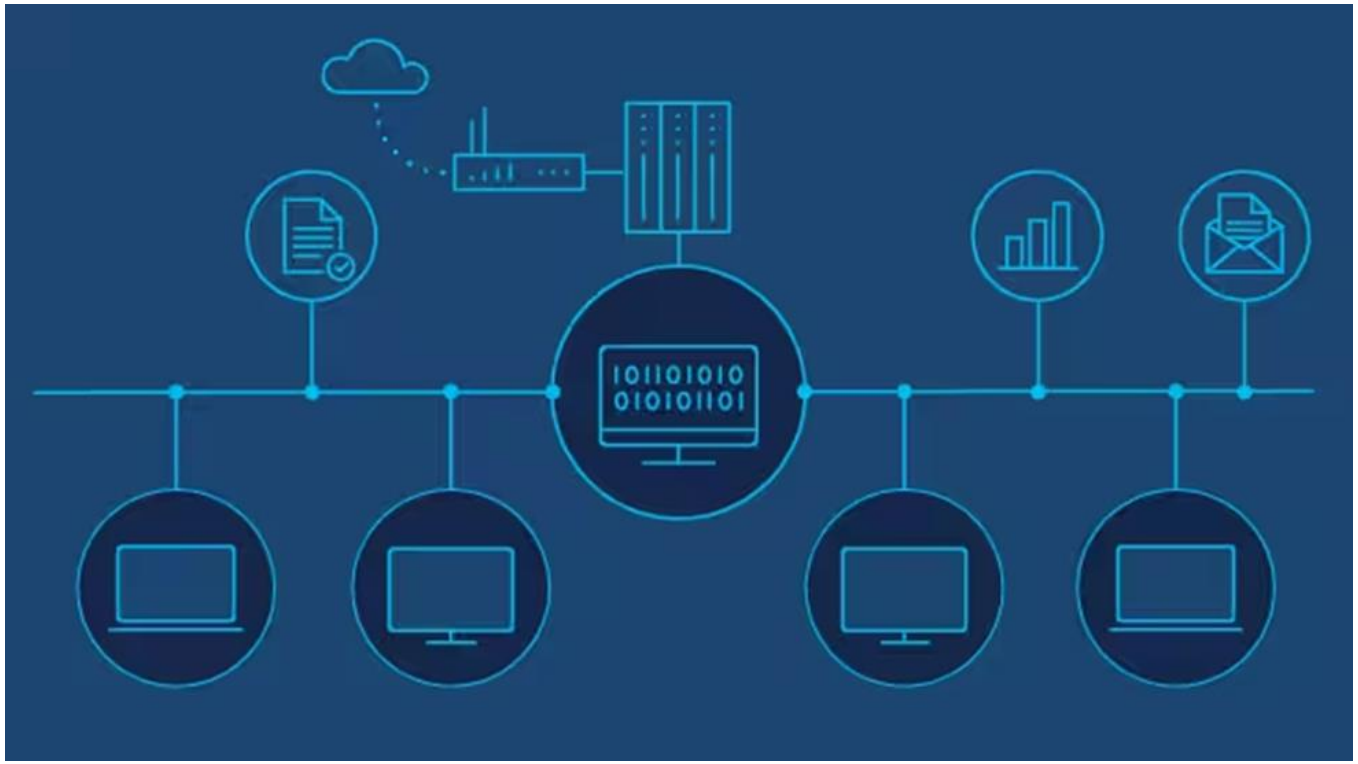
# Bài 1 và 2
# Tổng quan về lập trình mạng và lập trình Socket



