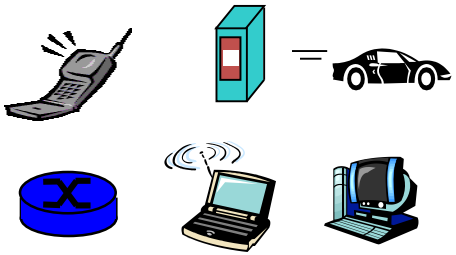
Chương 1: Ôn tập Mạng máy tính

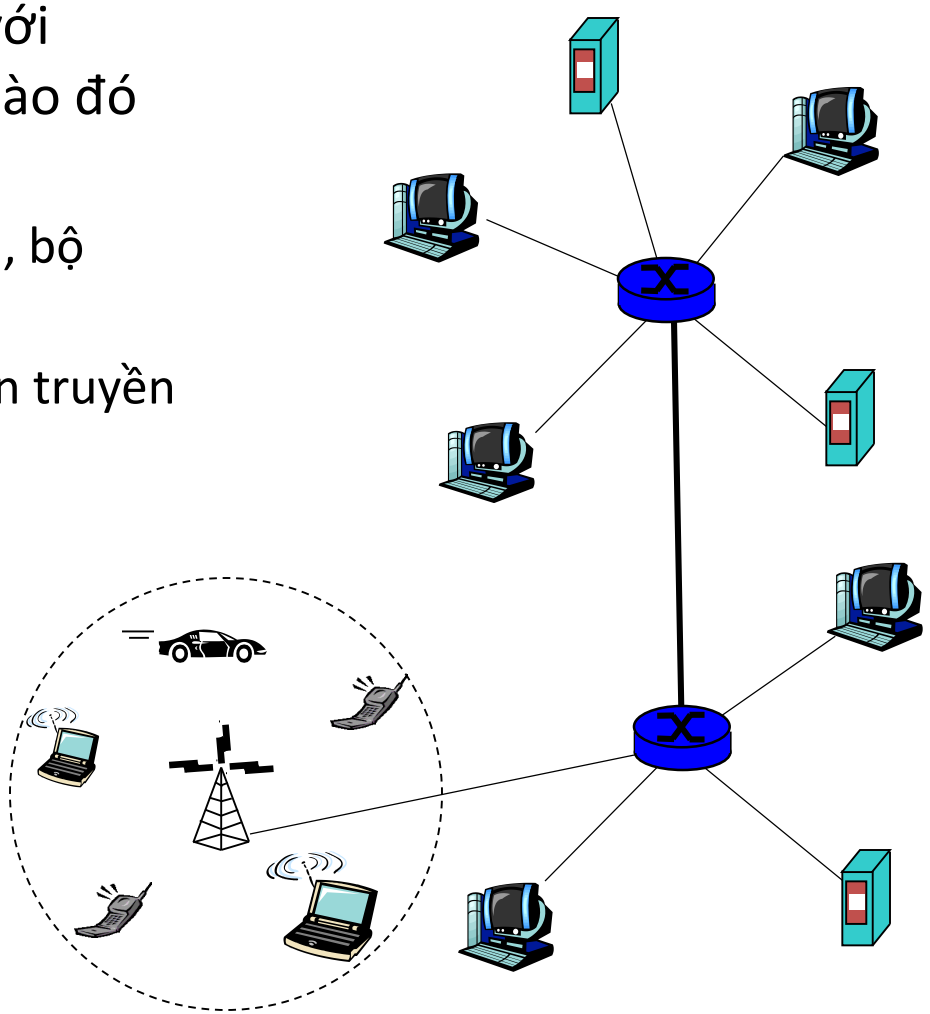
Nội dung

- Khái niệm mạng máy tính
  - Khái niệm
- OSI – TCP/IP
- Giao thức IP
- Tầng giao vận
- Hệ thống tên miền (DNS)
- Mô hình ứng dụng

Khái niệm

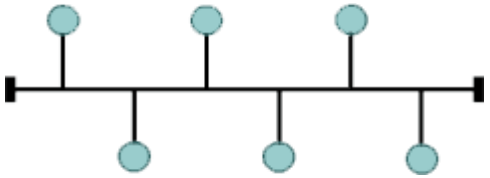
- Tập hợp các máy tính kết nối với nhau dựa trên một kiến trúc nào đó để có thể trao đổi dữ liệu
  - Máy tính: máy trạm, máy chủ, bộ định tuyến
  - Kết nối bằng một phương tiện truyền
  - Theo một kiến trúc mạng
- Các dạng máy tính?

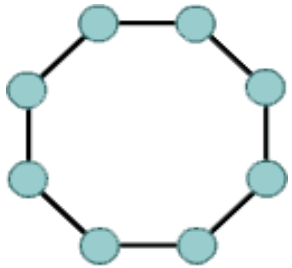


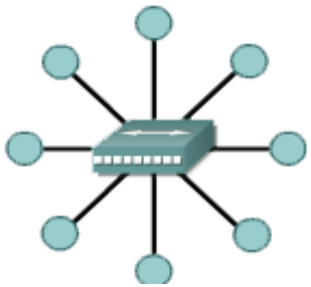


Kiến trúc mạng

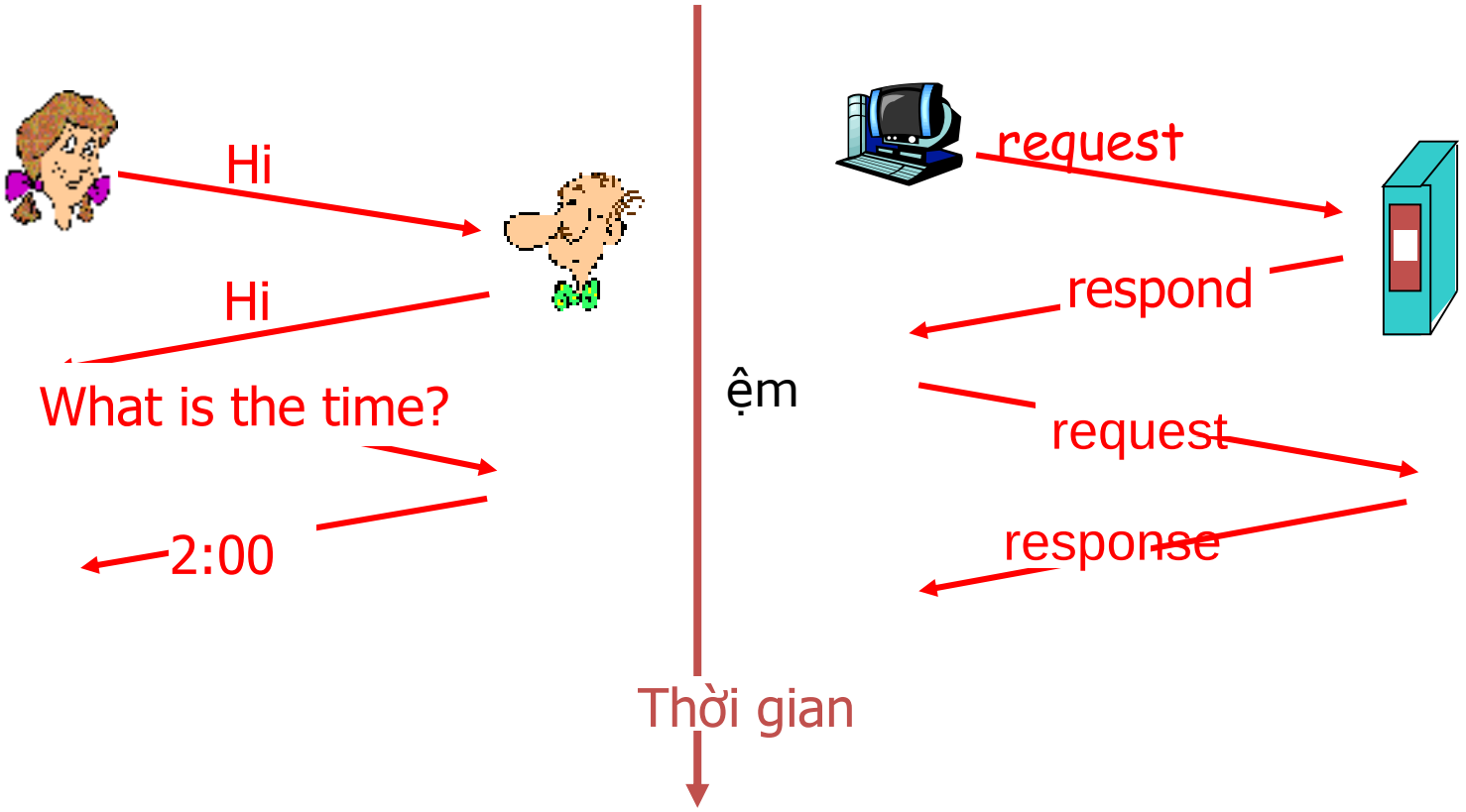
- Kiến trúc mạng: Hình trạng (topology) và giao thức (protocol)
- Hình trạng mạng
  - Trục (Bus), Vòng (Ring), Sao (Star)…
  - Thực tế là sự kết hợp của nhiều hình trạng khác nhau







Giao thức là gì?



