

## **Chương 4. Lớp Vận tải (Transport layer)**

### **Mục tiêu**

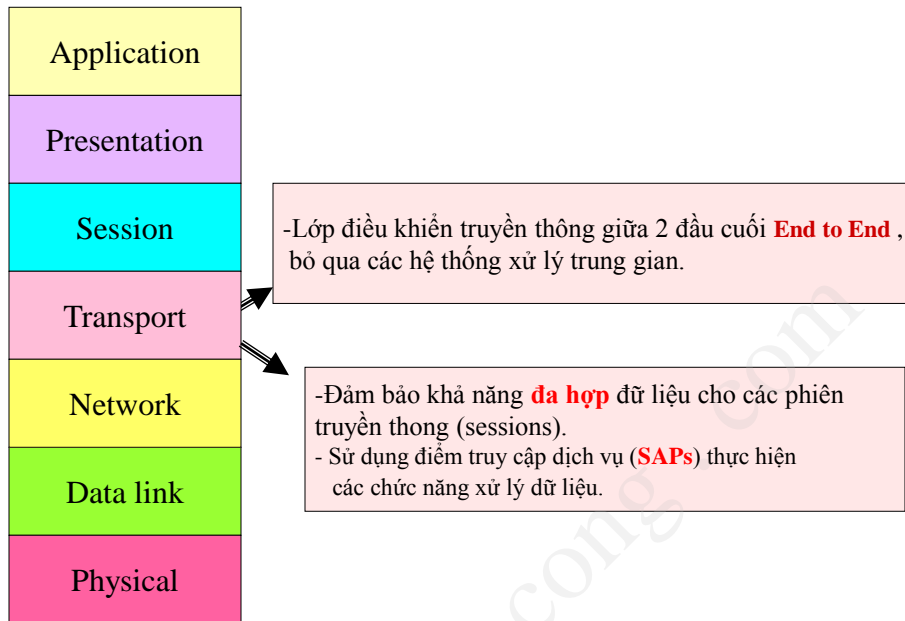
- ❖ Nhận diện được
  - Đặc điểm, chức năng điều khiển của lớp vận tải
  - Các chức năng được triển khai trong giao thức TCP
  - Các chức năng được triển khai trong giao thức UDP
  - Cơ chế ghép phiên truyền thông qua tham số SAPs trong mô hình OSI và qua port number trong mô hình TCP/IP
- Nắm rõ các tính năng điều khiển trong truyền có kết nối của TCP
- ❖ Nắm rõ các tính năng điều khiển trong truyền không kết nối của UDP
- ❖ Cơ chế đa hợp trong TCP; UDP qua SOCKET

## **Chương 4. Lớp Vận tải (Transport layer)**

### **Nội dung**

1. Đặc điểm và nhiệm vụ và các chức năng điều khiển
  - Kiểu kết nối
  - Đảm bảo độ tin cậy
  - Chức năng đa hợp với SAPs
2. **TCP**
  - Three way handshake
  - Điều khiển luồng & Điều khiển lỗi
  - Đa hợp với port number
3. **UDP**
  - Phát hiện lỗi
  - Đa hợp với port number

## Lớp **Transport** trong mô hình OSI



## Đặc điểm và chức năng điều khiển

### ❖ Đặc điểm:

- ❑ Điều khiển truyền thông giữa 2 đầu cuối, bỏ qua các hệ thống xử lý trung gian (nếu có)

📶 End to End

### ❖ Chức năng điều khiển truyền:

- ❑ Độ tin cậy trong quá trình truyền tải các gói dữ liệu giữa các ứng dụng khác nhau thông các hệ thống truyền thông cùng kiến trúc phân lớp.
- ❑ Khả năng đa hợp các phiên truyền trong cùng hệ thống

## Điều khiển kết nối

- ❖ Các chức năng điều khiển kết nối:
  - Kiểu truyền có kết nối:
    - Thiết lập kết nối với thông tin nhận diện kết nối
    - Truyền gói tuần tự.
    - Nhận diện số tuần tự gói truyền đầu tiên trong giai đoạn trao đổi dữ liệu.
    - Nhận diện số tuần tự gói truyền đầu tiên và cuối cùng trước khi kết thúc phiên truyền.
    - Giao thức liên quan trong mô hình TCP/IP: TCP
  - Kiểu truyền không kết nối không đảm bảo độ tin cậy:
    - Giao thức liên quan trong mô hình TCP/IP: UDP

## Điều khiển về độ tin cậy (Reliability)

- ❖ Trong truyền có kết nối, luồng gói được truyền tuần tự cho phép điều khiển luồng và điều khiển lỗi: đảm bảo độ tin cậy (TCP)
  - ☐ Điều khiển luồng
  - ☐ Điều khiển lỗi
- ❖ Ngược lại, truyền không thiết lập kết nối, luồng các gói được truyền tuần tự cho phép điều khiển luồng và điều khiển lỗi: đảm bảo độ tin cậy.

## Điều khiển về đa hợp phiên truyền (Multiplexing)

### ❖ Công tiếp nhận phiên truyền:

- ❑ SAPs (service access points) trong mô hình OSI
- ❑ Port number/ TCP/IP model
- ❑ minh họa về thông tin mô tả phiên truyền:

✚Src- session-ID

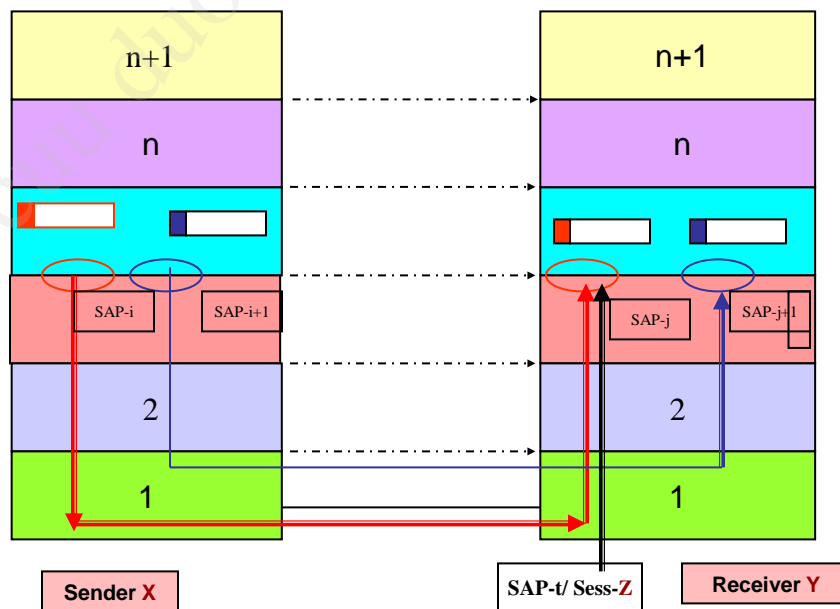
✚Src-portNumber

✚Dest- session-ID

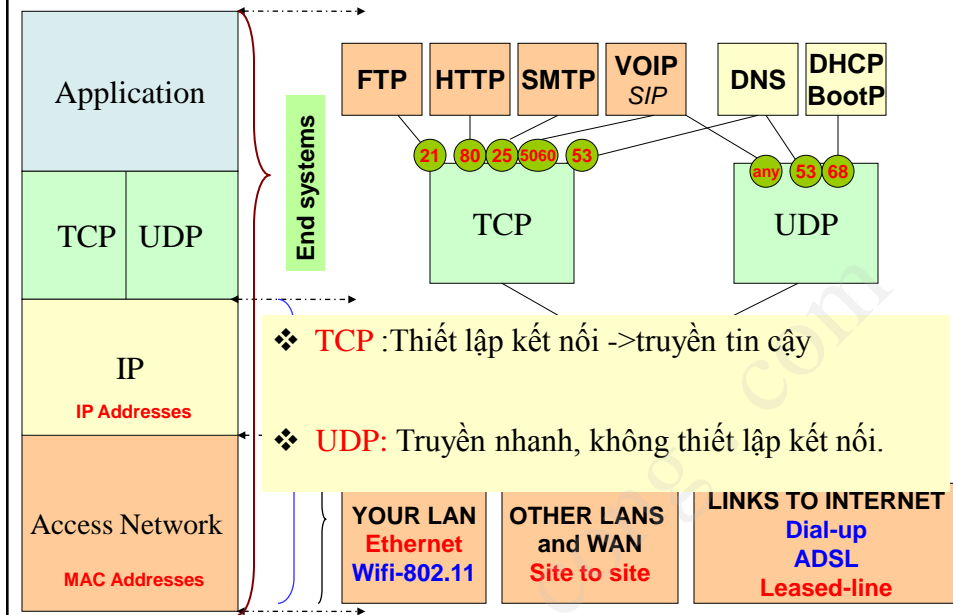
✚Dest-portNumber

### ❖ Ví dụ: dịch vụ web mở cổng 80 để nhận các phiên truy cập từ các web-clients (web browser)

## Điểm truy cập dịch vụ- SAPs



## Kiến trúc và bộ giao thức TCP/IP



## Giao thức TCP và UDP

- ❖ Đặc điểm và tiêu chí thiết kế của TCP và UDP
- ❖ Hoạt động đặc trưng của TCP
  - ☐ Cơ chế bắt tay 3 bước (Three way handshake)
  - ☐ Cơ chế trao đổi dữ liệu tin cậy
    - Điều khiển luồng
    - Điều khiển lỗi
  - Ứng dụng trong dịch vụ truyền thông
- ❖ Hoạt động đặc trưng của UDP
  - Ứng dụng trong dịch vụ truyền thông
- ❖ Đa hợp với Port number/Socket.

## Đặc điểm và tiêu chí thiết kế

### ❖ Đặc điểm:

- ❑ TCP và UDP là giao thức được thiết kế phục vụ lớp Transport để điều khiển truyền thông giữa 2 đầu cuối.

📡 End to End

### ❖ Tiêu chí thiết kế:

- ❑ Cho phép ghép các phiên truyền thông qua các cổng ứng dụng (ports).
- ❑ TCP: đảm bảo độ tin cậy cao nhất cho luồng các đoạn dữ liệu (segments)
- ❑ UDP đảm bảo độ trễ thấp nhất và hoạt động điều khiển truyền đơn giản nhất.

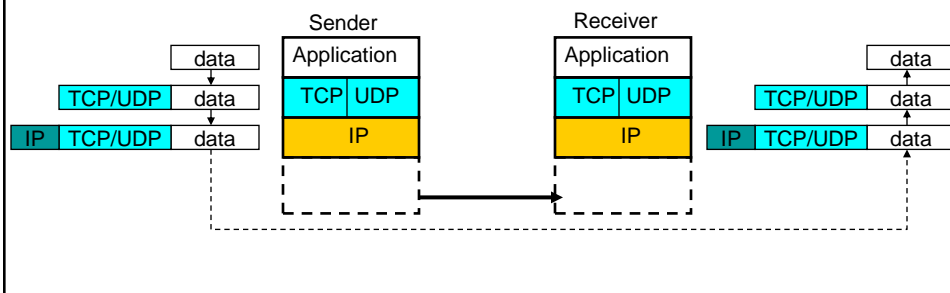
## TCP, UDP và lớp kế dưới IP

### ❖ IP: hoạt động không kết nối-> dễ bị mất gói

- ❑ IP routing, fragmentation, error detection...

### ❖ UDP: điều khiển truyền End to End và đơn giản → used for multiplexing/demultiplexing, error detection

### ❖ TCP: đảm bảo độ tin cậy cuối cùng trong điều khiển truyền thông → multiplexing/demultiplexing, flow and congestion control



## Transmission Control Protocol (TCP) (1/2)

### TCP

- ❖ Phải thiết lập kết nối trước khi truyền luồng Pkts tuần tự.
- ❖ Phân phát các Pkts bảo đảm độ tin cậy
- ❖ Phân phát các Pkts theo luồng theo thứ tự Pkts
- ❖ Chỉ định mỗi socket cho một kết nối