# Ví dụ 1: Lập trình mạng

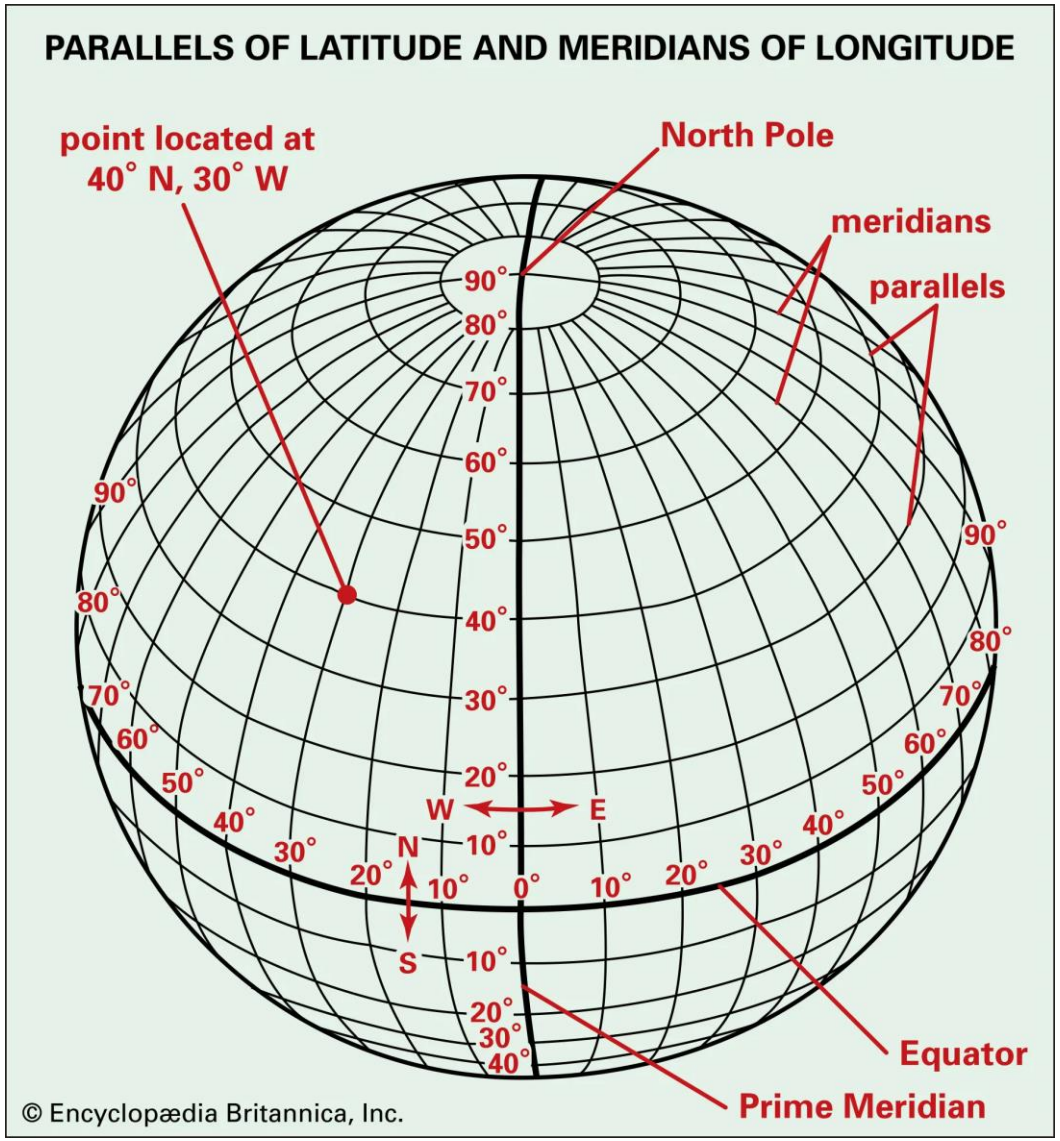
Tìm tọa độ, bao gồm vĩ độ và kinh độ, của "Bac Tu Liem District" từ Trang web sau:

https://nominatim.openstreetmap.org/ (130.117.76.5, 443).

- Giải pháp 1: Sử dụng một **trình duyệt Web** (ví dụ: Google Chrome).
- Giải pháp 2: Sử dụng một **chương trình chạy trên máy khách** dựa trên **lập trình mạng**

# Ví dụ 1: Lập trình mạng

- Giải pháp 1: Sử dụng một **trình duyệt Web** (ví dụ: Google Chrome).
- Giải pháp 2: Sử dụng một **chương trình chạy trên máy khách** dựa trên **lập trình mạng**



# Ví dụ 2: Lập trình mạng

- Gửi email với hai tập tin đính kèm (content.rar, content.txt)
  - Từ: laptrinhmang@outlook.com
  - Đến: laptrinhmang.haui@outlook.com, laptrinhmang.haui@gmail.com
- Nội dung Email:

*Dear Professor,*

*Example for introducing Network programming course. I has sent you research report as in the attached files.*

*Sincerely.*

# Bài 1 và 2: Lộ trình

- Góc nhìn của lập trình viên về Internet
- Mô hình Client-Server
- Socket
- Lập trình Socket với UDP và TCP

# Bài 1 và 2: Lộ trình

- Góc nhìn của lập trình viên về Internet
- Mô hình Client-Server
- Socket
- Lập trình Socket với UDP và TCP

# Góc nhìn của lập trình viên về Internet

- 1. Hosts được ánh xạ tới địa chỉ IP 32 bit hoặc 128 bit
  - 134.173.42.2
  - 2001:1878:301:902:214:22ff:fe7c:883f
- 2. Địa chỉ IP được ánh xạ tới các nhận dạng được (identifiers) được gọi là tên miền Internet
  - 116.104.86.99 ánh xạ tới dhcnhn.vn
  - 103.140.38.3 ánh xạ tới haui.edu.vn
- 3. Tiến trình trên một host Internet có thể giao tiếp với tiến trình trên host khác qua các kết nối

# Một số ứng dụng mạng

- Mạng xã hội
- Web
- Nhắn tin văn bản
- E-mail (Thư điện tử)
- Trò chơi người người qua Internet
- Phát video trực tuyến (YouTube, Hulu, Netflix)
- Chia sẻ tập tin P2P
- Truyền giọng nói qua IP (e.g., Skype)
- Hội nghị truyền hình thời gian thực (e.g., Zoom)
- Tìm kiếm Internet
- Đăng nhập từ xa
- …

*Q: Sở thích của bạn*?