Protocol person – person

Protocol machine - machine

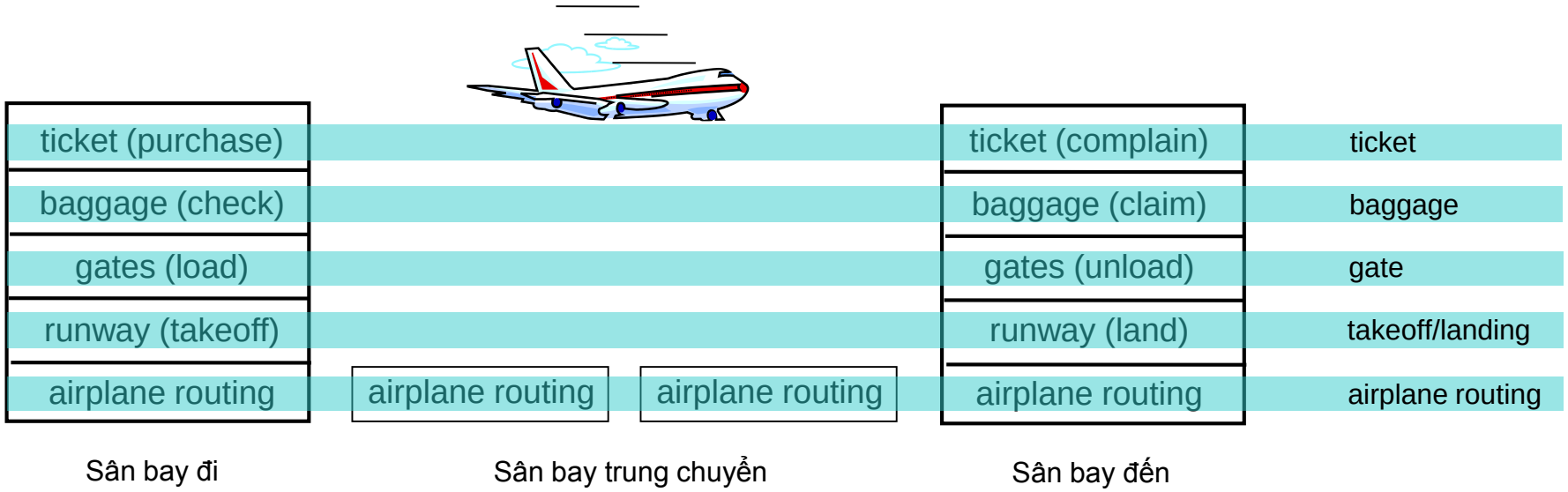
Nội dung

- Khái niệm mạng máy tính
- OSI – TCP/IP
  - Kiến trúc phân tầng
  - Mô hình OSI
  - Mô hình TCP/IP
- Giao thức IP
- Tầng giao vận
- Hệ thống tên miền (DNS)
- Mô hình ứng dụng

Vì sao phải phân tầng?

- Đối với các hệ thống phức tạp: nguyên lý *"chia để trị"*
- Cho phép xác định rõ nhiệm vụ của mỗi bộ phận và quan hệ giữa chúng
- Cho phép dễ dàng bảo trì và nâng cấp hệ thống
  - Thay đổi bên trong một bộ phận không ảnh hưởng đến các bộ phận khác
  - Như việc nâng cấp từ CD lên DVD player mà không phải thay loa.

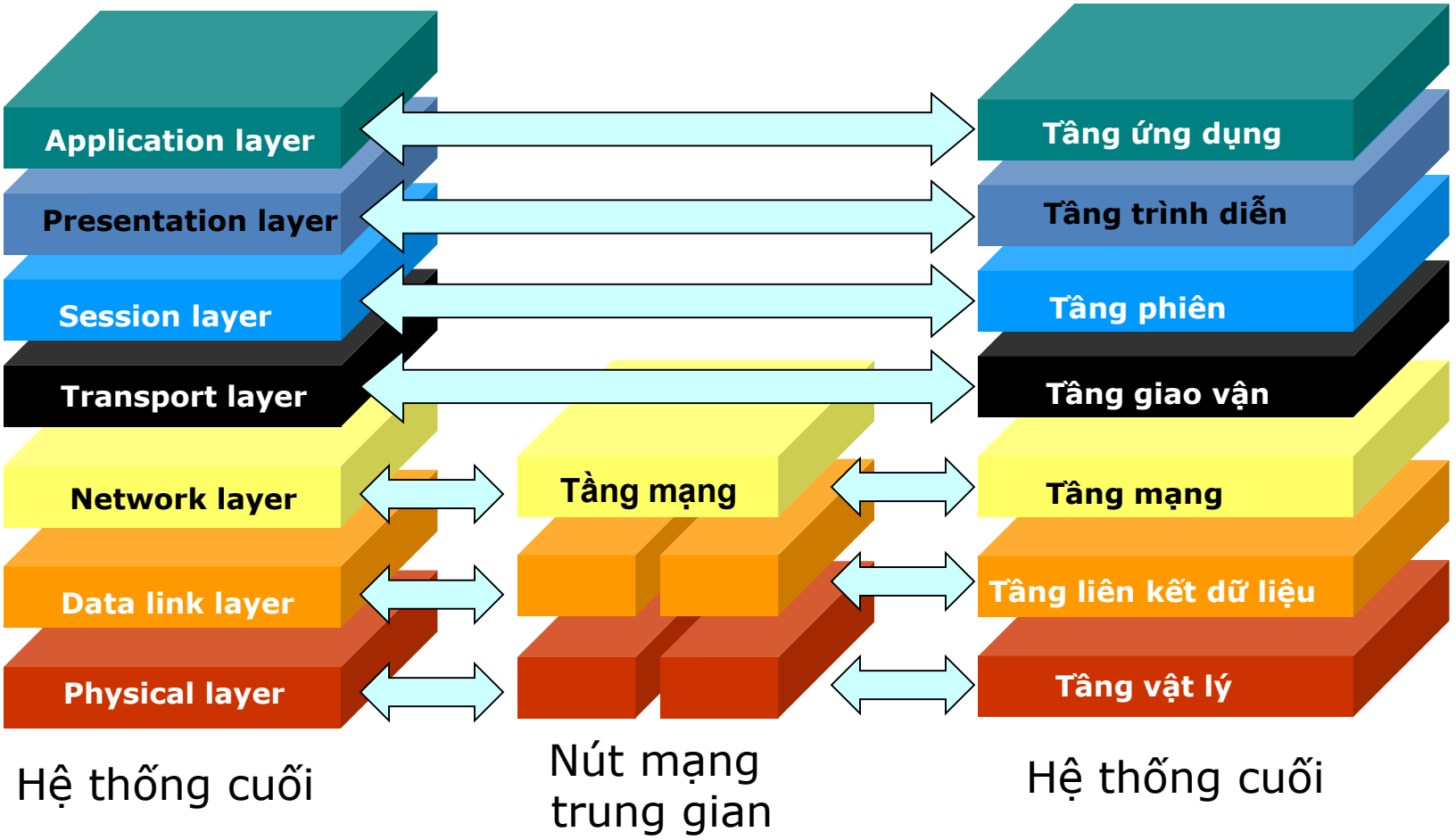
Phân tầng các chức năng hàng không



Tầng: Mỗi tầng có nhiệm vụ cung cấp 1 dịch vụ

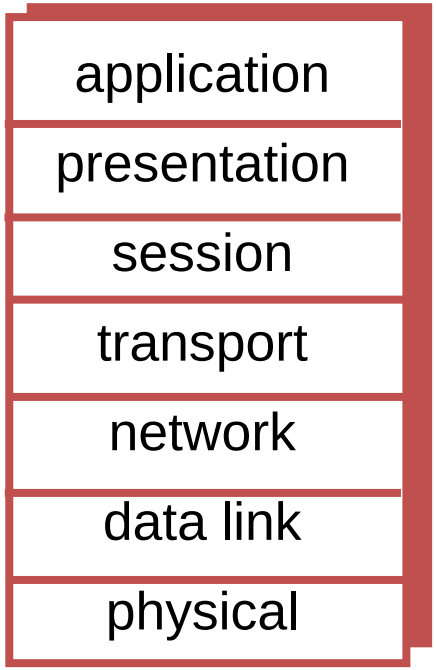
- Dựa trên các chức năng của chính tầng đó
- Dựa trên các dịch vụ cung cấp bởi tầng dưới

