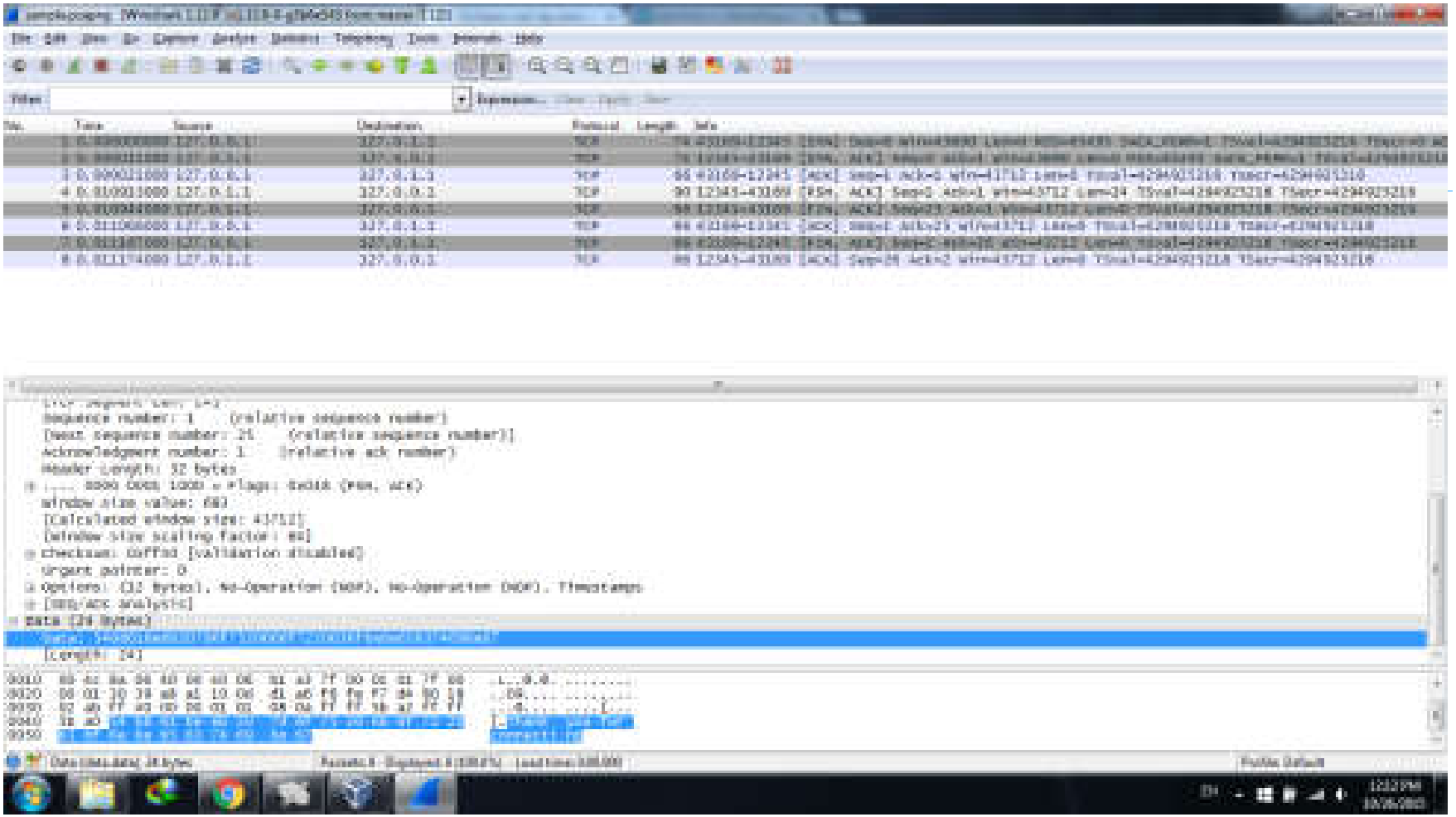
```
import socket            # Import socket module

s = socket.socket()      # Tạo đối tượng socket
host = socket.gethostname() # Xác định tên localhost
port = 12345             # Xác định port

s.connect((host, port)) #Kết nối đến Server
print s.recv(1024) #Nhận dữ liệu từ Server với Buffer là 1024 bytes
s.close()                # Đóng kết nối
```

❖ Chạy 2 script trên và đây là kết quả:

Got connection from ('127.0.0.1', 48437) #Server side

Thank you for connecting #Client side

❖ Bắt các gói tin bằng Wireshark



Python networking modules

Protocol	Common function	Port No	Python module
HTTP	Web pages	80	httplib, urllib, xmlrpclib
NNTP	Usenet news	119	nntplib
FTP	File transfers	20	ftplib, urllib
SMTP	Sending email	25	smtplib
POP3	Fetching email	110	poplib
IMAP4	Fetching email	143	imaplib
Telnet	Command lines	23	telnetlib
Gopher	Document transfers	70	gopherlib, urllib

Bài tập thực hành

- **Bài 1**: Tạo một lớp tcpserver.py và 1 lớp tcpclient.py kết nối thông qua port 8090. Từ client gửi lên thông điệp '*From CLIENT TCP*', từ server nhận thông điệp và trả về '*From SERVER TCP*'. In ra các thông điệp đó tương ứng tại server và client.
- **Bài 2**: Tạo một lớp tcpserver.py và 1 lớp tcpclient.py kết nối thông qua port 8091. Từ client gửi lên 2 số nguyên a và b, tại server tính tổng và trả về client giá trị của tổng của a + b.

❖**Bài 3**: Viết chương trình theo mô hình client-server để kiểm tra tính hợp lệ của mật khẩu mà người dùng nhập vào (mật khẩu được nhập từ bàn phím tại Client; hàm xử lý kiểm trả mật khẩu hợp lệ viết tại Server).

❖Các tiêu chí kiểm tra mật khẩu bao gồm:

- 1. Ít nhất 1 chữ cái nằm trong [a-z]
- 2. Ít nhất 1 số nằm trong [0-9]
- 3. Ít nhất 1 kí tự nằm trong [A-Z]
- 4. Ít nhất 1 ký tự nằm trong [$ # @]
- 5. Độ dài mật khẩu tối thiểu: 6
- 6. Độ dài mật khẩu tối đa: 12

Chương trình phải chấp nhận một chuỗi mật khẩu phân tách nhau bởi dấu phẩy và kiểm tra xem chúng có đáp ứng những tiêu chí trên hay không. Mật khẩu hợp lệ sẽ được in ra tại Client, mỗi mật khẩu cách nhau bởi dấu phẩy. Ví dụ mật khẩu nhập vào chương trình là: *Abc123@1,aF1#2w3E*,2We3345*. Thì đầu ra sẽ là: *Abc123@1*

❖THANKS FOR ATTENDING!