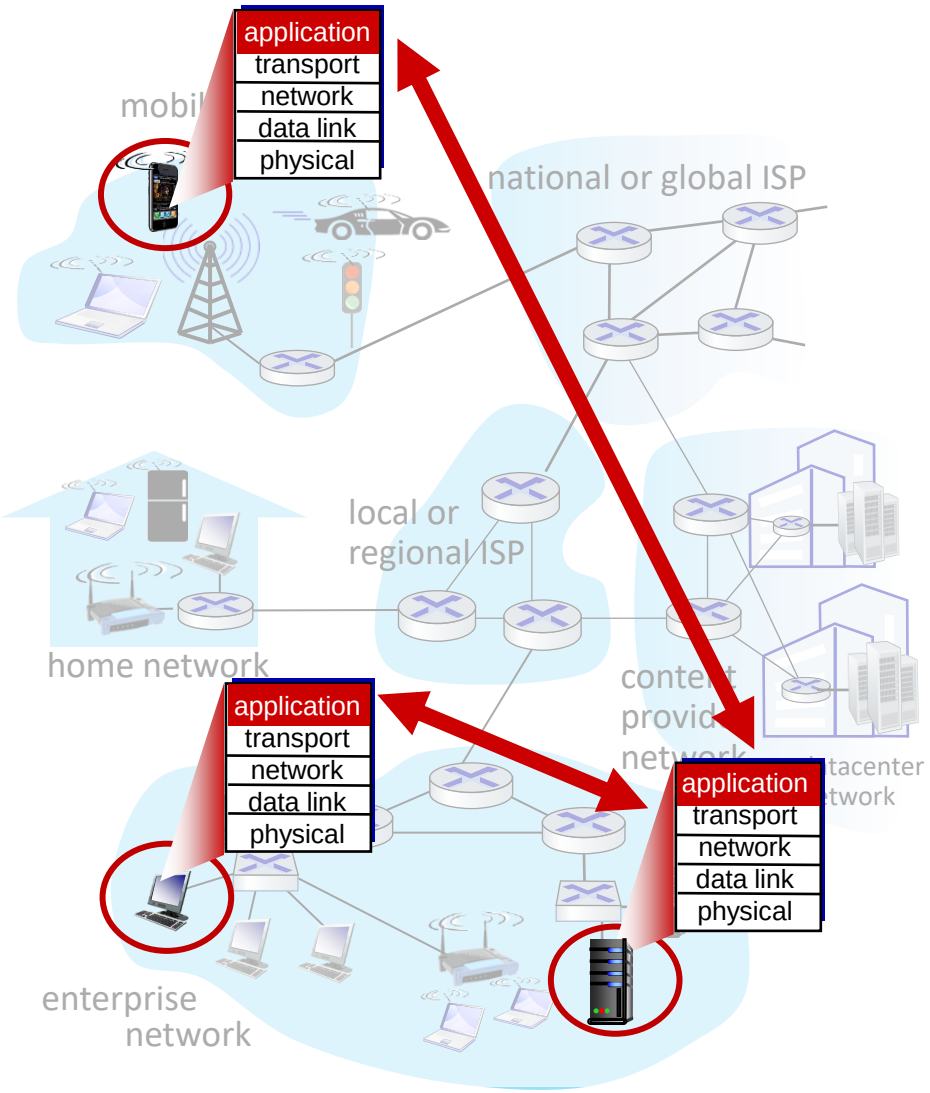
# Tạo một ứng dụng mạng

## Viết chương trình để:

- Chạy trên các hệ thống đầu cuối (khác nhau)
- Liên lạc qua mạng
- Ví dụ: phần mềm máy chủ web giao tiếp với trình duyệt

## Không cần viết phần mềm cho các thiết bị trong lõi của mạng

- Các thiết bị trong lõi mạng không chạy các ứng dụng người dùng
- Các ứng dụng trên các hệ thống đầu cuối cho phép phát triển ứng dụng và lan truyền nhanh chóng



# Bài 1 và 2: Lộ trình

- Góc nhìn của lập trình viên về Internet
- Mô hình Client-Server
- Socket
- Lập trình Socket với UDP và TCP