OSI - Open System Interconnection: Bao gồm 7 tầng

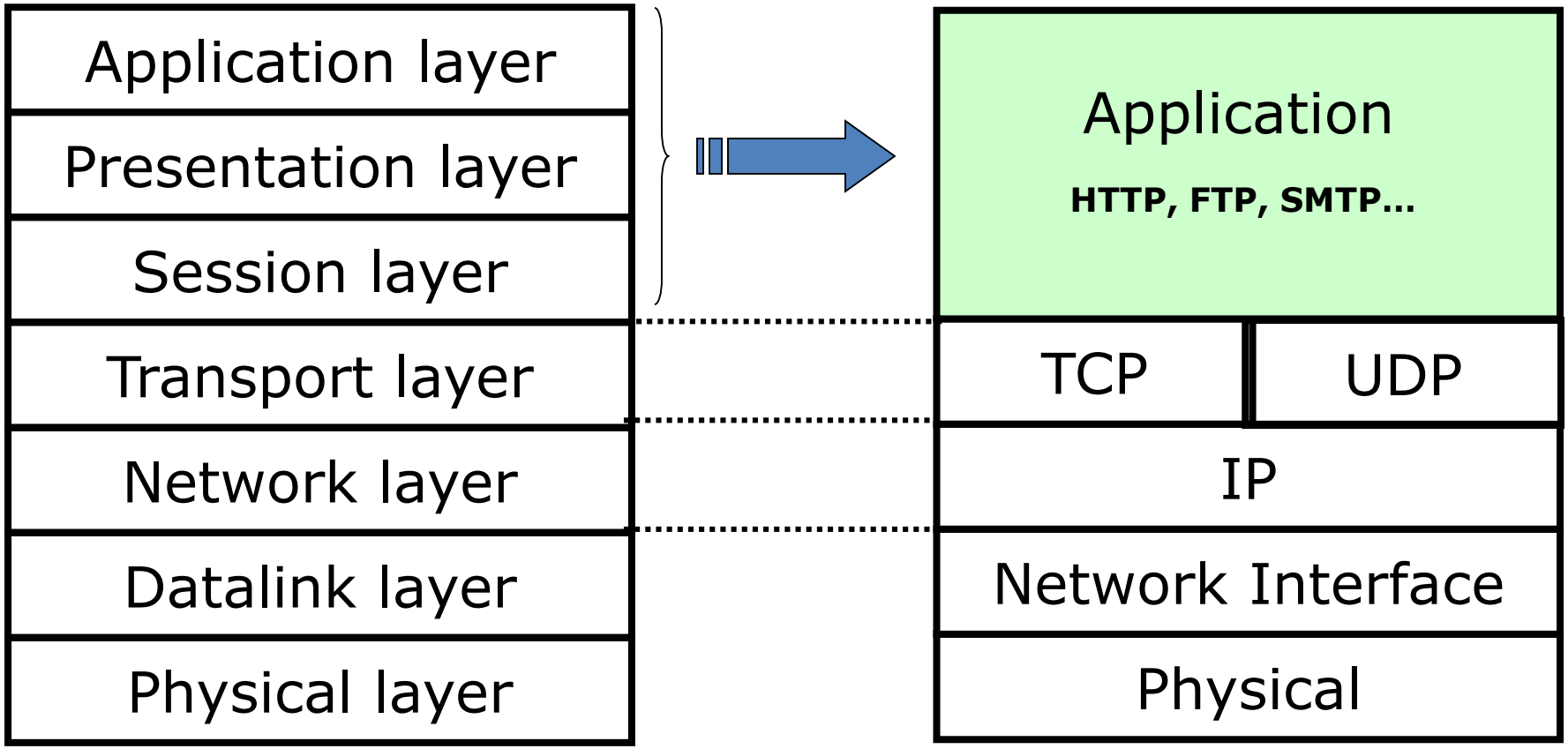


Chức năng chung của các tầng

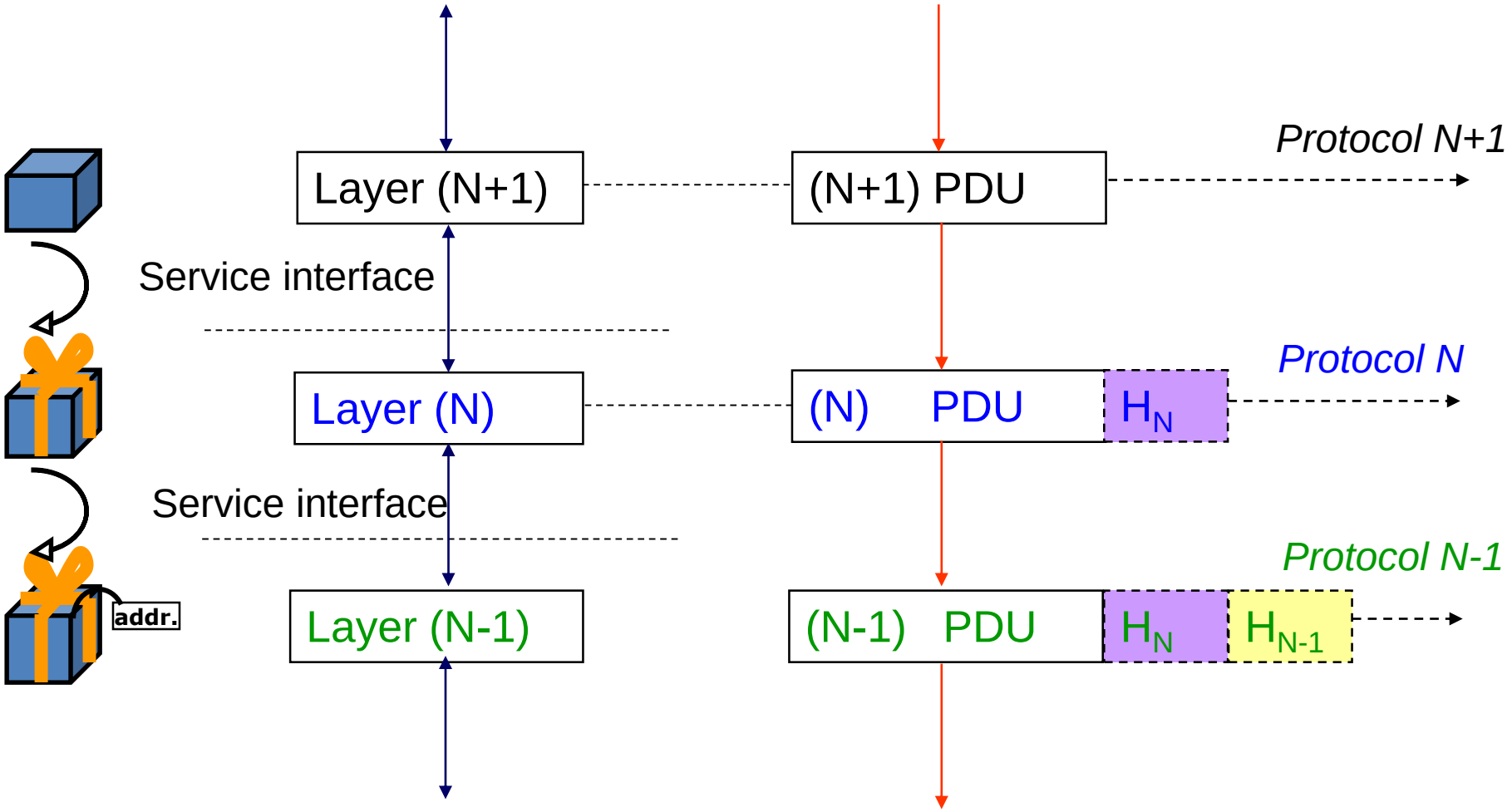
- Vật lý: Truyền bits “trên đường truyền”
- Liên kết dữ liệu: Truyền dữ liệu giữa các thành phần nối kết trong một mạng
- Mạng: Chọn đường, chuyển tiếp gói tin từ nguồn đến đích
- Giao vận: Xử lý việc truyền-nhận dữ liệu cho các ứng dụng
- Phiên: đồng bộ hóa, check-point, khôi phục quá trình trao đổi
- Trình diễn: cho phép các ứng dụng biểu diễn dữ liệu, e.g., mã hóa, nén, chuyển đổi…
- Ứng dụng: Hỗ trợ các ứng dụng trên mạng.



