



DÀI HỌC ĐÀ NẴNG

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG VIỆT - HÀN  
VIETNAM - KOREA UNIVERSITY OF INFORMATION AND COMMUNICATION TECHNOLOGY

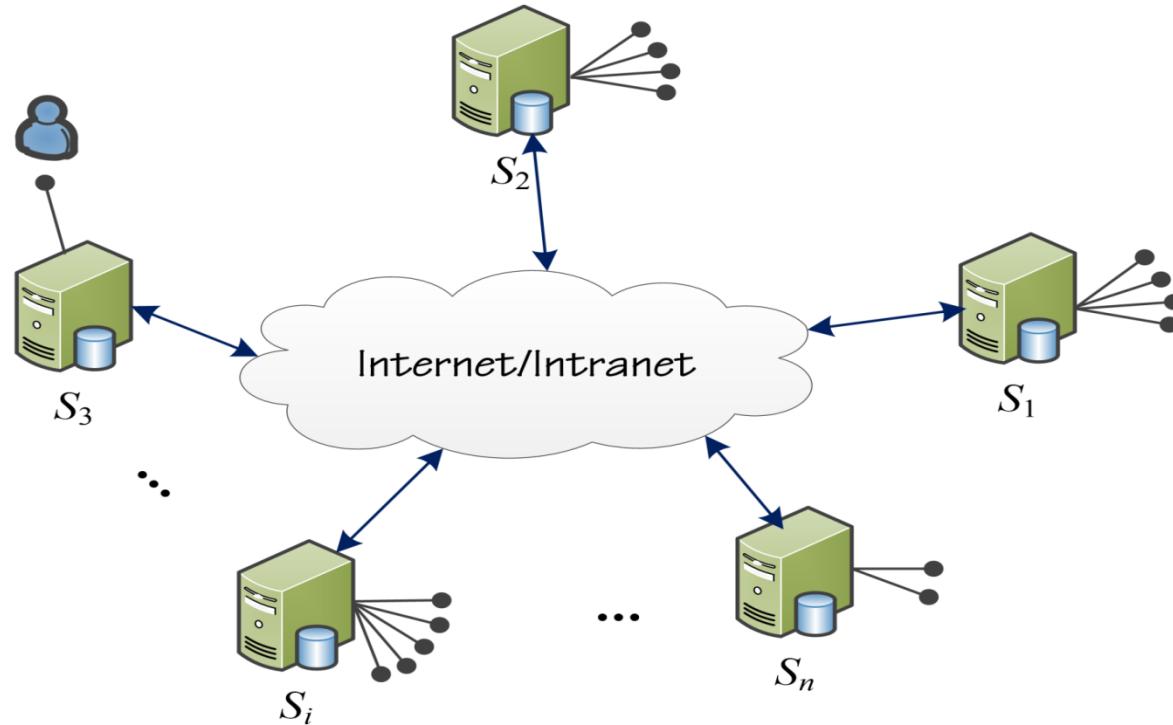
한-베정보통신기술대학교

# Chapter 6

# Network communication



Computer Science

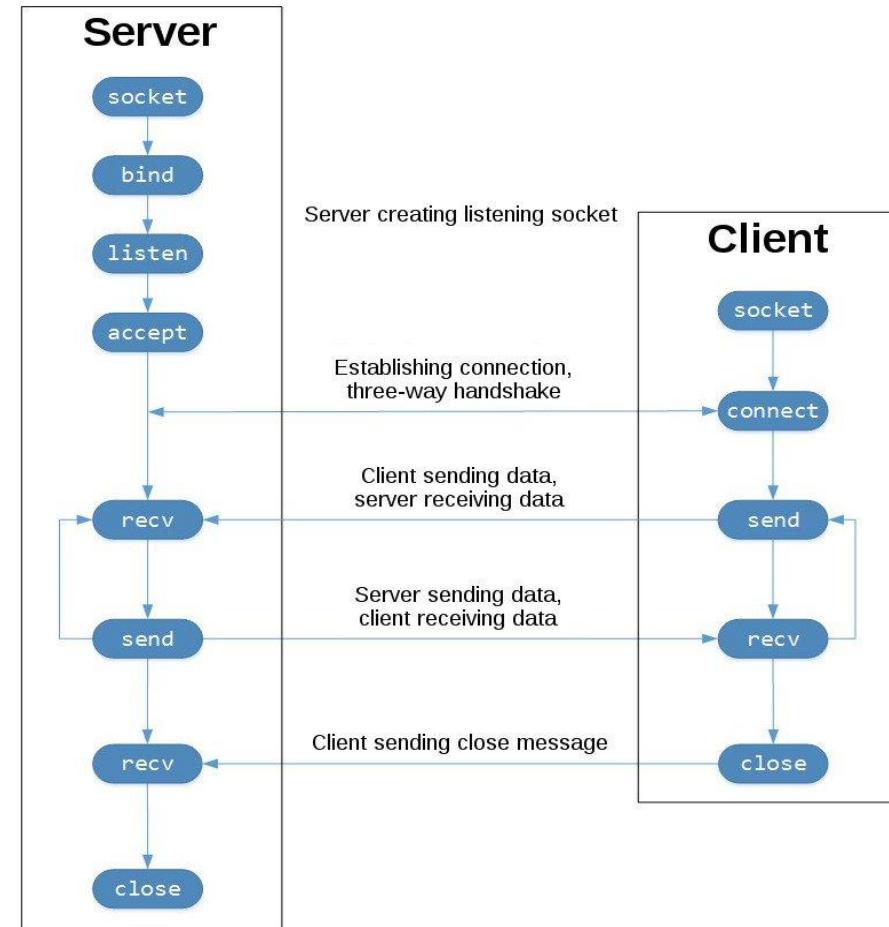