### Telephone Call

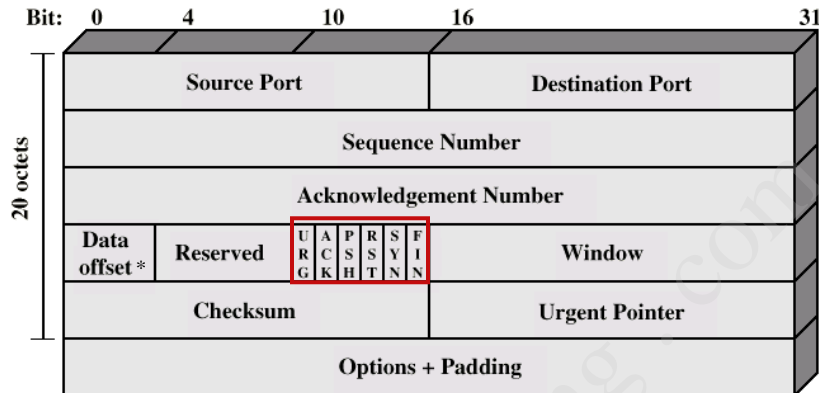
- ❖ Phải quay số để thiết lập cuộc gọi trước khi đàm thoại.
- ❖ Trao đổi đàm thoại bảo đảm độ tin cậy
- ❖ Trao đổi đàm thoại bảo đảm câu trước, câu sau
- ❖ Chỉ định mỗi kênh truyền cho một cuộc đàm thoại

## Transmission Control Protocol (TCP) (2/2)

- ❖ TCP cung cấp cơ chế điều khiển truyền thông tin cậy giữa các ứng dụng người.
- ❖ Các đặc điểm:
  - ☐ Connection-oriented và end-to-end với Three-Way Handshake
  - ☐ Điều khiển luồng bằng cửa sổ trượt “sliding windows” và sử dụng số tuần tự “sequence numbers” và “acknowledgments”,
  - ☐ Cơ chế điều khiển lỗi với ARQ- Go back N và cơ chế phục hồi lỗi sử dụng **Time-out** (Error Recovery)
  - ☐ Cung cấp đa hợp phiên cho mỗi loại ứng dụng thông qua port (port number)

## Định dạng đoạn dữ liệu TCP

Ethernet Hdr - 20 bytes (little-endian)	IP Header - 20 bytes (big-endian)	TCP Header - 20 bytes (big-endian)	App. Hdr & Data
--	--------------------------------------	---------------------------------------	--------------------



\* Length of TCP Header in bytes / 4 TCP Flags: U A P R S F

15

## Three-way handshake Establishment

- ❖ Thông tin điều khiển kết nối:
  - ☐ Sử dụng gói thiết lập kết nối SYN chứa số thứ tự (Seq)
    - ➡ Seq: số thứ tự của đoạn dữ liệu đầu tiên được gửi khi bắt đầu giai đoạn trao đổi dữ liệu (Data transfer)
    - ▶ Out-First-SN; In-First-SN;
  - ☐ Sử dụng xác nhận ACK cùng số thứ tự (Seq) vừa nhận được.
- ❖ Thông tin điều khiển sử dụng trong giai đoạn Data transfer:
  - ☐ WS: windows size
  - ☐ MSS: Maximum segment size
- ❖ Block of memory-> new connection:
  - ☐ Chứa các thông số điều khiển
  - ☐ Chứa các đoạn dữ liệu truyền và nhận



## Three-way handshake

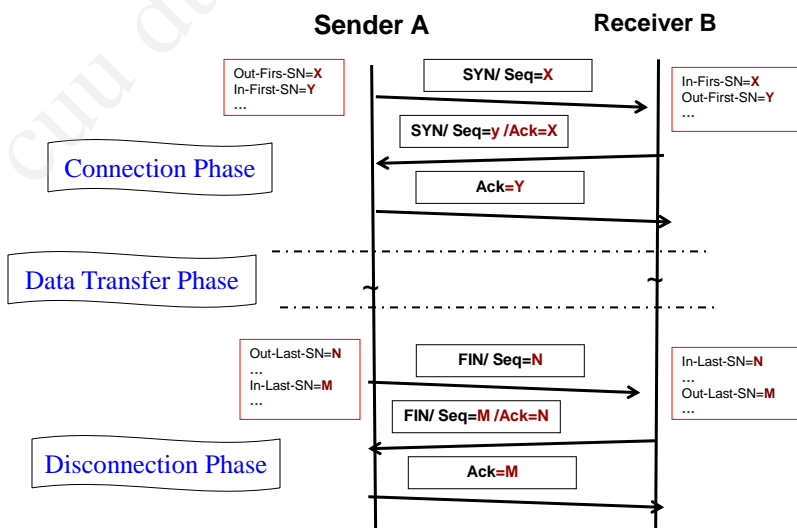
### Release

❖ Thông tin điều khiển kết nối:

- ☐ Sử dụng FIN với số thứ tự (Seq)
- ☐ Sử dụng xác nhận ACK cùng số thứ tự (Seq) vừa nhận được.
- ☐ Out-Last-SN: số thứ tự đoạn dữ liệu cuối cùng vừa được gửi đi trong giai đoạn trao đổi dữ liệu (Data transfer) của đầu cuối A
- ☐ In-Last-SN: số thứ tự đoạn dữ liệu cuối cùng vừa được gửi đi trong giai đoạn trao đổi dữ liệu (Data transfer) của đầu cuối B

## Điều khiển kết nối TCP

### Three-way handshake



## Điều khiển trao đổi dữ liệu

### Điều khiển luồng

- ❖ Số thứ tự (sequence number) được chỉ định cho **thứ tự byte** dữ liệu truyền (không phải thứ tự gói truyền)
  - ☐ Cơ chế điều khiển (**Sliding window**)
  - ☐ **Kích thước** cửa sổ truyền có thể thay đổi (**Dynamically window size**)
  - ☐ Số thứ tự byte gửi đi: **Sequence number**
  - ☐ Số thứ tự xác nhận khả năng tiếp nhận dữ liệu: **Acknowledge number**
  - ☐ Khả năng phát gói: tùy thuộc vào giá trị (có thể thay đổi) của **Window size** bên đầu nhận
    - ✚ Bufer phát: số gói được đưa vào để phát tuần tự  $N_s \rightarrow N_{s_{Max}}$
    - ✚  $N_{s_{Max}} = WS + ACK - num - 1$

### Điều khiển lỗi (ARQ- Go back N)

- ❖ Sử dụng cơ chế **ARQ- Goback N**:
  - ☐ Bên thu phát hiện 1 trong các trường hợp sau:
    - ✚ phát hiện mất gói tại đầu thu (so sánh  $V_r$  và  $N_s$  hay SN/TCP)
    - ✚ phát hiện sai checksum
  - ☐ Bên phát sẽ truyền lại kể từ gói có thứ tự được chỉ rõ **N**
- ❖ Sử dụng cơ chế tự động truyền tại **Time-out Recovery**
  - ☐ TCP của bên đầu phát sẽ duy trì đồng hồ định giờ (Timer) cho mỗi kết nối.
    - ✚ Thông số Time-out: thời gian chờ lớn nhất để nhận thông tin xác nhận từ đầu thu (RCV)
  - ☐ Hết thời gian chờ Time-out, bên phát sẽ **tự động truyền lại theo cơ chế điều khiển ARQ** được sử dụng.