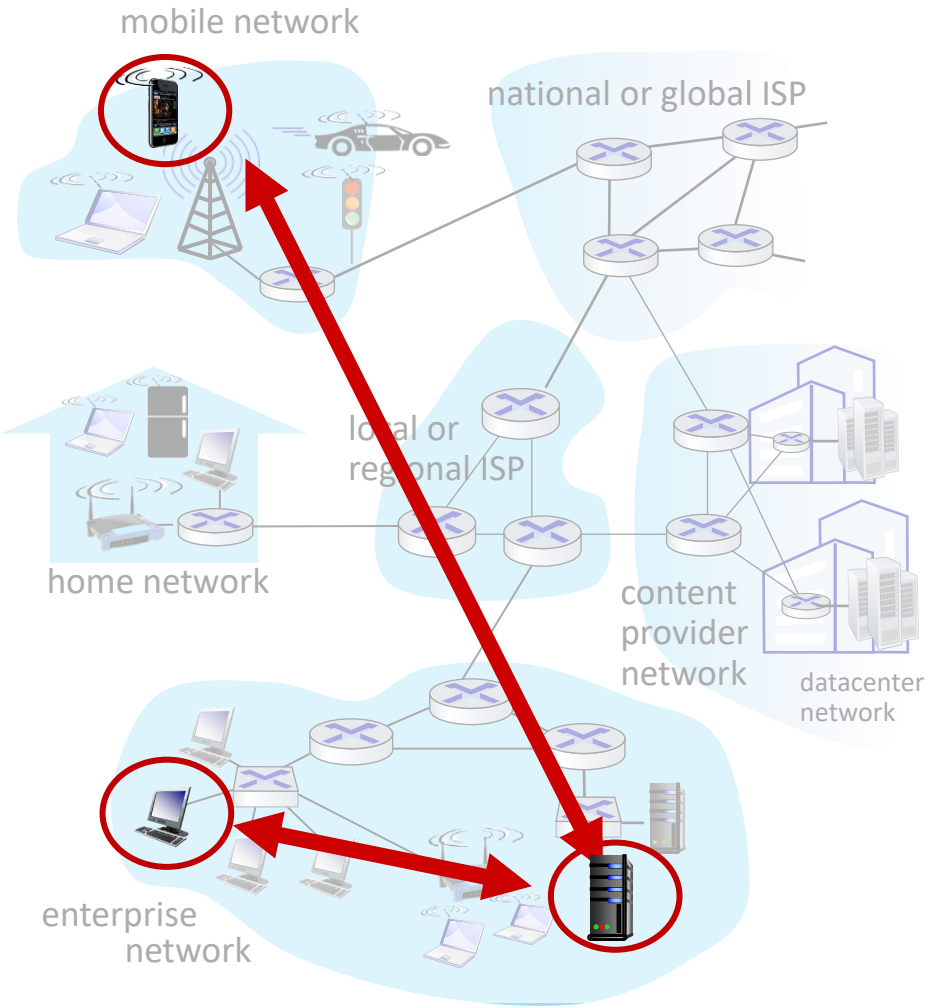
# Mô hình Client-Server

## Servers:

- Luôn luôn hoạt động
- Địa chỉ IP “cố định”
- Thường ở trong các trung tâm dữ liệu, để mở rộng quy mô

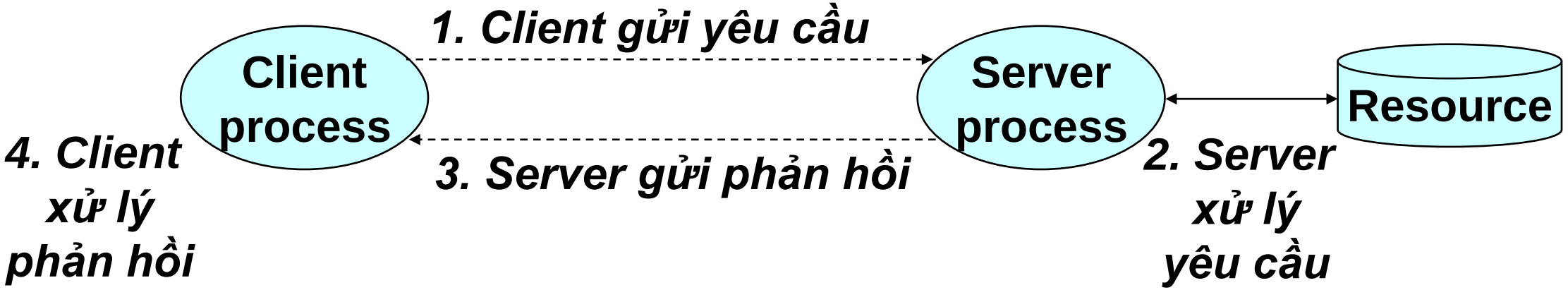
## Clients:

- Kết nối, giao tiếp với server
- Có thể kết nối không liên tục
- Có thể thay đổi địa chỉ IP
- Không giao tiếp trực tiếp với nhau
- Ví dụ: HTTP, IMAP, FTP



# Giao tiếp giữa Client và Server

- Hầu hết ứng dụng mạng đều dựa trên mô hình máy khách-máy chủ:
  - Tiến trình Server và một hoặc nhiều tiến trình Client
  - Server quản lý một số tài nguyên
  - Server cung cấp dịch vụ bằng cách xứ lý, phân phối tài nguyên cho Client




***Lưu ý: Clients và Servers các tiến trình chạy trên Hosts (có thể cùng hoặc khác host).***

# Giao tiếp giữa tiến trình (Processes communicating)

*Process (tiến trình):* chương trình chạy trên 1 host

- Với cùng 1 host, hai tiến trình giao tiếp với nhau bằng các sử dụng giao tiếp liên tiến trình - inter-process communication (xác định bởi hệ điều hành)
- Các tiến trình trong các hosts khác nhau giao tiếp bằng cách trao đổi các bản tin

clients, servers

*client process:* tiến trình khởi tạo kết nối

*server process:* tiến trình chờ để được kết nối

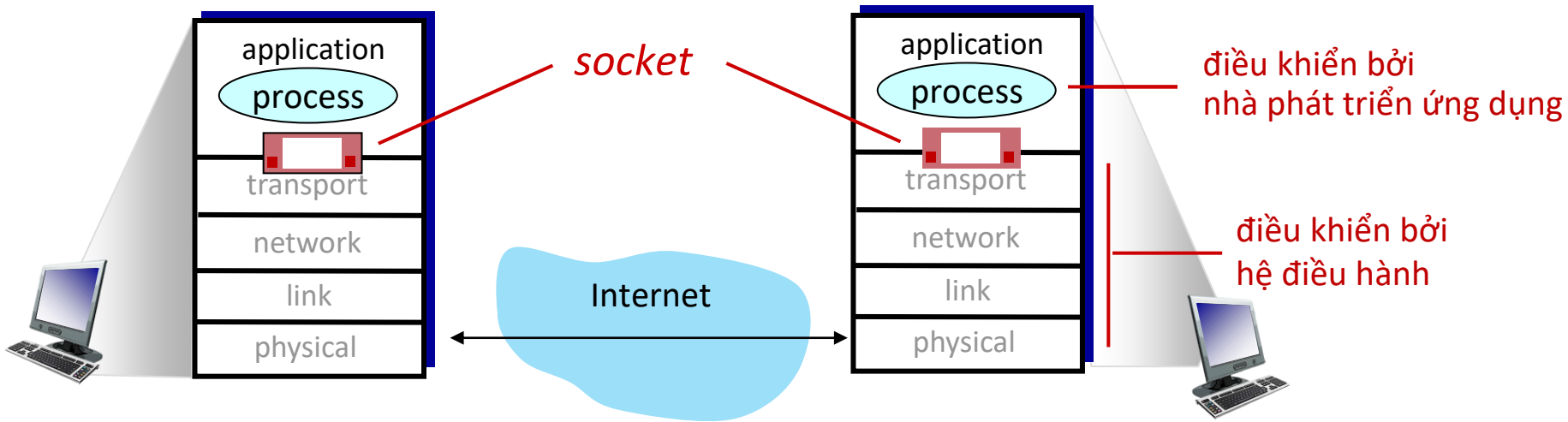
- Chú ý: các ứng dụng với kiến trúc P2P có cả tiến trình client và server