Mô hình OSI và TCP/IP

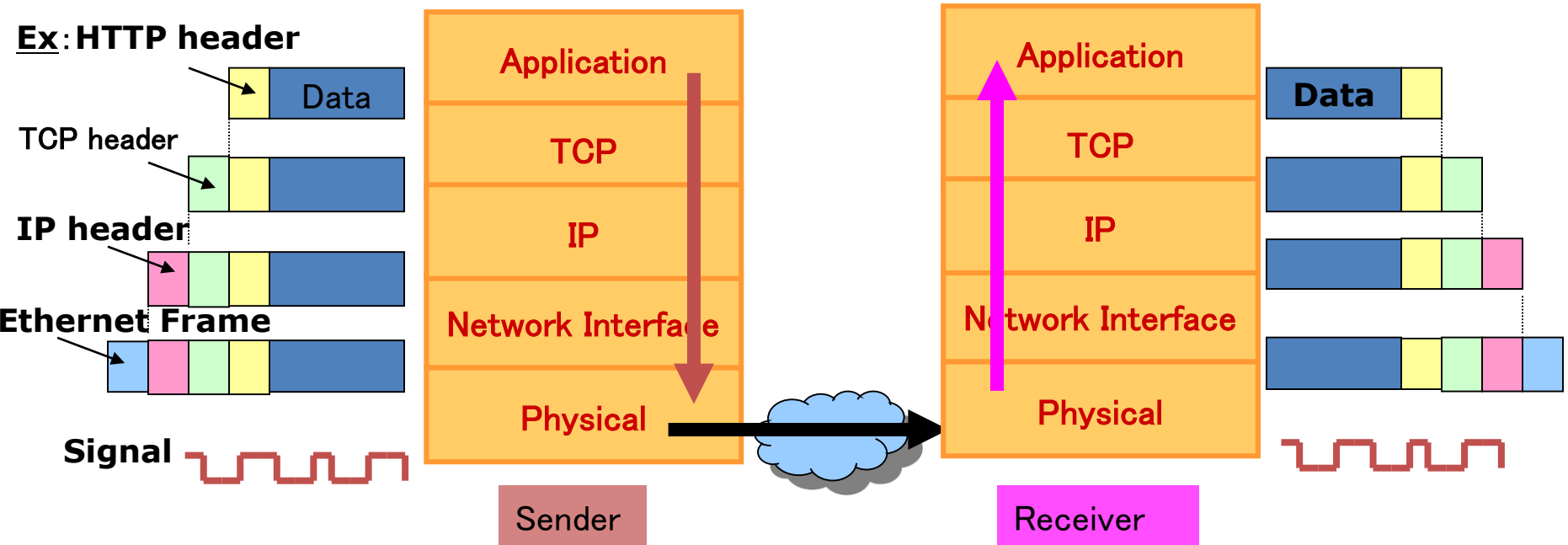


PDU: Protocol Data Unit – Đơn vị dữ liệu giao thức



Họ giao thức TCP/IP và quá trình đóng gói

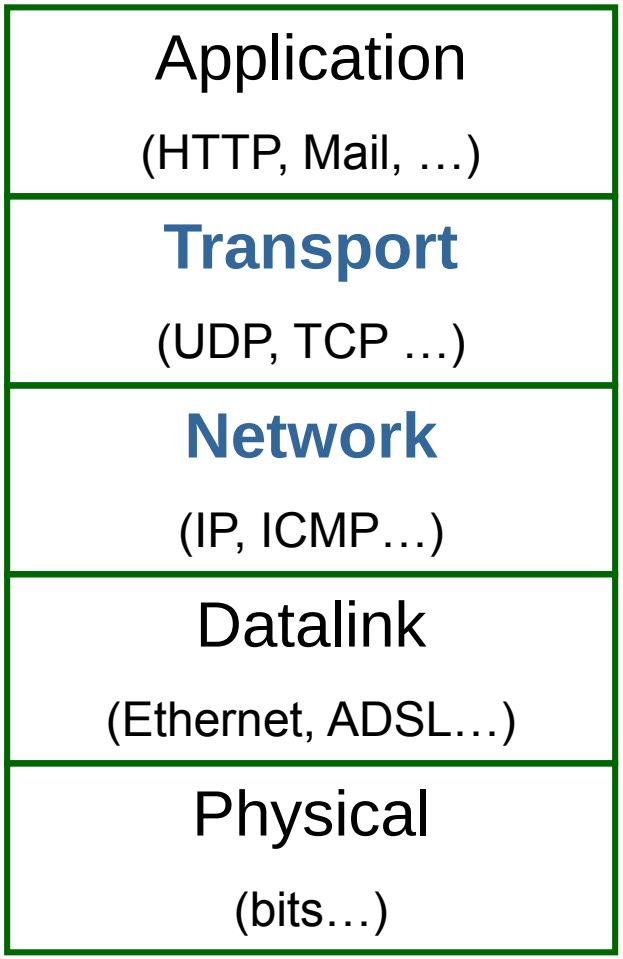
- Bên gửi
  - Mỗi tầng thêm vào các thông tin điều khiển vào phần đầu gói tin (header) và truyền xuống tầng dưới
- Bên nhận
  - Mỗi tầng xử lý gói tin dựa trên thông tin trong phần đầu, sau đó bỏ phần đầu, lấy phần dữ liệu chuyển lên tầng trên.



Nội dung

- Khái niệm mạng máy tính
- OSI – TCP/IP
- Giao thức IP
  - Giới thiệu giao thức IP
  - Địa chỉ IP
- Tầng giao vận
- Hệ thống tên miền (DNS)
- Mô hình ứng dụng

Tầng mạng – giao thức IP



Hỗ trợ các ứng dụng trên mạng

Truyền dữ liệu giữa các ứng dụng

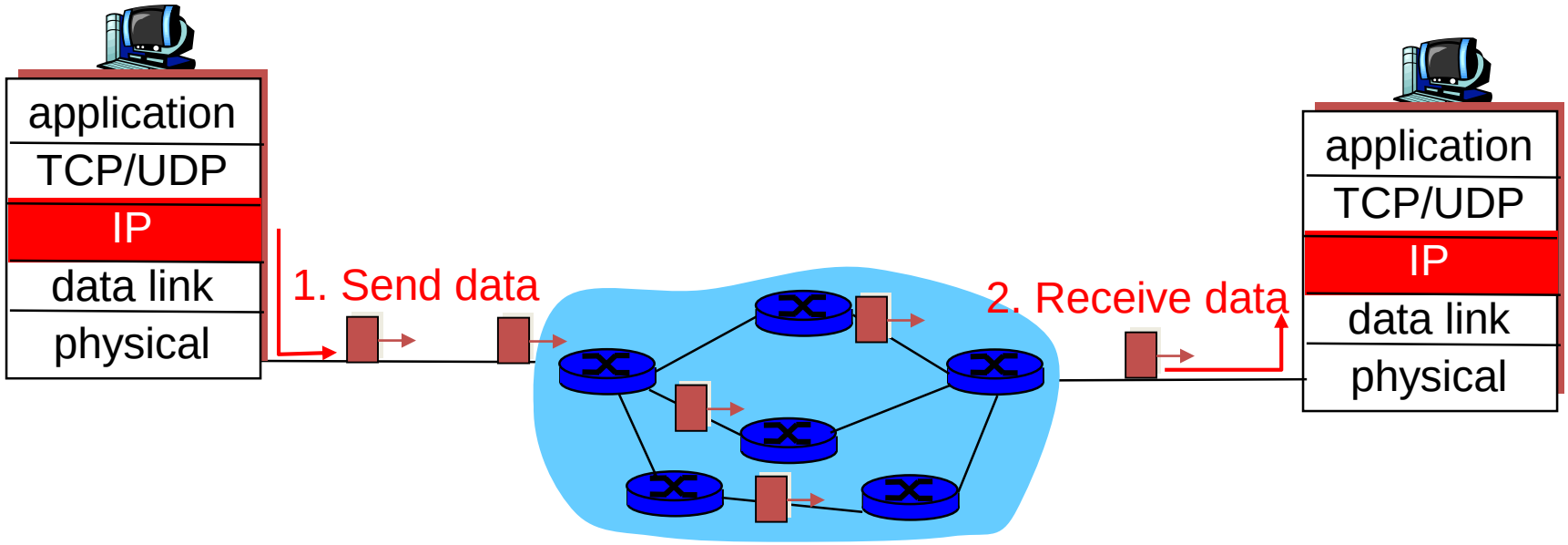
Chọn đường và chuyển tiếp gói tin giữa các máy, các mạng

Hỗ trợ việc truyền thông cho các thành phần kế tiếp trên cùng 1 mạng

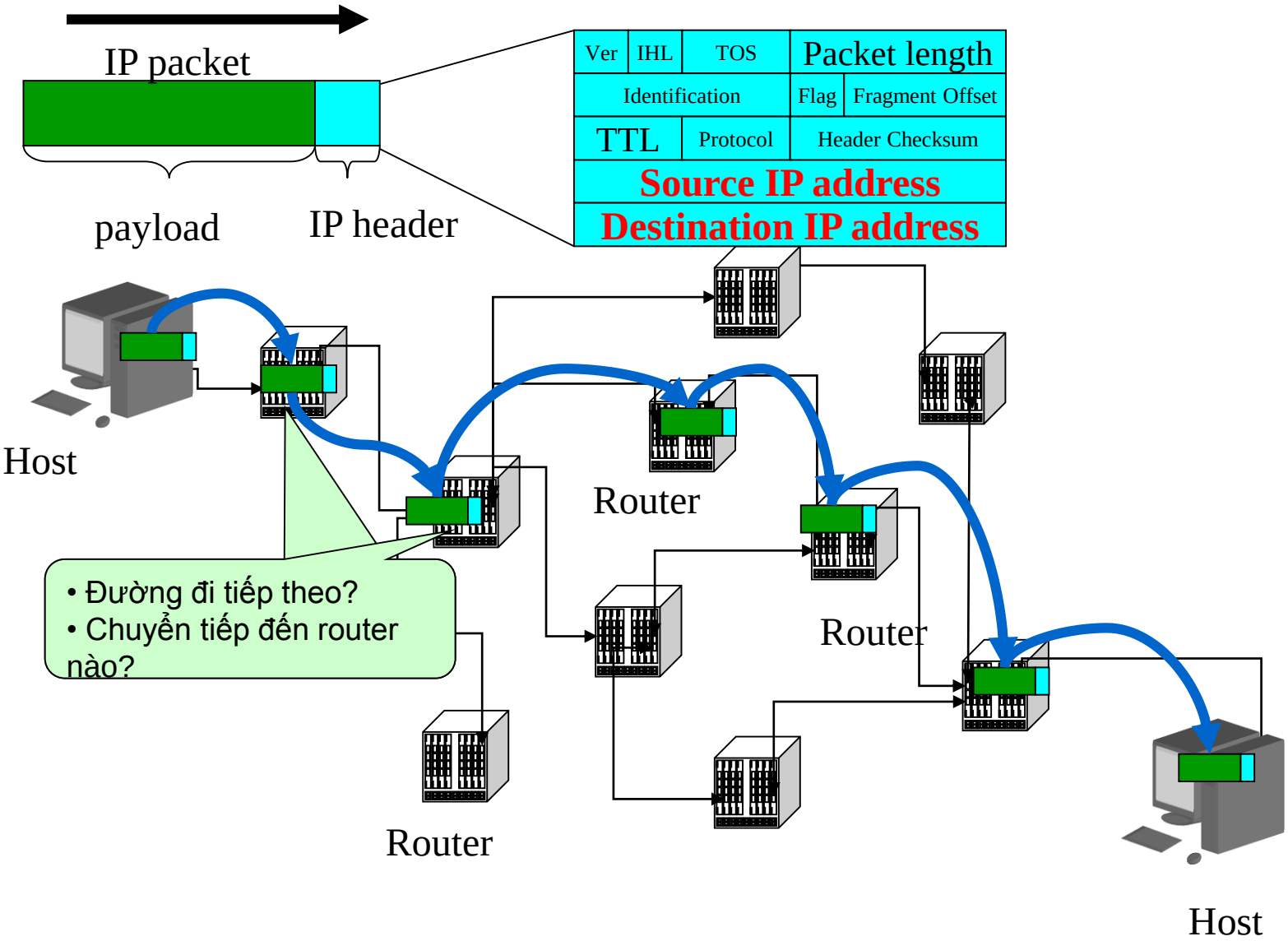
Truyền và nhận dòng bit trên đường truyền vật lý

Internet Protocol

- Là một giao thức ở tầng mạng
- Hai chức năng cơ bản
  - Chọn đường (*Routing*): Xác định đường đi của gói tin từ nguồn đến đích
  - Chuyển tiếp (*Forwarding)*: Chuyển dữ liệu từ đầu vào tới đầu ra của bộ định tuyến (router)