Connection model in Distributed System

- FTP (File transfer Protocol)
- Telnet: The terminal allows login to the server.
- SMTP (Simple Mail Transfer Protocol)
- DNS (Domain Name Service)
- SNMP (Simple Network Management Protocol)
- TCP (Transmission Control Protocol)
- IP (Internet Protocol)

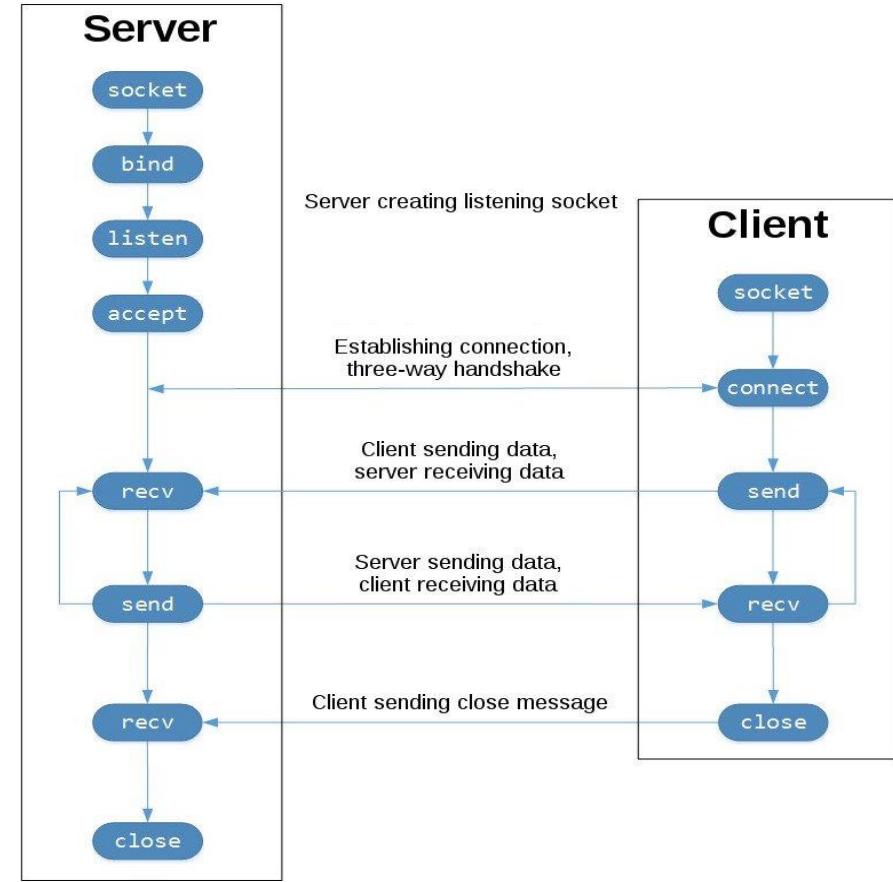
## Server:

- Used to **socket()** method to create a socket object
- Used to **bind()** method to create to bind the address and service port.
- Used to **listen()** method to listen and wait for signal from Client
- Used to **accept()**: method accept connection from Client, return 02 values :
  - A Connection with Client
  - Address of Client
- Used to **send()/ recv()** method to send/receive messages. The return value of these methods is bytes
- Used to **close()** method to close socket at Client



## Client:

- Used to **socket()** method to create a socket object
- Used to **connect()** method to create a connection from client to Server
- Used to **send()/recv()** method to send/receive messages. The return value of these methods is bytes
- Used to **close()** method to close socket at Client



## Server:

```
import socket
HOST = '127.0.0.1'
PORT = 1400
socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
socket.bind((HOST, PORT))
socket.listen(1)      # number of connection, windows: from 1 to 5 connection
print("Wait from Client \n")
socket, add_client = socket_server.accept()
print('Connected to address: ', add_client)
while (1):
    data_from_client = socket.recv(1024)
    if not data_from_client: break
    answer1 = "Receive a Message from Client, That is : ".encode()
    answer2 = ". Goode Bye, See you again".encode()
    socket.send(answer1 + data_from_client + answer2)
socket.close()
```

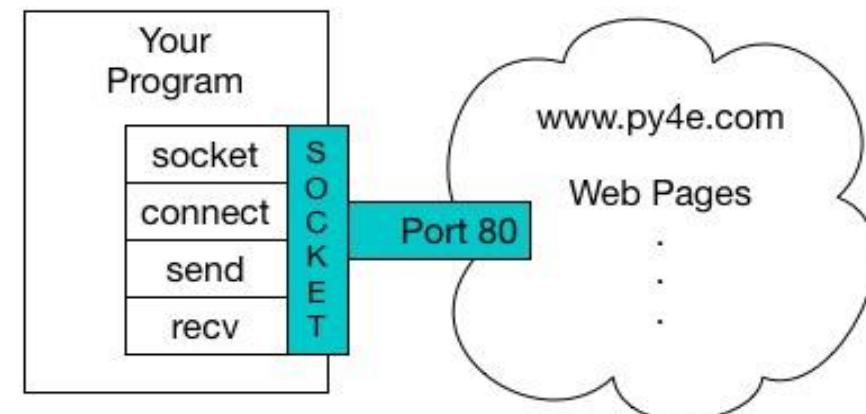
## Client:

```
import socket  
  
HOST = '127.0.0.1'  
  
PORT = 1400  
  
socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
socket.connect((HOST, PORT))  
  
send = input("Hi, Server: ").encode()  
socket.send(send)  
  
data_from_server = socket.recv(1024) #1024 Size  
print(data_from_server.decode())  
socket.close()
```

# Port corresponding to a network service

Service	Port
FTP-Data	20
FTP-Control	21
SSH	22
Telnet	23
SMTP (Mail)	25
HTTP (WWW)	80
POP3	110
IMAP	143
HTTPS (Secure WWW)	443

- Used to Steps
  - Step 1: Used to **socket()** method to a socket object
  - Step 2: Used to **connect()** method to create a connection between client and Server
  - Step 3: Used to **send()** method to send a HTTP request to Web Sever
  - Step 4: Used to **recv()** method to receiver web page (HTML) or a file
  - Steps to **close()** method to close socket



# Lab 6-2: Access to Website by Socket

- Access to <http://data.pr4e.org>

```
import socket

mysock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
mysock.connect(('data.pr4e.org', 80))
cmd = 'GET http://data.pr4e.org HTTP/1.0\r\n\r\n'.encode()
mysock.send(cmd)

while True:
    data = mysock.recv(512)
    if len(data) < 1:
        break
    print(data.decode())
mysock.close()
```