# Bài 1 và 2: Lộ trình

- Góc nhìn của lập trình viên về Internet
- Mô hình Client-Server
- Socket
- Lập trình Socket với UDP và TCP

# Sockets

- Tiến trình gửi/nhận bản tin đến/từ socket của nó
- Socket tương tự như cửa ra vào
  - Quá trình gửi đẩy bản tin ra khỏi cửa
  - Quá trình gửi dựa vào cơ sở hạ tầng vận chuyển ở phía bên kia cửa để gửi tin nhắn đến socket tại quá trình nhận
  - Hai socket giao tiếp với nhau: một socket ở mỗi bên.



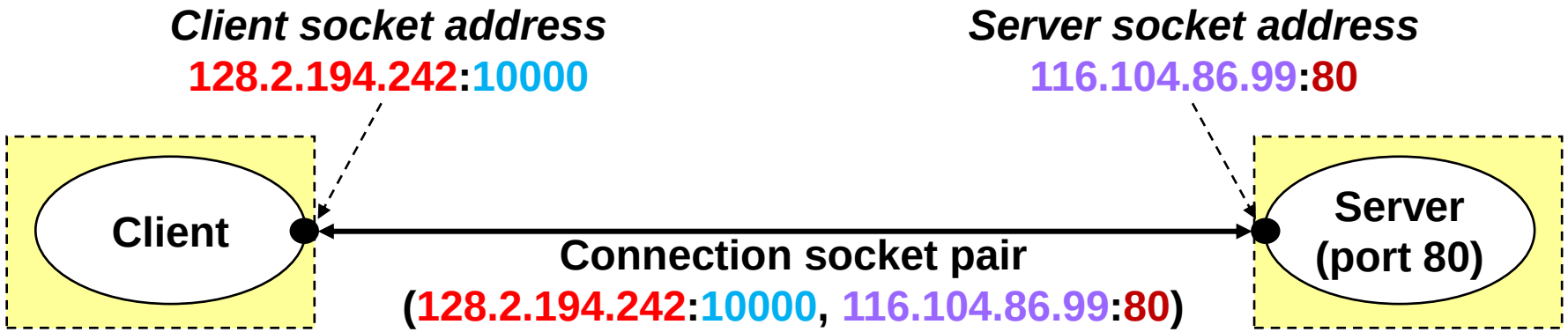
# Địa chỉ của tiến trình (Addressing processes)

- Để nhận bản tin, tiến trình phải có *định danh*
- Host có địa chỉ IP 32 bit duy nhất trên một giao diện
- *Q:* địa chỉ IP của host mà tiến trình đang chạy trên đó có đủ để xác định tiến trình đó là tiến trình nào hay không?
  - *A:* không, *nhiều tiến trình* có thể chạy đồng thời trên cùng một host

- *Định danh* bao gồm cả địa chỉ IP và cổng dịch vụ (port numbers) được liên kết với tiến trình trên giao diện của host.
- Ví dụ về cổng dịch vụ:
  - HTTP server: 80
  - mail server: 25
- Để gửi bản tin HTTP đến máy chủ Web dhcnhn.vn:
  - IP address: 128.119.245.12
  - port number: 80
- Còn nữa ....