Chọn đường và chuyển tiếp gói tin

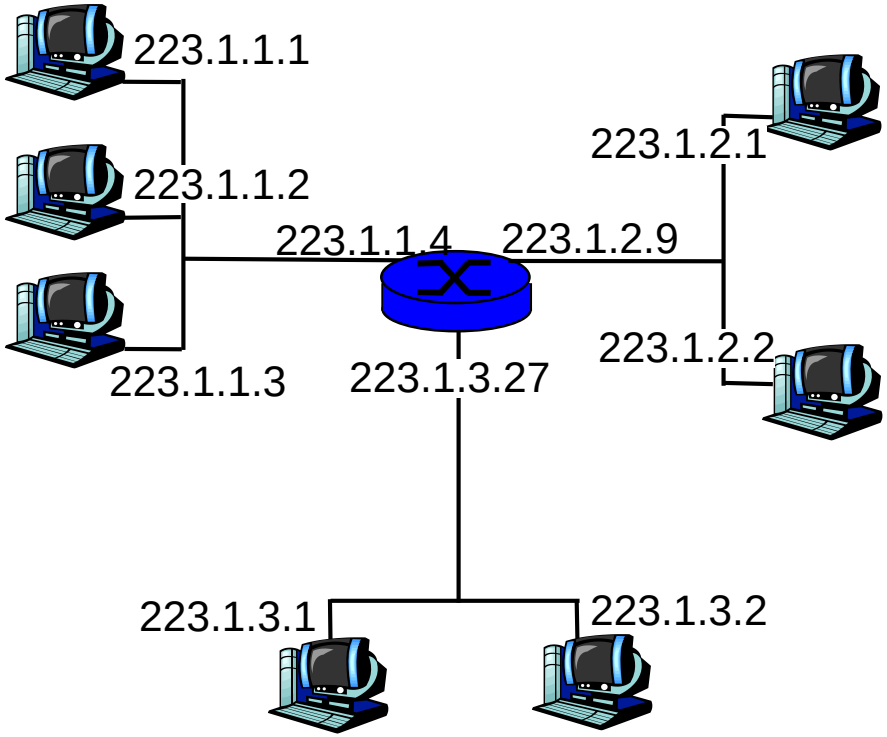


Đặc điểm của giao thức IP

- Không tin cậy / nhanh
  - Truyền dữ liệu theo phương thức "*best effort*"
  - IP không có cơ chế phục hồi lỗi
  - Khi cần, sẽ sử dụng dịch vụ tầng trên để đảm bảo độ tin cậy (TCP)
- Giao thức không liên kết
  - Các gói tin được xử lý độc lập

Địa chỉ IP (IPv4)

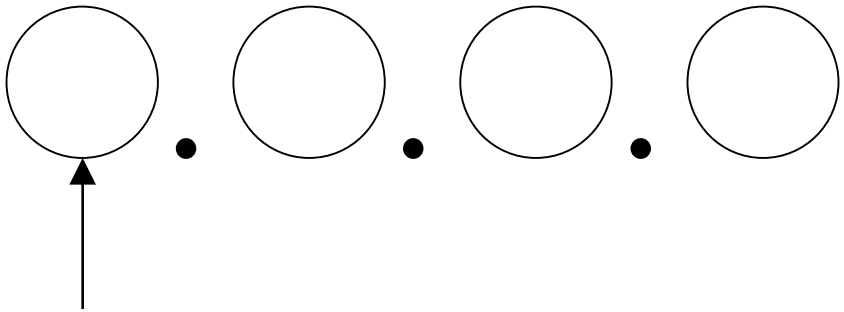
- Địa chỉ IP : Một số 32-bit để định danh giao diện máy trạm, bộ định tuyến
- Mỗi địa chỉ IP được gán cho một giao diện
- Địa chỉ IP có tính duy nhất



223.1.1.1 = 11011111 00000001 00000001 00000001

223 1 1 1

Ký hiệu thập phân có chấm



8 bits
0 – 255 integer

Ví dụ:

203.178.136.63	o
259.12.49.192	x
133.27.4.27	o

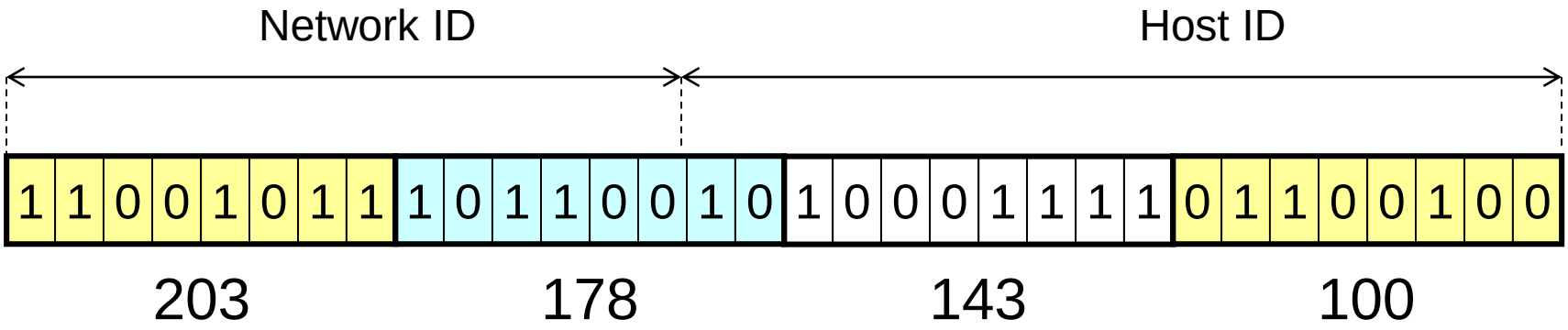
Sử dụng 4 phần 8 bits để miêu tả một địa chỉ 32 bits

3417476964

1	1	0	0	1	0	1	1	1	0	1	1	0	0	1	0	1	0	0	0	1	1	1	1	0	1	1	0	0	1	0	0
203								178								143								100							

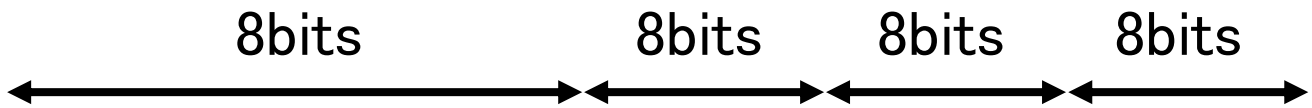
Địa chỉ máy trạm, địa chỉ mạng

- Địa chỉ IP có hai phần
  - Host ID – địa chỉ máy trạm
  - Network ID – địa chỉ mạng



- Làm thế nào biết được phần nào là cho máy trạm, phần nào cho mạng?
  - Phân lớp địa chỉ
  - Không phân lớp – CIDR

Phân lớp địa chỉ IP



Class A	0							7bit								H								H								H								
Class B	1				0						6bit								N								H								H				
Class C	1				1				0					5bit								N								N								H		
Class D	1				1				1				0																												
Class E	1				1				1				1																												

	# of network	# of hosts
Class A	128	2^24
Class B	16384	65536
Class C	2^21	256

Hạn chế của việc phân lớp địa chỉ

- Lãng phí không gian địa chỉ
  - Việc phân chia cứng thành các lớp (A, B, C, D, E) làm hạn chế việc sử dụng toàn bộ không gian địa chỉ

Cách giải quyết …

- CIDR: Classless Inter Domain Routing
  - Phần địa chỉ mạng sẽ có độ dài bất kỳ
  - Dạng địa chỉ: a.b.c.d/x, trong đó x (mặt nạ mạng) là số bit trong phần ứng với địa chỉ mạng

Mặt nạ mạng

- Mặt nạ mạng chia một địa chỉ IP làm 2 phần
  - Phần ứng với máy trạm
  - Phần ứng với mạng
- Dùng toán tử AND
  - Tính địa chỉ mạng
  - Tính khoảng địa chỉ IP