## Automatic Retransmission/ Time-out (RTO)

### ❑ Giá trị Time-out (**Retransmission Timeout (RTO) value.**)

- ✚ Nếu quá lớn : có khả năng ảnh hưởng đến độ trễ truyền thông
- ✚ Nếu quá nhỏ: có khả năng ảnh hưởng đến hiệu suất truyền thông
- ✚ Lệ thuộc độ trễ của mạng truyền thông (RTT: Round Trip Time)
- ✚ Thuật toán **Karn**

### ❑ Hoạt động của Timer:

- ✚ Khởi động khi truyền xong đoạn dữ liệu cuối cùng trong buffer phát.
- ✚ Ngưng hoạt động mỗi khi nhận được một xác nhận mới từ bên đầu thu (RCV)

## TCP và Ứng dụng trong dịch vụ truyền thông

### ❖ Các ứng dụng cho người dùng:

#### ❑ Truyền một lượng lớn dữ liệu (**bulk data transfer**)

- ✚ FTP, Mail, Web..

#### ❑ Trao đổi dữ liệu có tính tương tác (**interactive data transfer**):

- ✚ Chat, telnet, rlogin...

### ❖ Các ứng dụng cho hạ tầng mạng:

#### ❑ DNS: (**Zone Transfer**) trao đổi thông tin tên miền và địa chỉ IP giữa các máy chủ tên miền (Name server)

#### ❑ DCE/RPC: quản trị trong môi trường tính toán phân tán của Windows OS.

## User Datagram Protocol (UDP) (1/2)

### UDP

- ❖ Mở một socket để gửi Pkt
- ❖ Truyền không đảm bảo tin cậy.
- ❖ Pkts không gửi theo thứ tự
- ❖ Các Pkts được truyền thông không lệ thuộc vào nhau.
- ❖ Truyền thông phải chỉ định địa chỉ đích.

### Postal Mail

- ❖ Lập mailbox để gửi thư
- ❖ Gửi thư thông thường, không bảo đảm ☺
- ❖ Các lá thư gửi không theo thứ tự
- ❖ Các lá thư được gửi một cách độc lập.
- ❖ Phải chỉ định địa chỉ cho lá thư mỗi khi gửi đi.

## User Datagram Protocol (UDP) (2/2)

- ❖ UDP truyền dữ liệu không tin cậy giữa các host.
- ❖ Đặc tính:
  - ☐ Hoạt động End-to-End
  - ☐ Connectionless:
    - ✚ Không tin cậy, không kiểm tra thông điệp phân phát.
    - ✚ Không “acknowledgements”.
      - ▶ Không điều khiển luồng (no window)
      - ▶ Chỉ phát hiện lỗi mà không điều khiển lỗi
    - ✚ Cung cấp đa hợp phiên cho mỗi loại ứng dụng thông qua port (port number)
- ❖ Tùy chọn kiểm tra lỗi (checksum field)
- ❖ Không phân mảnh và tái hợp.

## UDP Header format



# of Bits	16	16	16	16	
	Source Port	Destination Port	Length	Check Sum	Data...

• Multiplexing by ports -- Error detection

❖ UDP là một giao thức datagram đơn giản:

- ☐ chức năng ghép kênh
- ☐ Chức năng kiểm tra lỗi.

## UDP và Ứng dụng trong dịch vụ truyền thông

❖ Các ứng dụng cho người dùng:

- ☐ VoIP
- ☐ Streaming Audio
- ☐ Gaming
- ☐ Video Conferencing

❖ Các ứng dụng cho hạ tầng mạng:

- ☐ SNMP-> quản trị mạng : tương tác giữa hệ thống quản trị và các hệ thống được quản trị.
- ☐ DNS: trao đổi các truy vấn tên miền giữa máy người dùng (DNS clients) và máy chủ tên meefi (Name server)
- ☐ DHCP: trao đổi các tương tác phục vụ việc cung cấp và cập nhật thông tin cấu hình mạng cho các host trong hệ thống mạng.

## Đa hợp trong TCP và UDP

### Port number và socket

#### Port number và socket :

❖ Port numbers được sử dụng để nhận dạng các phiên kết nối khác nhau diễn ra trên cùng một host.

❖ **Socket=Network address +protocol+ port number**

❖ Vd: <http://192.168.20.245:8080>

#### Dãy ports:

❖ 2 bytes: 0 – 65535.

- ☐ Numbers nhỏ hơn **255** : chỉ định cho dịch vụ phổ biến trên internet.
- ☐ Numbers từ **255 - 1023** : chỉ định cho dịch vụ ứng dụng đặc trưng của các tổ chức
- ☐ Numbers trên **1023** : chỉ định cho tiến trình trên máy client

## Ví dụ “Port number”

❖ Những giá trị port nguồn được gán tự động bởi host nguồn; thông thường có giá trị lớn hơn 1023.

- ☐ ứng dụng Web được gán port 80.
- ☐ ứng dụng Web client sử dụng port 32938

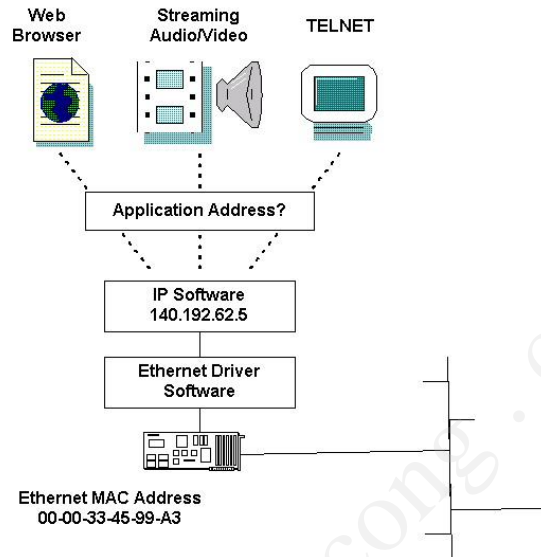
❖ TCP segment gọi từ client đến server có:

- ☐ source port : 32938
- ☐ destination port : 80

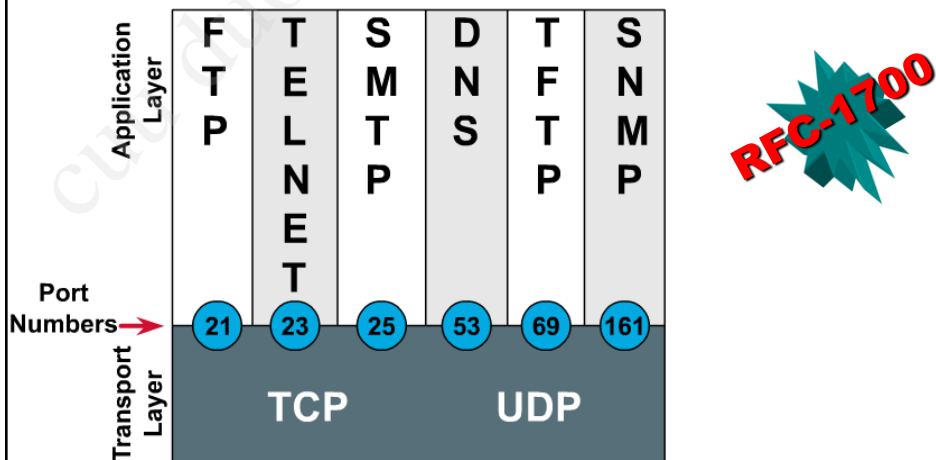
❖ Khi Web Server phản hồi, TCP có:

- ☐ source port : 80
- ☐ destination port : 32938

## Đa hợp cho mỗi ứng dụng

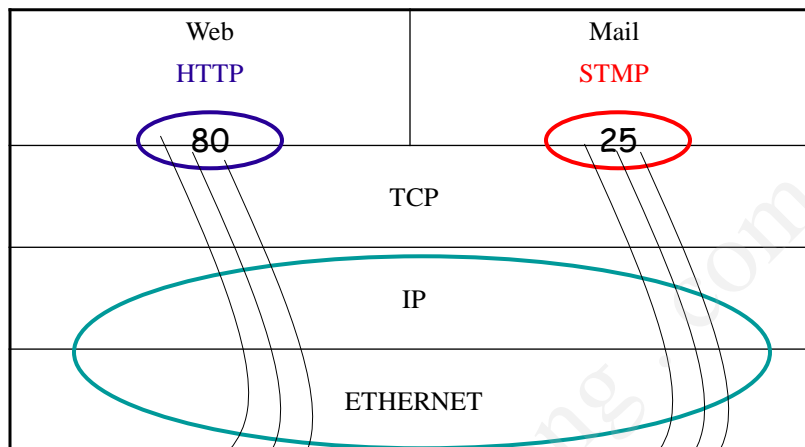


## TCP and UDP port numbers



Cả TCP và UDP đều sử dụng port (socket) để chuyển giao gói tin giữa 2 lớp kề nhau trên hệ thống.

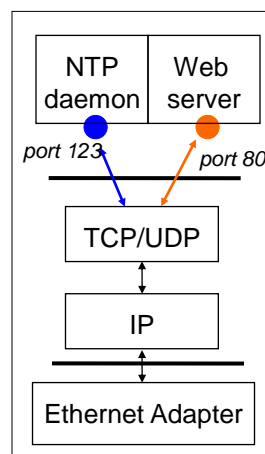
## Ghép kênh của các phiên bằng Port



The same of MAC address : IP address

## Port Numbers

- ❑ **Port numbers** dùng để nhận diện một thực thể truyền thông trên một host.
- ❑ Port numbers có thể là 1 trong 2 dạng
  - ✚ Chỉ định trước với giá trị dạng Well-known (port 0-1023)
  - ✚ Chỉ định động hay mang tính cục bộ (port 1024-65535)
- ❑ Các máy chủ quản lý dịch vụ ứng dụng thường sử dụng well-known ports để được biết rộng rãi bởi các clients
  - ✚ Any client can identify the server/service
  - ✚ HTTP = 80, FTP = 21, Telnet = 23, ...
  - ✚ /etc/service defines well-known ports
- ❑ Clients thường sử dụng ports cấp động
  - ✚ Chỉ định bởi kernel





## SOCKET

1. Ứng dụng và mô hình Client-Server
2. Khái niệm về socket
3. Cấu trúc socket
4. Sock-Stream và các hàm chức năng
5. Sock-Dgram và các hàm chức năng

### Mô hình Client-Server và ứng dụng

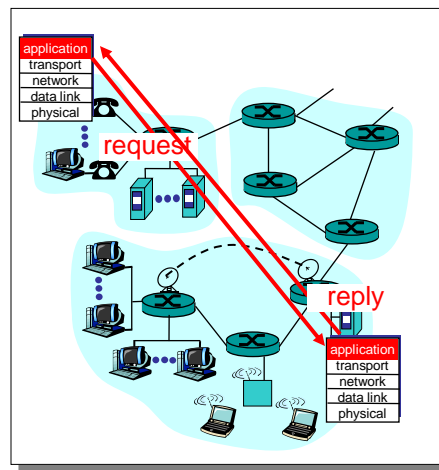
- ❖ Mô hình Client-Server là ứng dụng truyền thông qua mạng với vai trò xác định giữa 2 máy: **client** and **server**

#### **Client:**

- ❖ Khởi động một yêu cầu đến server: Request

#### **Server:**

- ❖ Cung cấp dịch vụ được yêu cầu đến Client bởi **Reply**



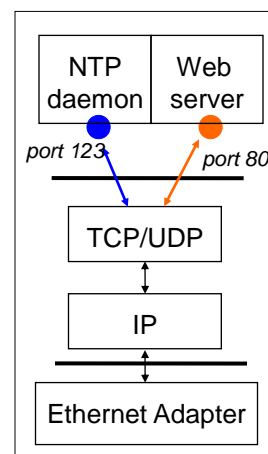
## Các yêu cầu của ứng dụng đối với mạng truyền thông

1. **Data loss** -> VoIP vs File transfer
2. **Timing** -> Internet telephony, interactive games
3. **Bandwidth** -> multimedia