# Kết nối Internet

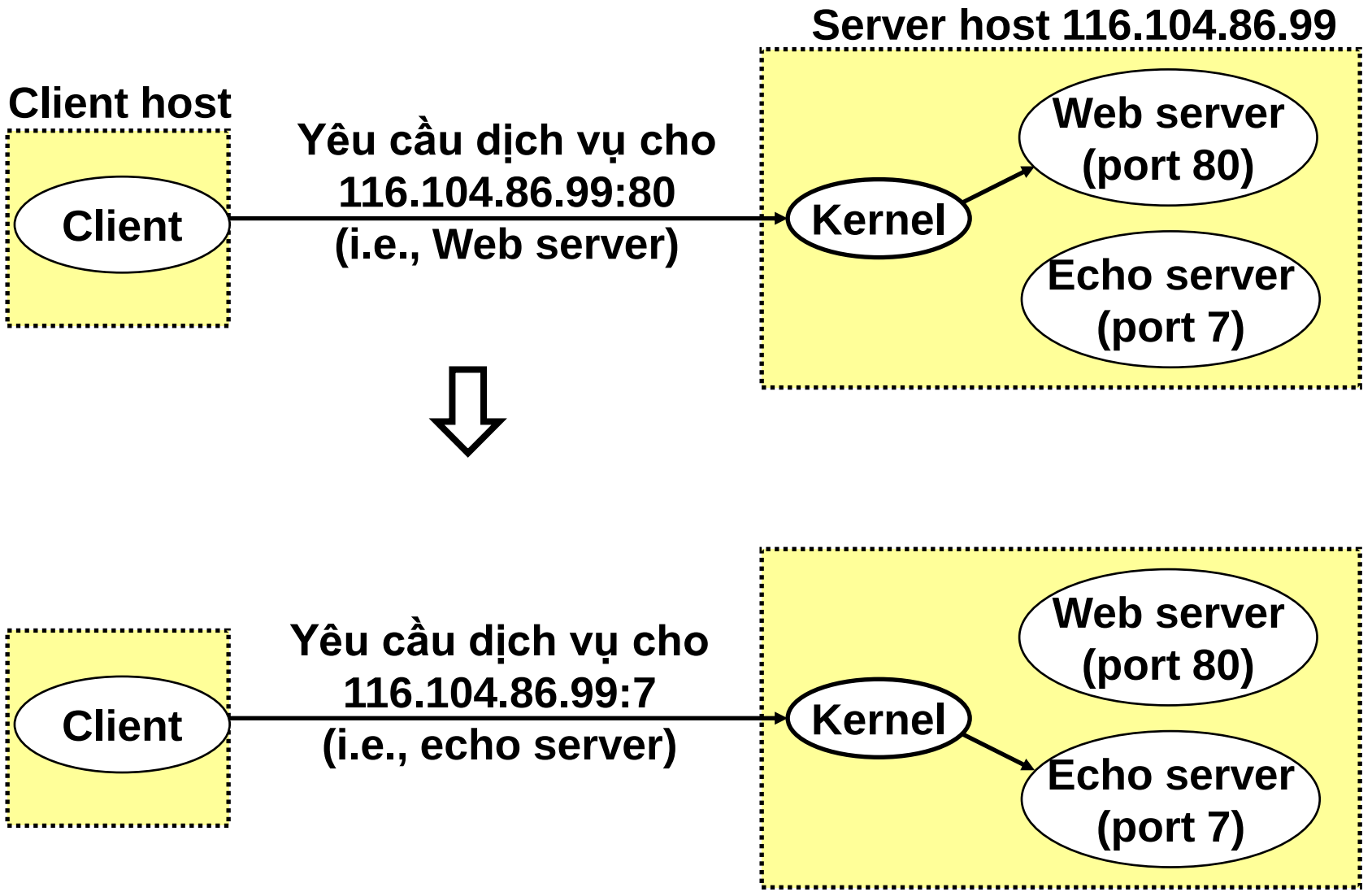
- Clients và servers giao tiếp với nhau bằng cách gửi dòng dữ liệu dưới dạng byte qua các kết nối
- Kết nối là điểm đến điểm, truyền song công (giao tiếp 2 chiều) và đáng tin cậy




*Chú ý: 10000 là cổng dịch vụ ngắn hạn, được dùng tạm thời và được phân bổ bởi hệ điều hành*

*Chú ý: 80 là cổng dịch vụ phổ biến được liên kết với máy chủ Web*

# Sử dụng cổng để xác định dịch vụ



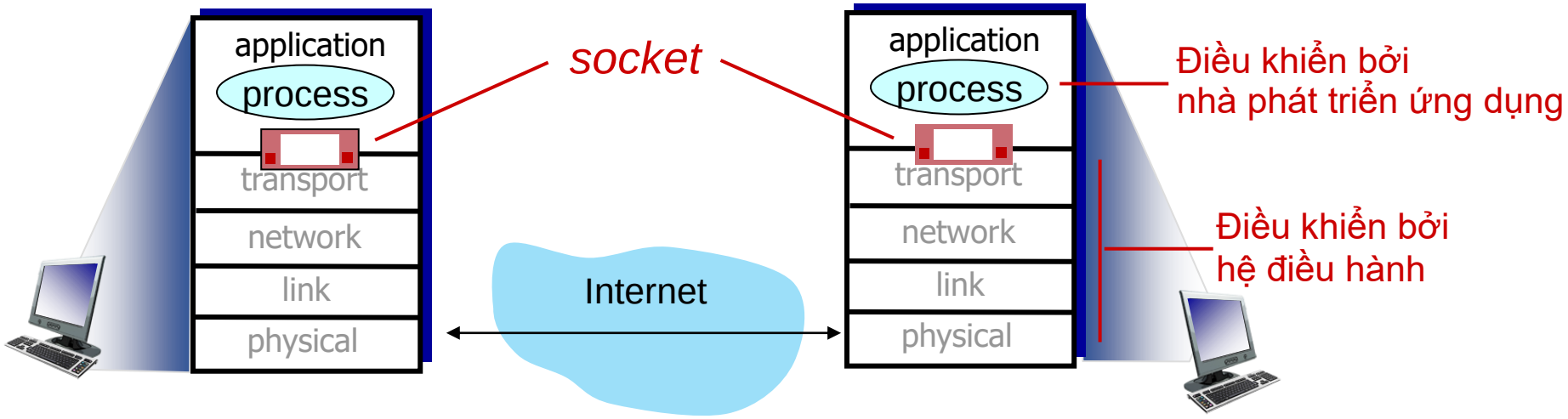
# Bài 1 và 2: Lộ trình

- Góc nhìn của lập trình viên về Internet
- Mô hình Client-Server
- Socket
- Lập trình Socket với UDP và TCP