# Lab 6-3: Access to Website by Socket

- Access to a text file at <http://data.pr4e.org/romeo.txt>

```
import socket

mysock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
mysock.connect(('data.pr4e.org', 80))
cmd = 'GET http://data.pr4e.org/romeo.txt
HTTP/1.0\r\n\r\n'.encode()

mysock.send(cmd)

while True:
    data = mysock.recv(512)
    if len(data) < 1:
        break
    print(data.decode())
mysock.close()
```

# VKU Lab 6-4: Access to Website by Urlopen() method

- Access to file

```
import urllib.request  
fhand = urllib.request.urlopen('http://data.pr4e.org/romeo.txt')  
for line in fhand:  
    print(line.decode().strip())
```

- Access to http://www.dr-chuck.com/page1.htm

```
import urllib.request  
fhand = urllib.request.urlopen('http://www.dr-chuck.com/page1.htm')  
for line in fhand:  
    print(line.decode().strip())
```

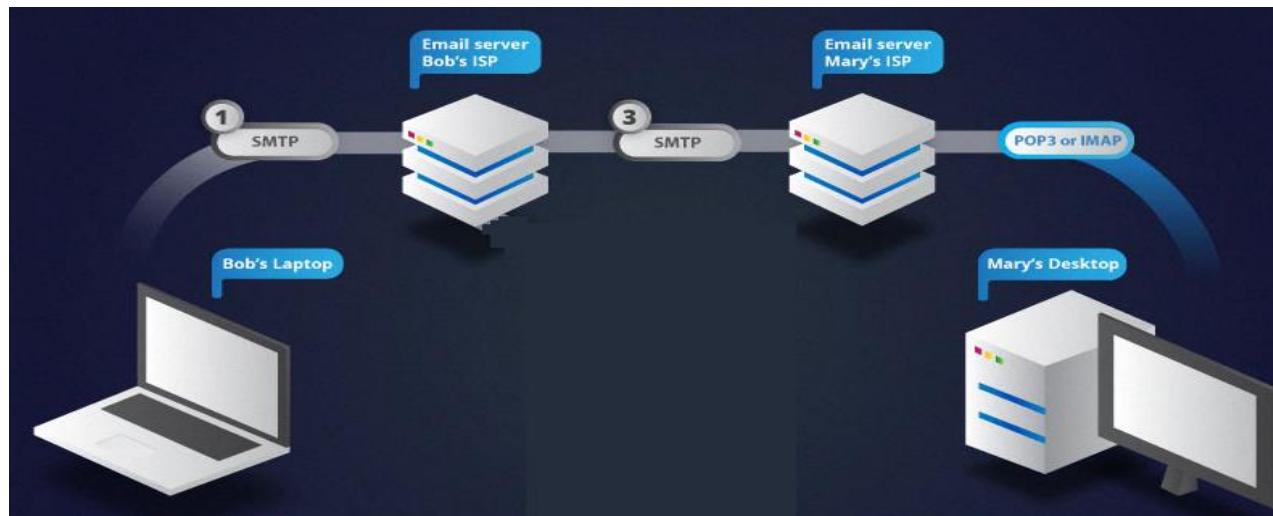
# VKU Lab 6-5: Access to Website by Urlopen() method

- Access to a file and return the frequency of occurrence of words in a file

```
import urllib.request
fhand = urllib.request.urlopen('http://data.pr4e.org/romeo.txt')
Dics_word = dict()

for line in fhand:
    word_in_line = line.decode().split()
    for word in word_in_line:
        Dics_word[word] = Dics_word.get(word, 0) + 1
print(Dics_word)
```

- SMTP (Simple Mail Transfer Protocol) is a standard TCP/IP protocol
- Used to transfer e-mail via internet.
- It establishes a connection channel between mail client and mail server; between the sending mail server and the receiving mail server.
- Email is pushed from a sending mail client to sending mail server → from sending mail server to receiving mail server:



- Sử dụng thư viện smtplib

- The ports of SMTP:

- Port 25: not encode
    - Port 465: SSL - Secure Sockets Layer
    - Post 587: TLS - Transport Layer Security.