Application	Data loss	Bandwidth	Time Sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
web documents	no loss	elastic	no
real-time audio/ video	loss-tolerant	audio: 5Kb-1Mb video: 10Kb-5Mb	yes, 100's msec
stored audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	few Kbps	yes, 100's msec
financial apps	no loss	elastic	yes and no

## Port Numbers

- ❑ **Port numbers** dùng để nhận diện một thực thể truyền thông trên một host.
- ❑ Port numbers có thể là 1 trong 2 dạng
  - + Chỉ định trước với giá trị dạng Well-known (port 0-1023)
  - + Chỉ định động hay mang tính cục bộ (port 1024-65535)
- ❑ Các máy chủ quản lý dịch vụ ứng dụng thường sử dụng well-known ports để được biết rộng rãi bởi các clients
  - + Any client can identify the server/service
  - + HTTP = 80, FTP = 21, Telnet = 23, ...
  - + `/etc/service` defines well-known ports
- ❑ Clients thường sử dụng ports cấp động
  - + Chỉ định bởi kernel



## Định nghĩa SOCKET

❑ SOCKET là giao diện giữa ứng dụng và mạng, được sử dụng để nhận diện điểm đầu cuối của một tiến trình truyền thông. SOCKET có thể mô tả:

- ✚ Loại truyền thông

  - ✚ reliable vs. best effort

  - ✚ connection-oriented vs. connectionless

- ✚ Họ địa chỉ mạng (AF: Addressing Family)

❑ SOCKET được tạo bởi một ứng dụng cụ thể

❑ SOCKET **sau khi được khởi tạo** có thể:

- ✚ Chuyển dữ liệu đến socket để truyền thông trên mạng -> sendto()/UDP hay write()/TCP

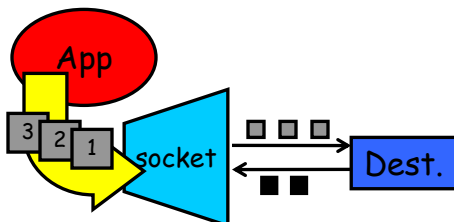
- ✚ Nhận dữ liệu từ socket sau khi đã tiếp nhận từ mạng vào.-> recvfrom()/UDP hay read()/TCP

## Phân loại SOCKET

### ❖ SOCK\_STREAM

❑ TCP

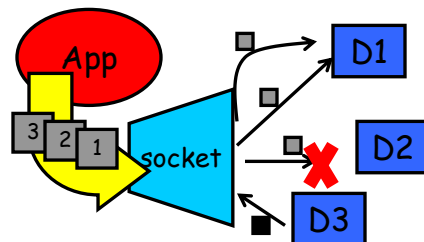
- ✚ Bidirectional



### ❖ SOCK\_DGRAM

❑ UDP

- ✚ Uni-directional (sending + Receiving)



38

## Cấu trúc địa chỉ một SOCKET

### ❖ Cấu trúc chung:

```
struct sockaddr {  
    u_short sa_family;  
    char sa_data[14];  
};
```

✚ **sa\_family** : socket address family

- chỉ ra họ địa chỉ được sử dụng (họ Internet hay họ apple talk...)
- học địa chỉ sẽ quyết định việc sử dụng 14 Bytes chuỗi dữ liệu còn lại.

### ❖ Cấu trúc địa chỉ Internet :

```
struct sockaddr_in {  
    short sin_family;  
    u_short sin_port;  
    struct in_addr sin_addr;  
    char sin_zero[8];  
};
```

✚ **sin\_family** = AF\_INET

✚ **sin\_port**: port # (0-65535)

✚ **sin\_addr**: IP-address

✚ **sin\_zero**: unused

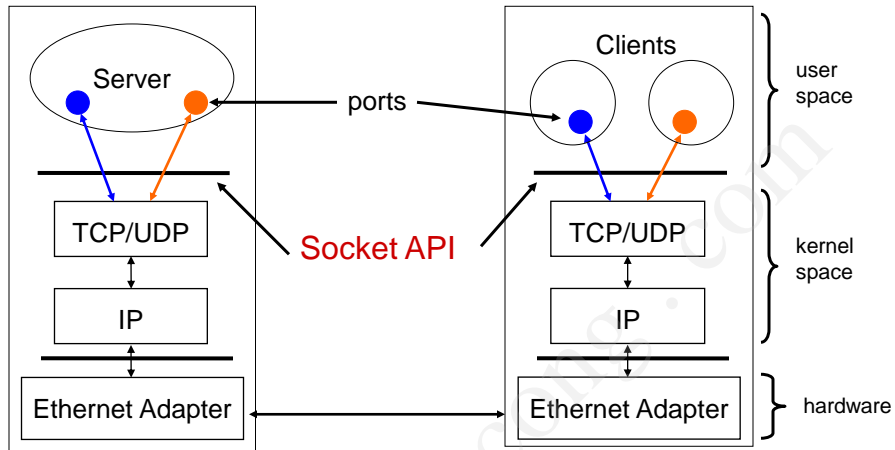
## Minh họa cấu trúc địa chỉ một SOCKET họ Internet

```
#include <netinet/in.h>  
  
/* Internet address structure */  
struct in_addr {  
    u_long s_addr;          /* 32-bit IPv4 address */  
};                          /* network byte ordered */  
  
/* Socket address, Internet style. */  
struct sockaddr_in {  
    u_char sin_family;      /* Address Family */  
    u_short sin_port;       /* UDP or TCP Port# */  
                          /* network byte ordered */  
    struct in_addr sin_addr; /* Internet Address */  
    char sin_zero[8];       /* unused */  
};
```

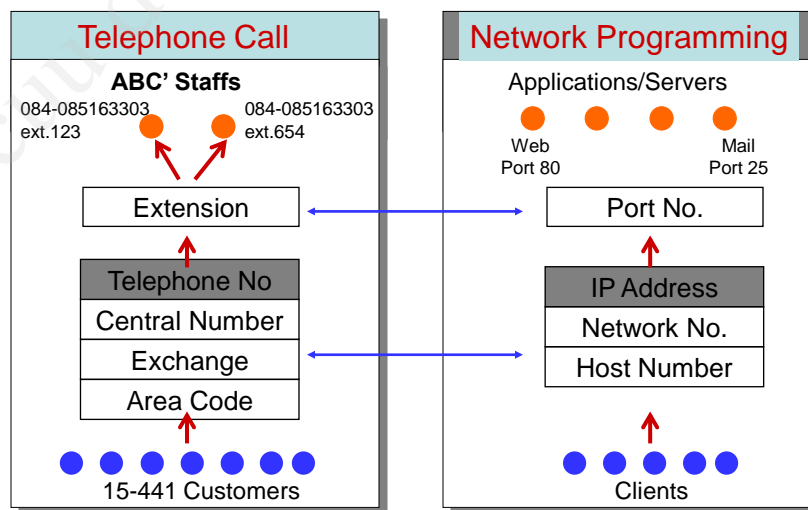
❖ **sin\_family** = AF\_INET selects Internet address family