# Lập trình Socket

*Mục tiêu:* Hiểu cách xây dựng các ứng dụng client/server giao tiếp với nhau bằng cách sử dụng socket

*Socket:* Cánh cửa giữa các tiến trình ứng dụng và giao thức truyền vận đầu cuối



# Lập trình Socket

Hai loại socket cho hai dịch vụ truyền vận:

- *UDP:* gói tin datagram không tin cậy
- *TCP: gói tin* đáng tin cậy, có hướng theo luồng byte

Ví dụ về ứng dụng lập trình socket:

1. Client gửi dữ liệu văn bản đến Server
2. Server nhận dữ liệu và chuyển đổi các ký tự thành chữ hoa
3. Server gửi dữ liệu đã viết hoa cho client
4. Client nhận dữ liệu đã viết hoa và hiển thị ra trên màn hình của nó

# Lập trình Socket với UDP

## UDP: không có "kết nối" giữa client và server

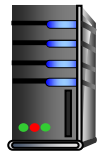
- Không bắt tay trước khi gửi dữ liệu
- Bên gửi chỉ định rõ ràng địa chỉ đích IP và cổng dịch vụ cho mỗi gói tin
- Bên nhận trích xuất địa chỉ IP và cổng dịch vụ của bên gửi từ gói tin đã nhận

## UDP: dữ liệu được truyền có thể bị mất hoặc nhận không đúng thứ tự

## Góc nhìn ứng dụng:

- UDP cung cấp khả năng truyền các nhóm byte ("datagram") *không tin cậy* giữa các tiến trình client và server

# Tương tác giữa Socket Client và Server: UDP



server (running on serverIP)

Tạo socket, cổng=x:
serverSocket =
socket(AF_INET,SOCK_DGRAM)

Đọc datagram từ
serverSocket

Phản hồi tới
serverSocket
chỉ định rõ địa chỉ
IP và cổng dịch vụ
của client

client 

Tạo socket:
clientSocket =
socket(AF_INET,SOCK_DGRAM)

Tạo datagram với địa chỉ serverIP
và cổng=x; Gửi datagram qua
clientSocket

Đọc datagram từ
clientSocket

Ngắt kết nối
clientSocket

# Ví dụ: UDP client

*Python UDPClient*

include Python's socket library →

```
from socket import *
serverIP = "127.0.0.10"
serverPort = 12000
```

create UDP socket for server →

```
clientSocket = socket(AF_INET, SOCK_DGRAM)
```

raw string →

```
message = "Number one"
print("Client send:", message)
```

attach server name, port to message; send into socket →

```
clientSocket.sendto(message.encode(), (serverI
```

read reply characters from socket into string →

```
modifiedMessage, serverAddress = clientSocket.
```

print out received string and close socket →

```
print("Client receive:", modifiedMessage.decod
clientSocket.close()
```

# Ví dụ: UDP server

*Python UDPServer*

```
from socket import *
serverIP = "127.0.0.10"
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind((serverIP, serverPort))
print ("The server is ready to receive")
while True:
    message, clientAddress = serverSocket.recvfrom
    print("Server receive:", message.decode())

    capitalizedSentence = message.decode().upper()
    serverSocket.sendto(capitalizedSentence.encode
```

create UDP socket →

bind socket to local port number 12000 →

loop forever →

Read from UDP socket into message, getting client's address (client IP and port) →

send upper case string back to this client →

# Lập trình Socket với TCP

## Client phải tạo kết nối với server

- Tiến trình server phải được chạy trước
- Server phải tạo socket (cánh cửa) để chào đón kết nối từ client

## Client kết nối với server bằng cách:

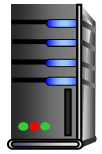
- Tạo socket TCP, chỉ định địa chỉ IP và cổng dịch vụ của tiến trình server
- *Khi client tạo socket:* client thiết lập kết nối tới socket TCP của server

- Khi được kết nối bởi client, *server tạo socket TCP mới* cho tiến trình server để giao tiếp client cụ thể
  - Cho phép server tương tác với nhiều clients
  - Cổng dịch vụ nguồn được sử dụng để phân biệt các client khác nhau (chi tiết ở bài học sau)