- Các bước gửi thư:

- Step 1: import smtplib library to used to SMTP:

```
import smtplib
```

- Step 2: Create to a connection smtp with SSL encode

```
connect = smtplib.SMTP_SSL("address ip of mail server", 465)
```

- Step 3: Login sending mail:

```
connect.login(sender_email, password)
```

- Step 4: Send

```
connect.sendmail(sender_email, receiver_email, text)
```

```
import smtplib  
  
sender_email = "python02021972@gmail.com"  
password = "123456"  
  
receiver_email = minhnhatvh@gmail.com  
text = "Hi, How are you? "  
  
connect = smtplib.SMTP_SSL("smtp.gmail.com", 465)  
connect.login(sender_email, password)  
connect.sendmail(sender_email, receiver_email, text)
```

Note: To send to gmail:

Login mail and access link to turn on the less safe mode

<https://myaccount.google.com/lesssecureapps>

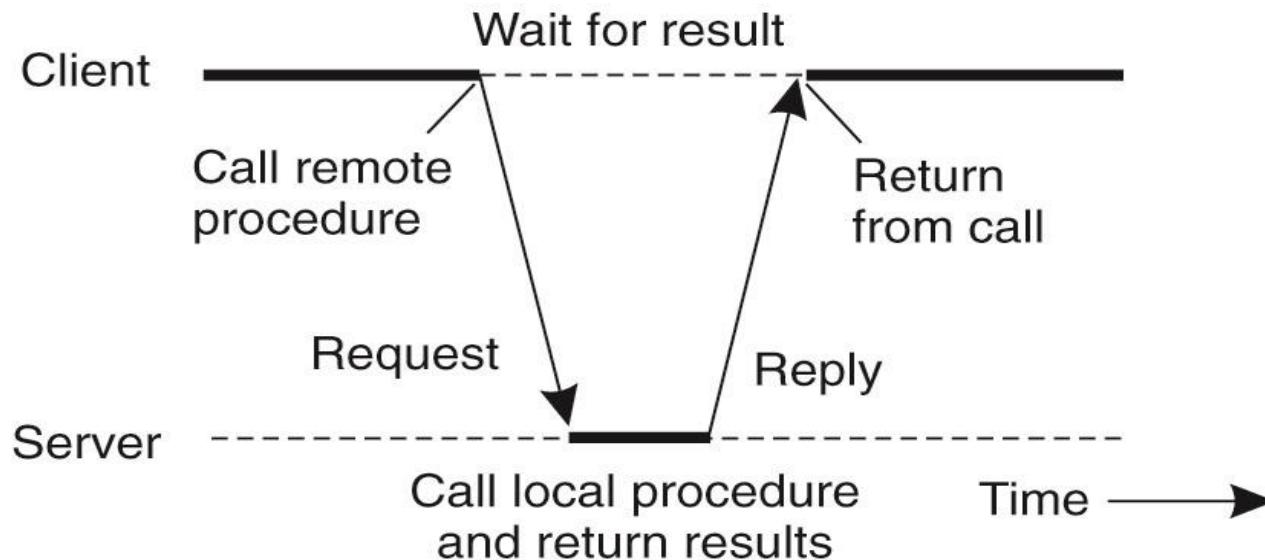
# Lab 6-7: Client/Server Commuication by Socket (many client)

```
# Server
import socket
HOST = '127.0.0.1'
PORT = 1400
Socket= socket.socket(socket.AF_INET, socket.SOCK_STREAM)
Socket.bind((HOST, PORT))
Socket.listen(2)
print("Server is ready !!!")
while True:
    conn, addr= Socket.accept()
    Received_from_client = conn.recv(1024)
    print("Message of client: ", Received_from_client.decode())
    message = input("Enter message to send client: ")
    conn.send(message.encode())
    x = input("Shutdown Server? y/n: ")
    if x == "y":
        conn.close()
        break
```