## SOCKET giao tiếp giữa Server và Client

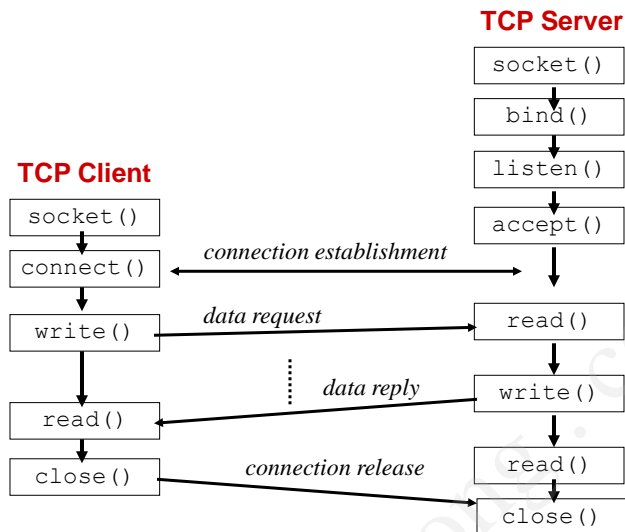
Server and Client exchange messages over the network through a common **Socket API**



## Mô hình tương tự



## Lưu đồ tương tác với TCP SOCKET



from UNIX Network Programming Volume 1, figure 4.1

## Hoạt động phía Client sử dụng SOCK-STREAM

### ❖ Giai đoạn khởi động và yêu cầu kết nối:

- ❑ socket – tạo socket loại SOCK-STREAM
- ❑ gethostbyname – xác định server
- ❑ connect – yêu cầu kết nối đến server

### ❖ Trao đổi dữ liệu giữa server và client :

- ❑ recv – nhận dữ liệu từ server
- ❑ send – gửi dữ liệu đến server

### ❖ Giải phóng kết nối :

- ❑ close – giải phóng kết nối

## Hoạt động phía **Server** sử dụng **SOCK-STREAM**

### ❖ **Giai đoạn khởi động:**

- ☐ **socket** – tạo socket với loại SOCK-STREAM.
- ☐ **bind** – liên kết socket đã tạo với một địa chỉ socket cụ thể.
- ☐ **listen** – bắt đầu sẵn sàng tiếp nhận các yêu cầu kết nối đến từ client

### ❖ **Thiết lập truyền (session) có kết nối và trao đổi dữ liệu :**

- ☐ **accept** – tiếp nhận và thiết lập socket mới cho kết nối từ một yêu cầu của client đến socket chờ tại listen()
- ☐ **recv** – nhận dữ liệu từ client
- ☐ **send** – gửi dữ liệu đến client

### ❖ **Giải phóng kết nối:**

- ☐ **close** – giải phóng kết nối bằng việc đóng lại socket đang sử dụng.

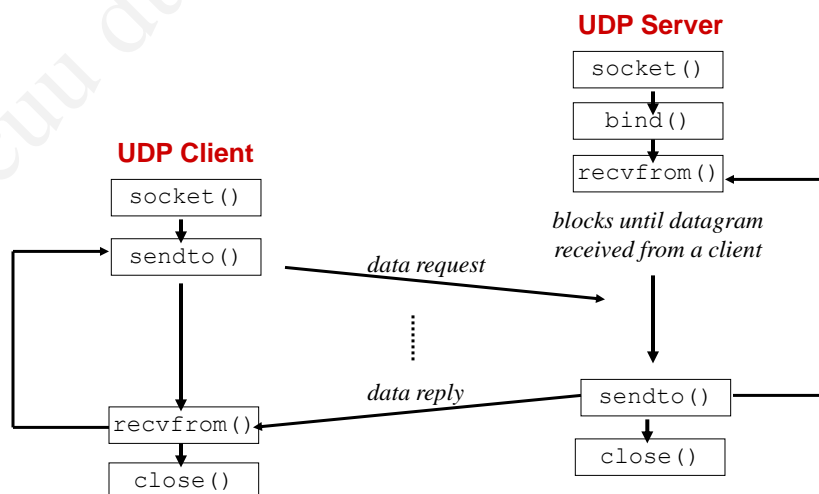
## Các hàm chức năng cho Stream socket (TCP) 1/2

STT	Hàm chức năng và tham số liên quan	Chức năng hoạt động
1	<b>Socket</b> (Int family, Int type, Int protocol)	<ul style="list-style-type: none"><li>• Tạo một Sock-Stream.</li><li>• Kết quả trả về số nhận diện tương ứng với socket được chỉ định</li><li>• Lỗi xảy ra nếu kết quả trả về là -1</li></ul>
2	<b>Bind</b> (Int sockfd, Const struct sockaddr_in *localaddr, Int localaddrlen)	<ul style="list-style-type: none"><li>• Đăng kí với hệ thống socket đã khởi tạo với địa chỉ socket cục bộ.</li><li>• Kết quả thành công nếu là 0, và ngược lại là -1</li></ul>
3	<b>Listen</b> (Int sockfd, Int backlog)	<ul style="list-style-type: none"><li>• Lập trạng thái sẵn sàng tiếp nhận yêu cầu kết nối từ clients</li><li>• Kết quả thành công nếu là 0, và ngược lại là -1</li></ul>
4	<b>Accept</b> (Int sockfd, Const struct sockaddr_in *clientaddr, Int clientaddrlen)	<ul style="list-style-type: none"><li>• Chấp nhận kết nối từ client và tạo một socket mới.</li><li>• Kết quả trả về là số hiệu của socket mới.</li></ul>