**Góc nhìn ứng dụng**

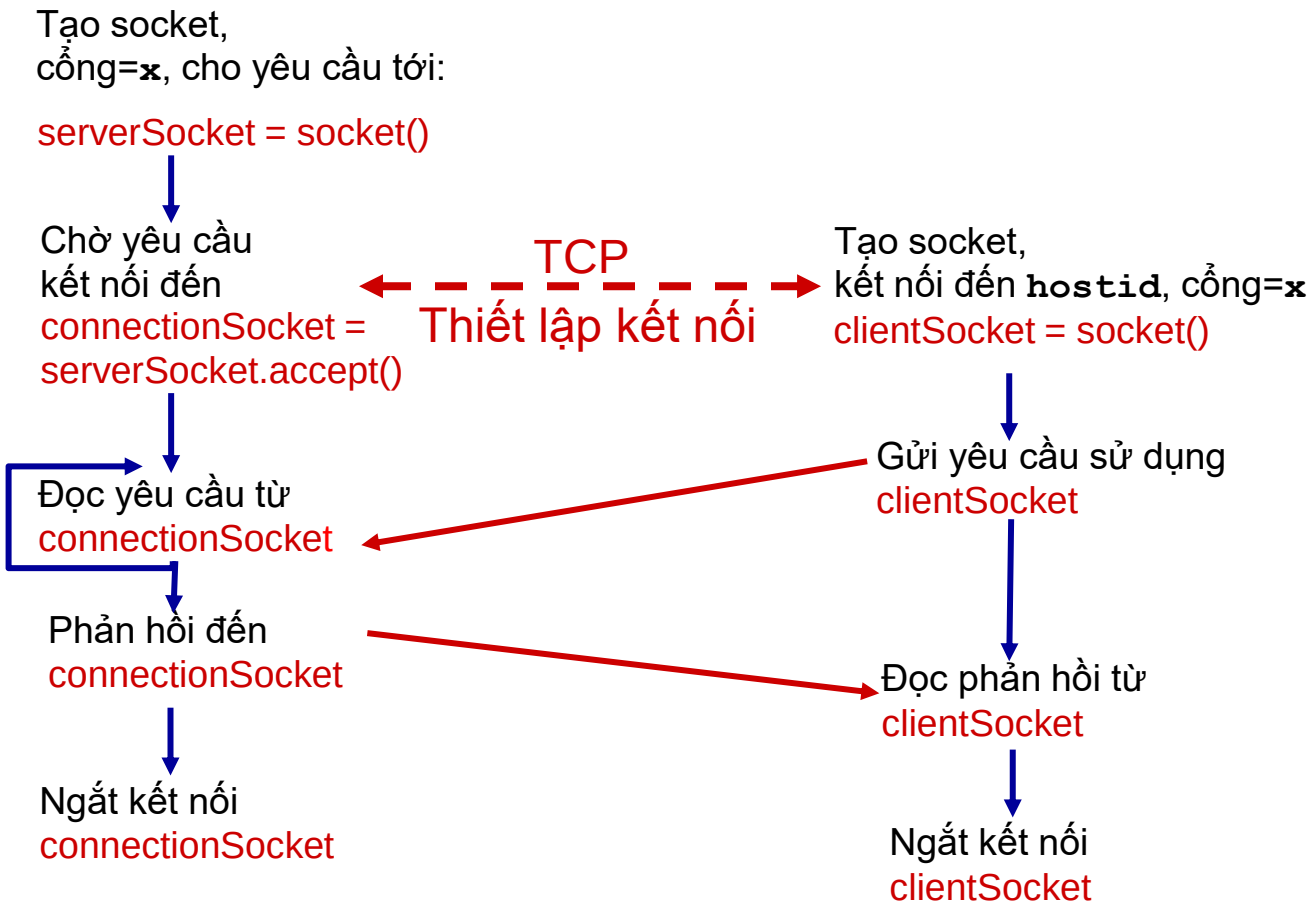
TCP cung cấp truyền dữ liệu tin cậy, theo thứ tự ("theo đường ống") giữa tiến trình của client và server

# Tương tác giữa Socket Client và Server: TCP



server (running on hostid)

client 



# Ví dụ: TCP client

*Python TCPClient*

```
from socket import *
serverIP = "127.0.0.11"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverIP,serverPort))
message = "Number one"
print("Client send:", message)

clientSocket.send(message.encode())
modifiedMessage = clientSocket.recv(1024)
print("Client receive:", modifiedMessage.de
clientSocket.close()
```

create TCP socket for server, remote port 12000

No need to attach server name, port

# Ví dụ: TCP server

*Python TCPServer*

```
from socket import *
serverIP = "127.0.0.11"
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((serverIP,serverPort))
serverSocket.listen(1)
print("The server is ready to receive")
while True:
  connectionSocket, addr = serverSocket.accep

  sentence = connectionSocket.recv(1024).deco
  print("Server receive:", sentence)
  capitalizedSentence = sentence.upper()
  connectionSocket.send(capitalizedSentence.e
  connectionSocket.close()
```

create TCP welcoming socket →

server begins listening for incoming TCP requests →

loop forever →

server waits on accept() for incoming requests, new socket created on return →

read bytes from socket (but not address as in UDP) →

close connection to this client (but *not* welcoming socket) →

# Bài 1 và 2: Tổng kết

- Góc nhìn của lập trình viên về Internet
- Mô hình Client-Server
- Socket
- Lập trình Socket với UDP và TCP

# Bài tập về nhà

- Viết chương trình Server và Client chạy trên localhost sử dụng **UDP** và **TCP** để truyền dữ liệu cho nhau. Dữ liệu là văn bản chứa trong tập tin content.txt với nội dung: “Lập trình mạng”. Client gửi văn bản đến Server và yêu cầu Server viết hoa văn bản đó và gửi lại cho Client.
- Viết chương trình Server và Client chạy trên localhost sử dụng **UDP** và **TCP** để truyền dữ liệu cho nhau. Dữ liệu là hình ảnh với tên image.png. Gợi ý: có thể tạo hình ảnh bằng phần mềm Paint. Kích thước hình ảnh nhỏ hơn 4096 bytes. Máy khách gửi hình ảnh đến Máy chủ, sau đó Máy chủ lưu hình ảnh vào bộ nhớ cục bộ của nó.

# Thank you