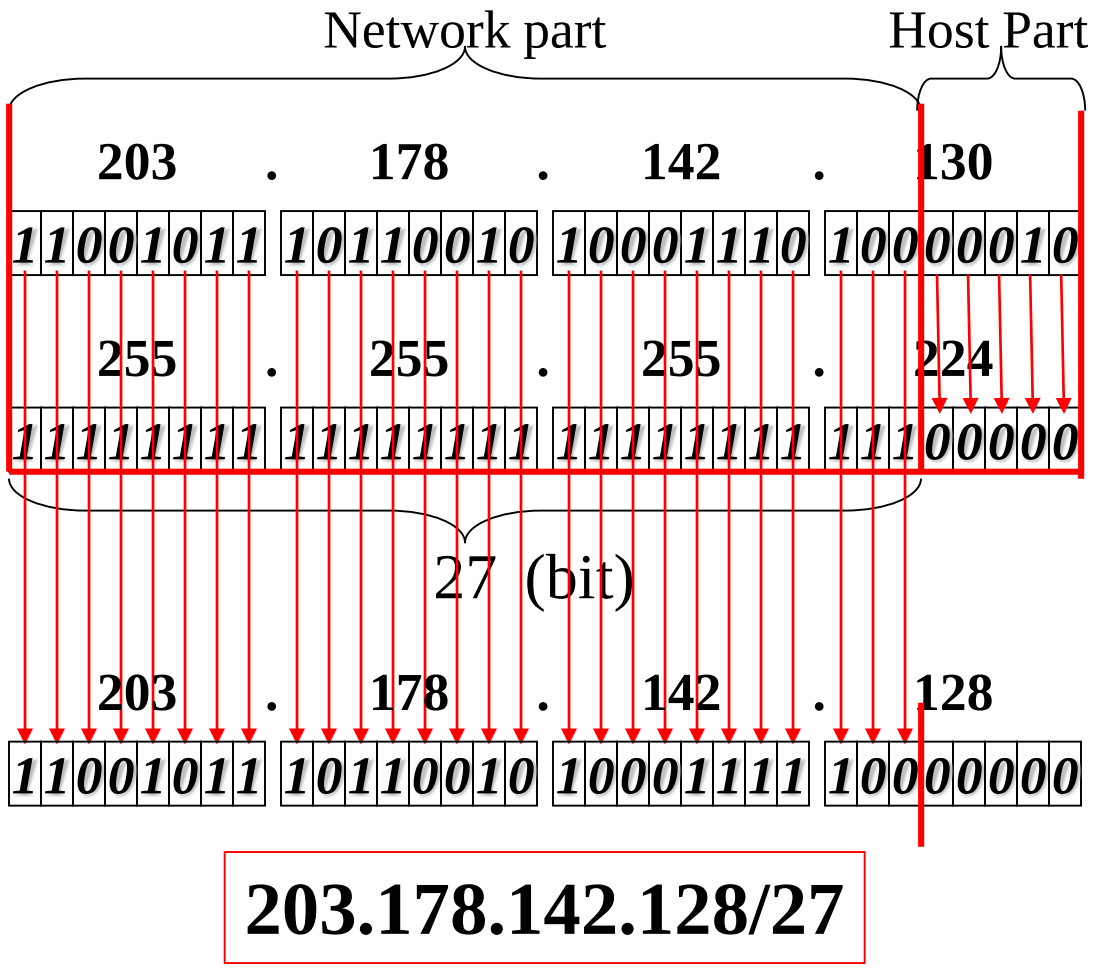
Cách tính địa chỉ mạng

IP Address

Netmask (/27)

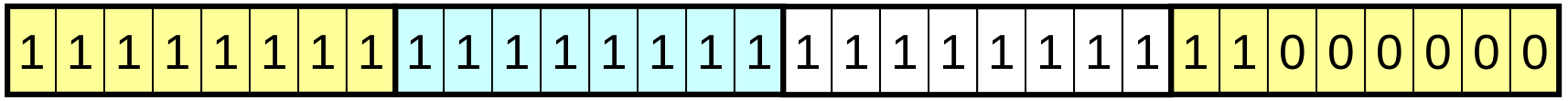
AND

Network address



203.178.142.128/27

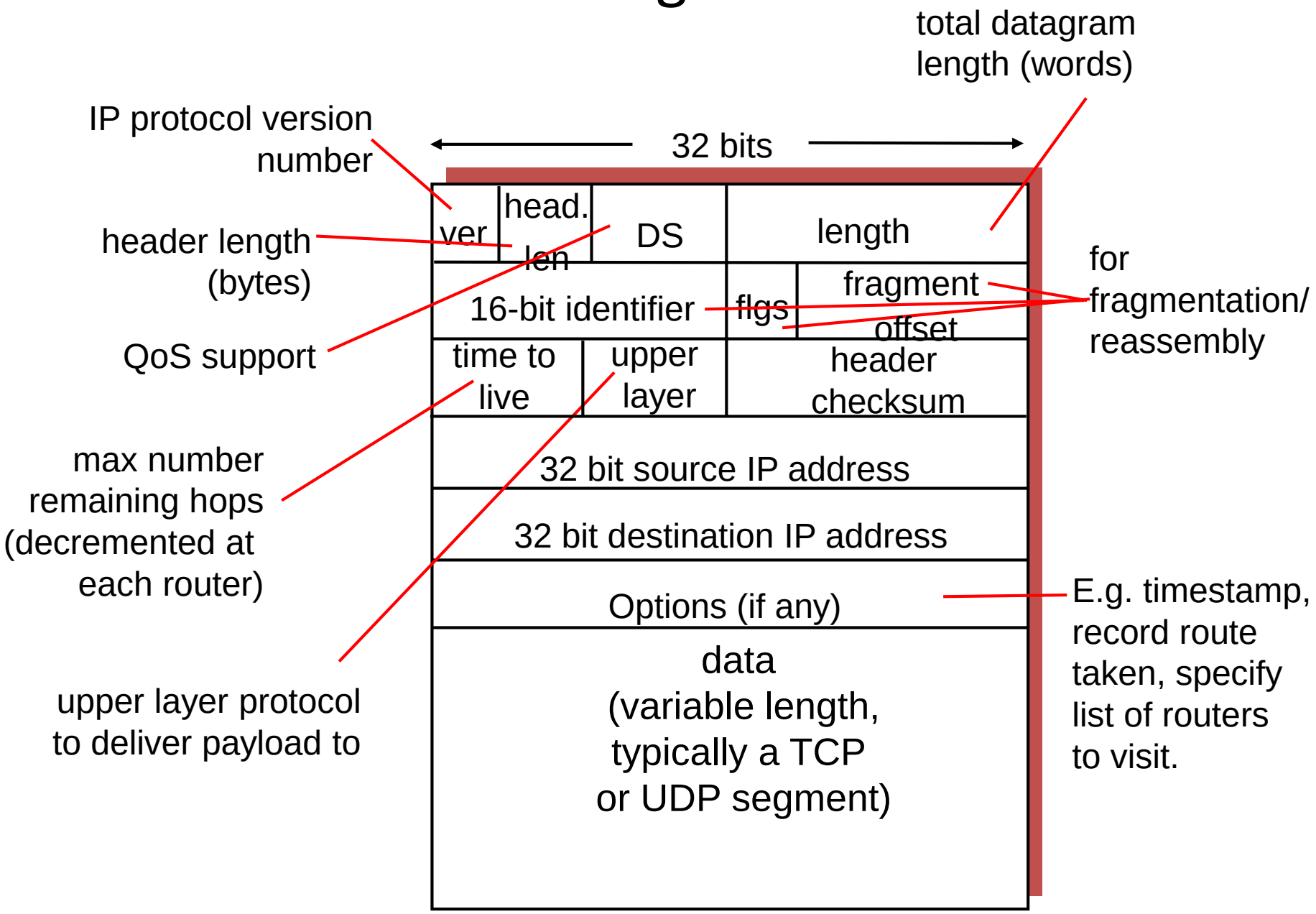
Mặt nạ mạng và kích thước mạng



- Kích thước
  - Theo lũy thừa 2
- RFC1878

- Trong trường hợp /26
  - Phần máy trạm = 6 bits
  - 2^6=64
  - Dải địa chỉ có thể gán:
    - 0 - 63
    - 64 - 127
    - 128 - 191
    - 192 - 255

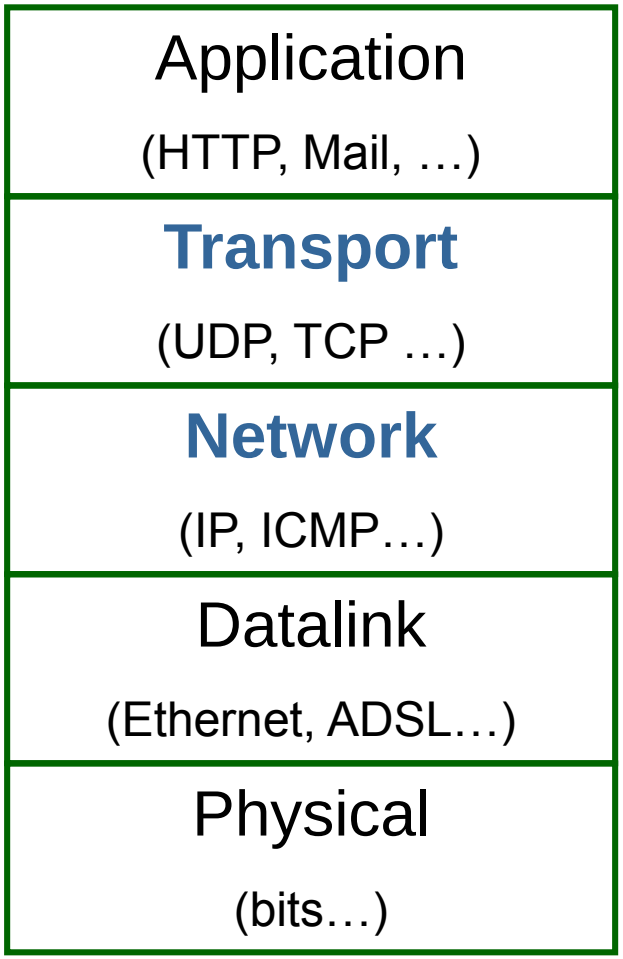
Phần đầu gói tin IP



Nội dung

- Khái niệm mạng máy tính
- OSI – TCP/IP
- Giao thức IP
- Tầng giao vận
  - Giới thiệu
  - UDP
  - TCP
- Hệ thống tên miền (DNS)
- Mô hình ứng dụng