## Các hàm chức năng cho Stream socket (TCP) 2/2

5	<b>Connect</b> (Int sockfd, Const struct sockaddr_in *serveraddr, Int serveraddrlen)	<ul style="list-style-type: none"> <li>Được sử dụng bên phía client để gửi yêu cầu kết nối tới Server.</li> <li>Kết quả thành công nếu là 0, và ngược lại là -1</li> </ul>
6	<b>Read</b> (Int sockfd, Const void *buf, Int len)	<ul style="list-style-type: none"> <li>Tiếp nhận dữ liệu từ socket tương ứng với kết nối và bộ nhớ thu (Recv-Buffer)</li> <li>Trả về số byte đọc được nếu thành công, trả về 0 nếu không có dữ liệu.</li> <li>Trả về -1 nếu thất bại.</li> </ul>
7	<b>Write</b> (Int sockfd, Const void *buf, Int len)	<ul style="list-style-type: none"> <li>Gửi dữ liệu từ bộ nhớ phát (Sndr-Buffer) vào socket tương ứng với kết nối.</li> <li>Trả về số byte ghi được nếu thành công</li> <li>Trả về -1 nếu thất bại.</li> </ul>
8	<b>Close</b> ()	<ul style="list-style-type: none"> <li>Giải phóng socket vừa sử dụng cho kết nối.</li> </ul>

## Lưu đồ tương tác với UDP SOCKET



from UNIX Network Programming Volume 1, figure 8.1



## Hoạt động phía **Client** sử dụng **SOCK-DGRAM**

### ❖ **Giai đoạn khởi động SOCKET:**

- ☐ **socket** – tạo socket
- ☐ **gethostbyname** – xác định server

### ❖ **Thiết lập phiên truyền (session) **không** kết nối và trao đổi dữ liệu:**

- ☐ **recv** – nhận dữ liệu từ server
- ☐ **send** – gửi dữ liệu đến server

### ❖ **Giải phóng phiên truyền (session) không kết nối :**

- ☐ **close** – hủy socket

## Hoạt động phía **Server** sử dụng **SOCK-DGRAM**

### ❖ **Giai đoạn khởi động:**

- ☐ **socket** – tạo socket với loại SOCK-DGRAM.
- ☐ **bind** – liên kết socket đã tạo với một địa chỉ socket cụ thể.

### ❖ **Thiết lập phiên truyền (session) **không** kết nối và trao đổi dữ liệu:**

- ☐ **recv** – chỉ ra địa chỉ socket từ xa từ đó tiếp nhận dữ liệu từ client
- ☐ **send** – chỉ ra địa chỉ socket từ xa từ đó gửi dữ liệu đến client

### ❖ **Giải phóng phiên truyền (session) không kết nối tương ứng:**

- ☐ **close** – hủy socket

## Các hàm chức năng cho datagram socket (UDP)

Hàm chức năng và tham biến liên quan	Chức năng hoạt động
Socket(Int family, Int type, Int protocol)	<ul style="list-style-type: none"> <li>Tạo một Sock-Dgram</li> <li>Trả lại một số hiệu nhận diện socket được cấp phát</li> <li>Kết quả trả về là -1 khi có lỗi xảy ra</li> </ul>
Bind(Int sockfd, Const struct sockaddr_in *localaddr, Int localaddrlen, Int port)	<ul style="list-style-type: none"> <li>Đăng kí với hệ thống socket đã khởi tạo với địa chỉ socket local. Trả về 0 nếu thành công, -1 nếu thất bại</li> </ul>
Sendto(Int sockfd, Const void *buf, Int len, Int flags, Const struct sockaddr_in *toaddr, Int toaddrlen)	<ul style="list-style-type: none"> <li>Gửi dữ liệu đến một địa chỉ socket từ xa.</li> <li>Trả về số byte gửi được nếu thành công,</li> <li>Trả về -1 nếu thất bại.</li> </ul>
Recvfrom(Int sockfd, Const void *buf, Int len, Int flags, Const struct sockaddr_in *fromaddr, Int fromaddrlen)	<ul style="list-style-type: none"> <li>Nhận dữ liệu từ một địa chỉ socket từ xa.</li> <li>Trả về số byte nhận được nếu thành công,</li> <li>Trả về -1 nếu thất bại</li> </ul>

## Ý nghĩa các tham biến được sử dụng trong các hàm chức năng (1/2)

- ❖ Family: họ địa chỉ sử dụng ở lớp mạng (họ Internet hay họ IPX/SPX...)
- ❖ Type: kiểu socket (Sock-stream hay Sock-Dgram)
- ❖ Protocol: giao thức sử dụng truyền tải (thường đặt là 0 nếu sử dụng họ giao thức Internet)
- ❖ Sockfd: số hiệu mô tả socket đã tạo bởi hàm socket()
- ❖ \*localaddr: con trỏ chỉ đến địa chỉ socket cục bộ
- ❖ Localaddrlen: chiều dài của địa chỉ socket cục bộ
- ❖ \*serveraddr: con trỏ chỉ đến địa chỉ socket của server
- ❖ Serveraddrlen: chiều dài của địa chỉ socket server
- ❖ Backlog: số kết nối được yêu cầu tối đa có thể xếp vào hàng đợi
- ❖ \*clientaddr: con trỏ chỉ đến địa chỉ socket của client kết nối đến

## Ý nghĩa các tham biến được sử dụng trong các hàm chức năng (2/2)

- ❖ Clentaddrlen: chiều dài của địa chỉ client- socket
- ❖ \*buf: con trỏ chỉ đến vị trí bộ đệm để lưu thông tin nhận được
- ❖ Len: chiều dài của bộ đệm
- ❖ Flags: thường đặt là 0
- ❖ \*toaddr: con trỏ chỉ đến địa chỉ socket từ xa gửi dữ liệu đến
- ❖ Toaddrlen: chiều dài địa chỉ socket từ xa gửi dữ liệu đến
- ❖ \*fromaddr: con trỏ chỉ đến địa chỉ socket từ xa nhận dữ liệu về
- ❖ Fromaddrlen: chiều dài địa chỉ socket từ xa nhận dữ liệu về

## Vấn đề phân giải tên miền và địa chỉ

- ❖ Client application
  - ☐ Extract server name (e.g., from the URL)
  - ☐ Do *gethostbyname()* to trigger resolver code
- ❖ Server application
  - ☐ Extract client IP address from socket
  - ☐ Optional *gethostbyaddr()* to translate into name