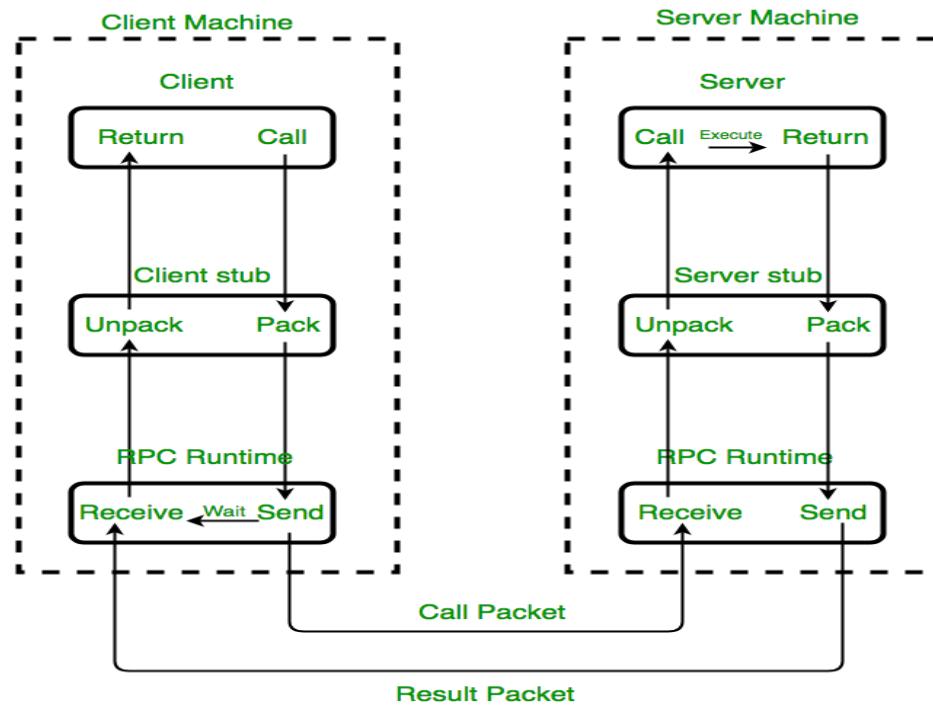
```
# Client 1
import socket
HOST = '127.0.0.1'
PORT = 1400
Socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
Socket.connect((HOST, PORT))
while True:
    message = input("Client 1 send a message to server: ")
    Socket.send(message.encode())
    data_from_server = Socket.recv(1024)
    print("message of Server sending: ", data_from_server.decode())
    x = input("Shutdown Client ? y/n: ")
    if x=="y":
        Socket.close()
        break
```

```
# Client 2
import socket
HOST = '127.0.0.1'
PORT = 1400
Socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
Socket.connect((HOST, PORT))
while True:
    message = input("Client 2 send a message to server: ")
    Socket.send(message.encode())
    data_from_server = Socket.recv(1024)
    print("message of Server sending: ", data_from_server.decode())
    x = input("Shutdown Client ? y/n: ")
    if x=="y":
        Socket.close()
        break
```

- Remote Procedure Calls (RPC): A technique for transmitting information between separate programs in a distributed system/Network.



- Step to Remote Procedure Calls (RPC):
  - **Client stub:** the process to execute the procedure at the client
  - **Server stub:** the process to execute the procedure at the Server



```
# Server
from xmlrpc.server import SimpleXMLRPCServer

def is_even(n):
    return n**2

server = SimpleXMLRPCServer(("localhost", 8000))
print(" Listen to the signal from the port 8000...")
server.register_function(is_even)
server.serve_forever()
```

Listen to the signal from the port 8000...