Tầng giao vận



Hỗ trợ các ứng dụng trên mạng

Truyền dữ liệu giữa các ứng dụng

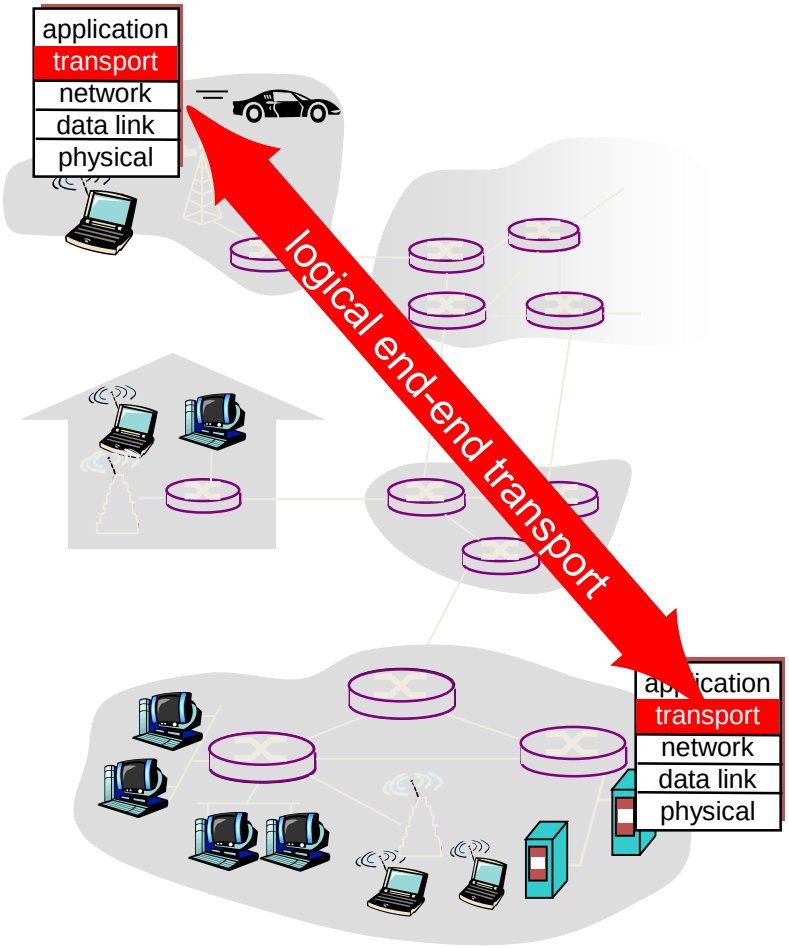
Chọn đường và chuyển tiếp gói tin giữa các máy, các mạng

Hỗ trợ việc truyền thông cho các thành phần kế tiếp trên cùng 1 mạng

Truyền và nhận dòng bit trên đường truyền vật lý

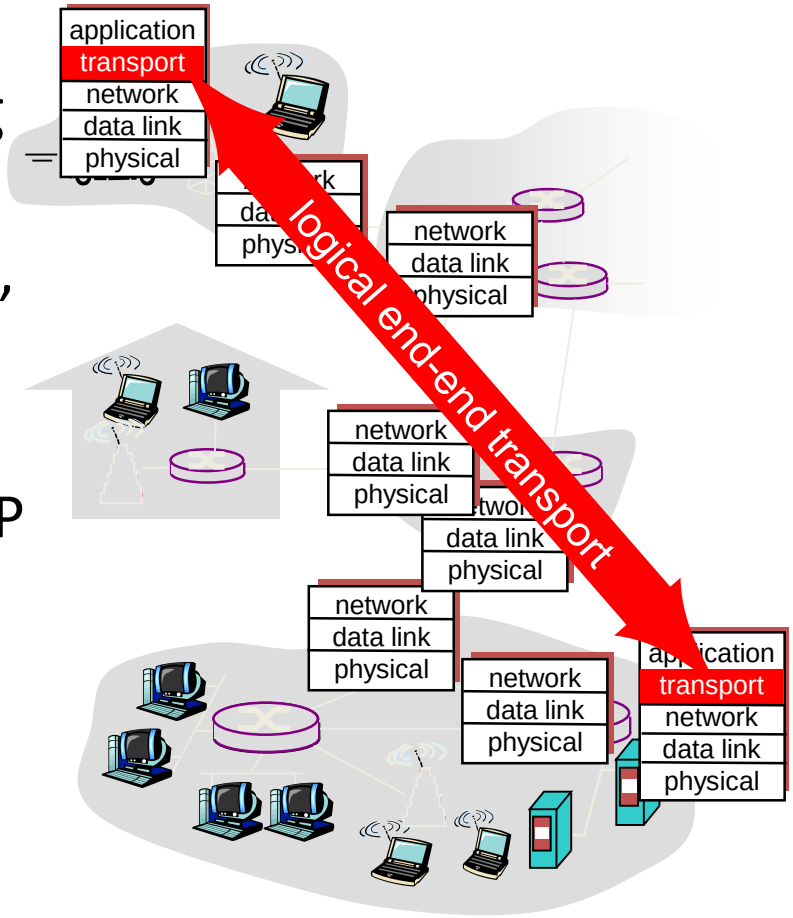
Tổng quan về tầng giao vận (1)

- Cung cấp phương tiện truyền giữa các ứng dụng cuối
- Bên gửi:
  - Nhận dữ liệu từ ứng dụng
  - Đặt dữ liệu vào các đoạn tin và chuyển cho tầng mạng
  - Nếu dữ liệu quá lớn, nó sẽ được chia làm nhiều phần và đặt vào nhiều đoạn tin khác nhau
- Bên nhận:
  - Nhận các đoạn tin từ tầng mạng
  - Tập hợp dữ liệu và chuyển lên cho ứng dụng



Tổng quan về tầng giao vận (2)

- Được cài đặt trên các hệ thống cuối
  - Không cài đặt trên các routers, switches…
- Hai dạng dịch vụ giao vận à
  - Tin cậy, hướng liên kết, e.g TCP
  - Không tin cậy, không liên kết, e.g. UDP
  - Tại sao cần 2 giao thức TCP và UDP ??

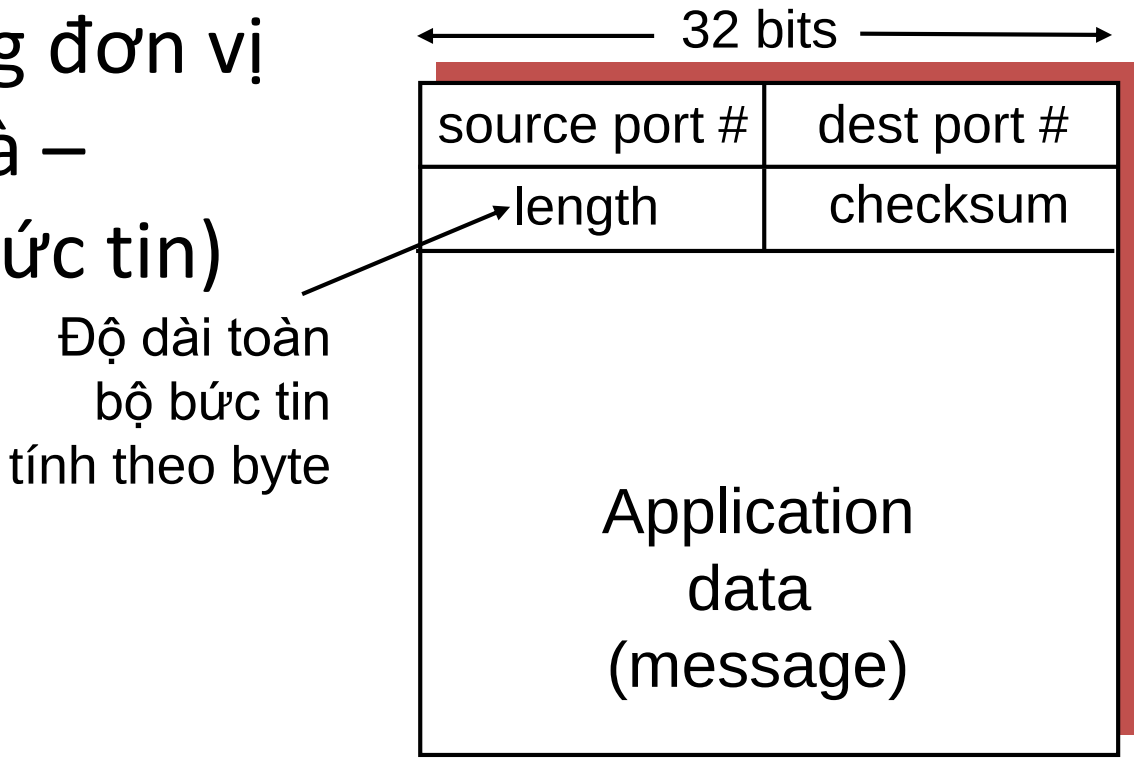


UDP – User Datagram Protocol

- Vì sao cần UDP?
  - Không cần thiết lập liên kết (tăng độ trễ)
  - Đơn giản: Không cần lưu lại trạng thái liên kết ở bên gửi và bên nhận
  - Phần đầu đoạn tin nhỏ
  - Không có quản lý tắc nghẽn: UDP cứ gửi dữ liệu nhanh nhất, nhiều nhất nếu có thể
- UDP có những chức năng cơ bản gì?
  - Dồn kênh/phân kênh
  - Phát hiện lỗi bit bằng checksum

Khuôn dạng bức tin (datagram)

- UDP sử dụng đơn vị dữ liệu gọi là – datagram (bức tin)