127.0.0.1 - - [05/Dec/2020 14:16:01] "POST / HTTP/1.1" 200 -

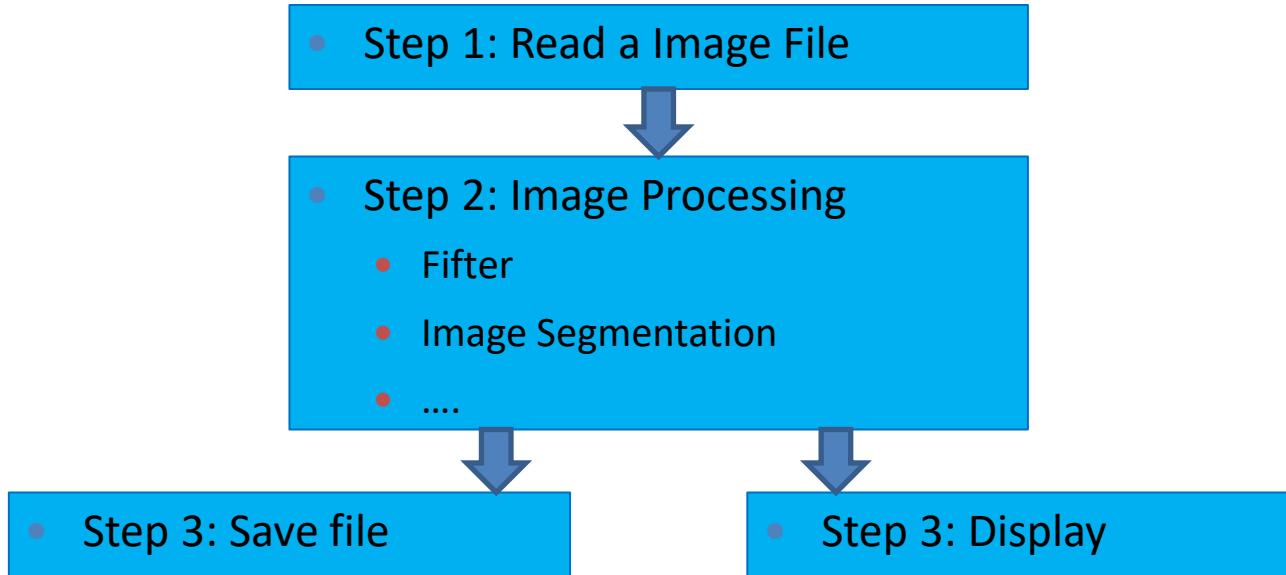
# Lab 6-8: Remote Procedure Calls

```
# Client
import xmlrpclib

with xmlrpclib.ServerProxy("http://localhost:8000/") as proxy:
    print("Square a number ")
    a = int(input("Enter a positive integer :"))
    print(" Square a number is:", str(proxy.is_even(a)))
```

Square a number  
Enter a positive integer: 34

Square a number is: 1156



- Install Pillow Library: Pip install Pillow
- Open

```
from PIL import Image      # import Image class  
img = Image.open("VD.jpg")  # Open image file VD.jpg
```

- Display: img.show()
- Rotate: img\_rotate = img.rotate(90)
- Save: img\_rotate.save("VD\_new.jpg")
- Thumbnail: img.thumbnail((400, 600))
- Convert to grayscale image : img\_trang\_den = img.convert("L")
- Crop : img\_cat\_xen = img.crop((100, 100, 350, 350))
- Resize: img\_resize = img.resize((5000, 1000))

- Filter:

```
from PIL import Image, ImageFilter  
img = Image.open("mau3.jpg")  
img_smoothed = img.filter(ImageFilter.SMOOTH)  
blurred = img.filter(ImageFilter.BLUR)  
img_sharpened = img.filter(ImageFilter.SHARPEN)  
contoured = img.filter(ImageFilter.CONTOUR)  
emboss = img.filter(ImageFilter.EMBOSS)
```

- Contrast

```
from PIL import Image, ImageEnhance  
img = Image.open("mau3.jpg")  
img_contrast = ImageEnhance.Contrast(img)  
  
img_contrast.enhance(2.0).save("mau3_img_contrast.jpg")  
img_new = Image.open("mau3_img_contrast.jpg")  
img_new.show()
```

- Merge

```
from PIL import Image  
img1 = Image.open("maul.jpg")  
img2 = Image.open("mau2.jpg")  
img1.paste(img2)
```

- Blend two images: Sử dụng phương thức blend()

```
from PIL import Image
image1 = Image.open("mau1.jpg")
image2 = Image.open("mau2.gif")
image3 = image1.resize((800, 500))
image4 = image2.resize((800, 500))
image5 = image3.convert("RGBA")
image6 = image4.convert("RGBA")
alphaBlended = Image.blend(image5, image6, alpha=.2)

alphaBlended1.show()
```

### Lab 6-9: Used to Remote Procedure Calls to Image Processing