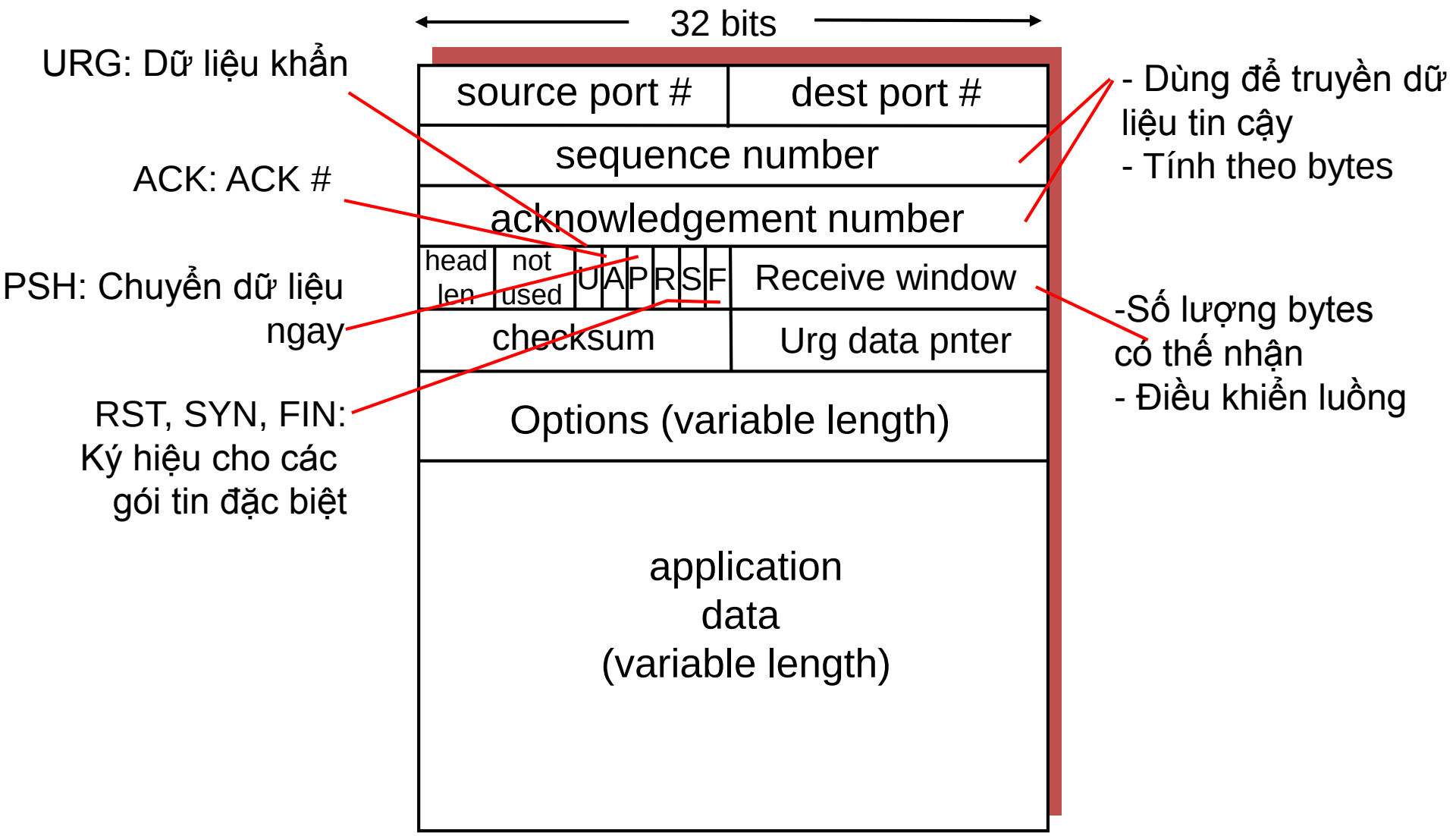


Khuôn dạng đơn vị
dữ liệu của UDP

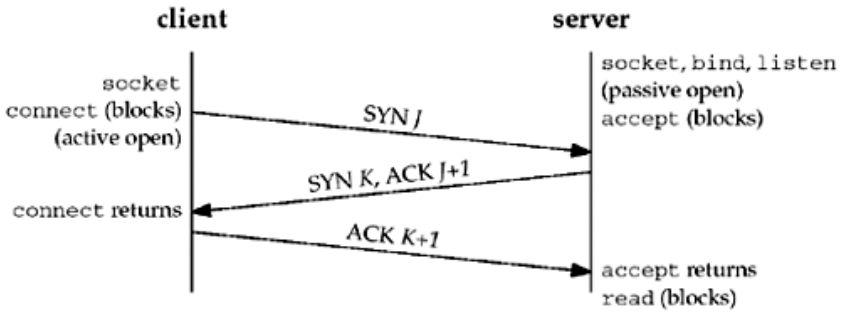
TCP – Transmission Control Protocol

- Giao thức hướng liên kết
  - Bắt tay ba bước
- Giao thức truyền dữ liệu theo dòng byte, tin cậy
  - Sử dụng vùng đệm
- Truyền theo kiểu pipeline
  - Tăng hiệu quả
- Kiểm soát luồng
  - Bên gửi không làm quá tải bên nhận (thực tế: quá tải)
- Kiểm soát tắc nghẽn
  - Việc truyền dữ liệu không nên làm tắc nghẽn mạng (thực tế: luôn có tẵc nghẽn)

Khuôn dạng đoạn tin - TCP segment



Thiết lập liên kết TCP : Giao thức bắt tay 3 bước

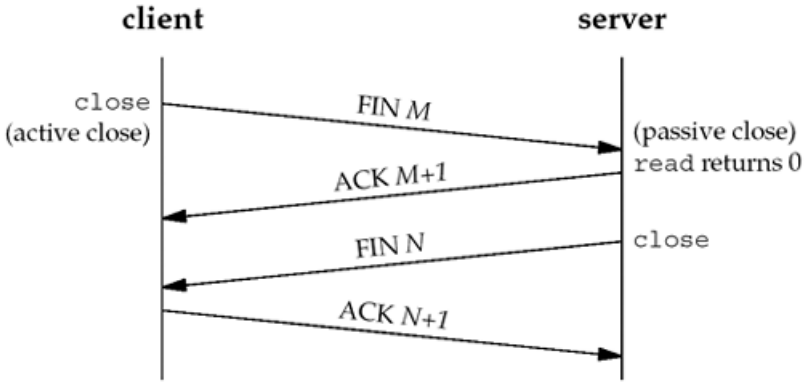


- Bước 1: A gửi SYN cho B
  - chỉ ra giá trị khởi tạo seq # của A
  - không có dữ liệu
- Bước 2: B nhận SYN, trả lời bằng SYNACK
  - B khởi tạo vùng đệm
  - chỉ ra giá trị khởi tạo seq. # của B
- Bước 3: A nhận SYNACK, trả lời ACK, có thể kèm theo dữ liệu

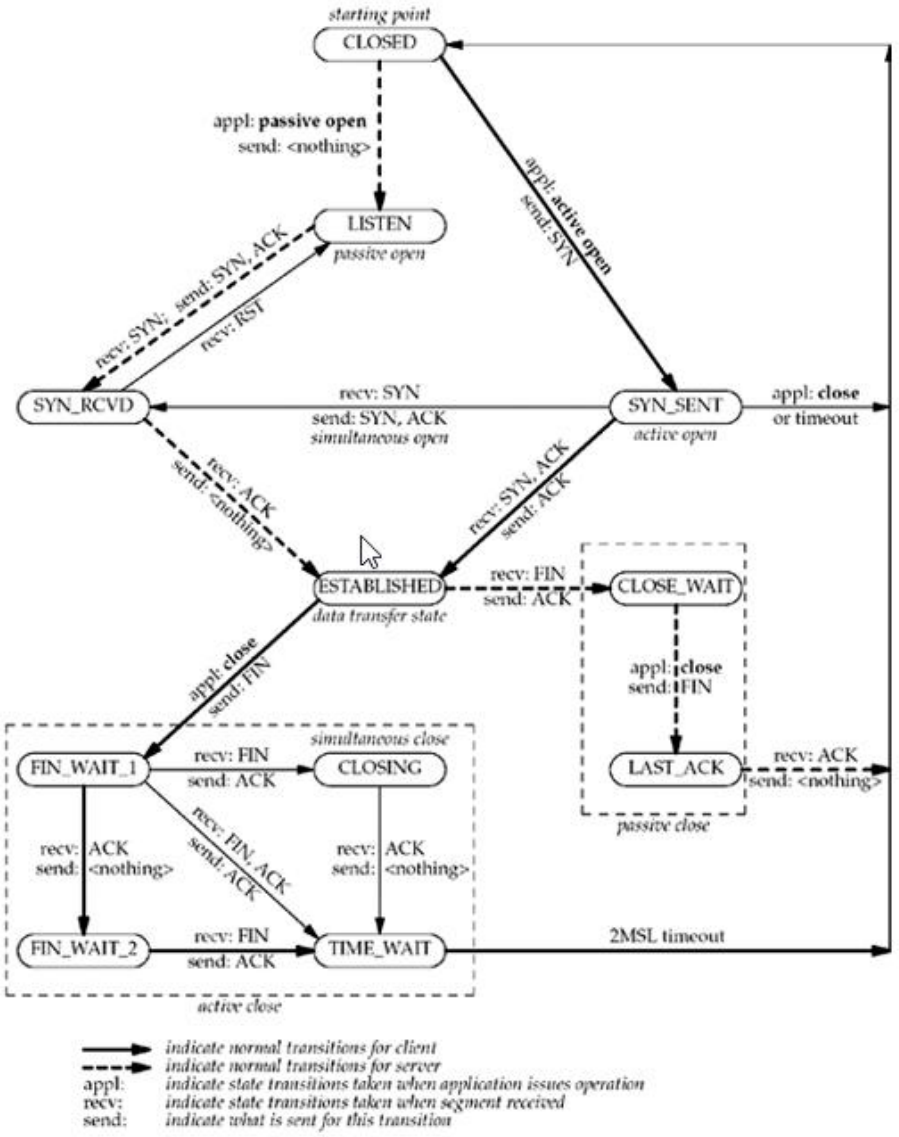
Ví dụ về việc đóng liên kết

- Bước 1: Gửi FIN cho B
- Bước 2: B nhận được FIN, trả lời ACK, đồng thời đóng liên kết và gửi FIN.
- Bước 3: A nhận FIN, trả lời ACK, vào trạng thái "chờ".
- Bước 4: B nhận ACK. đóng liên kết.

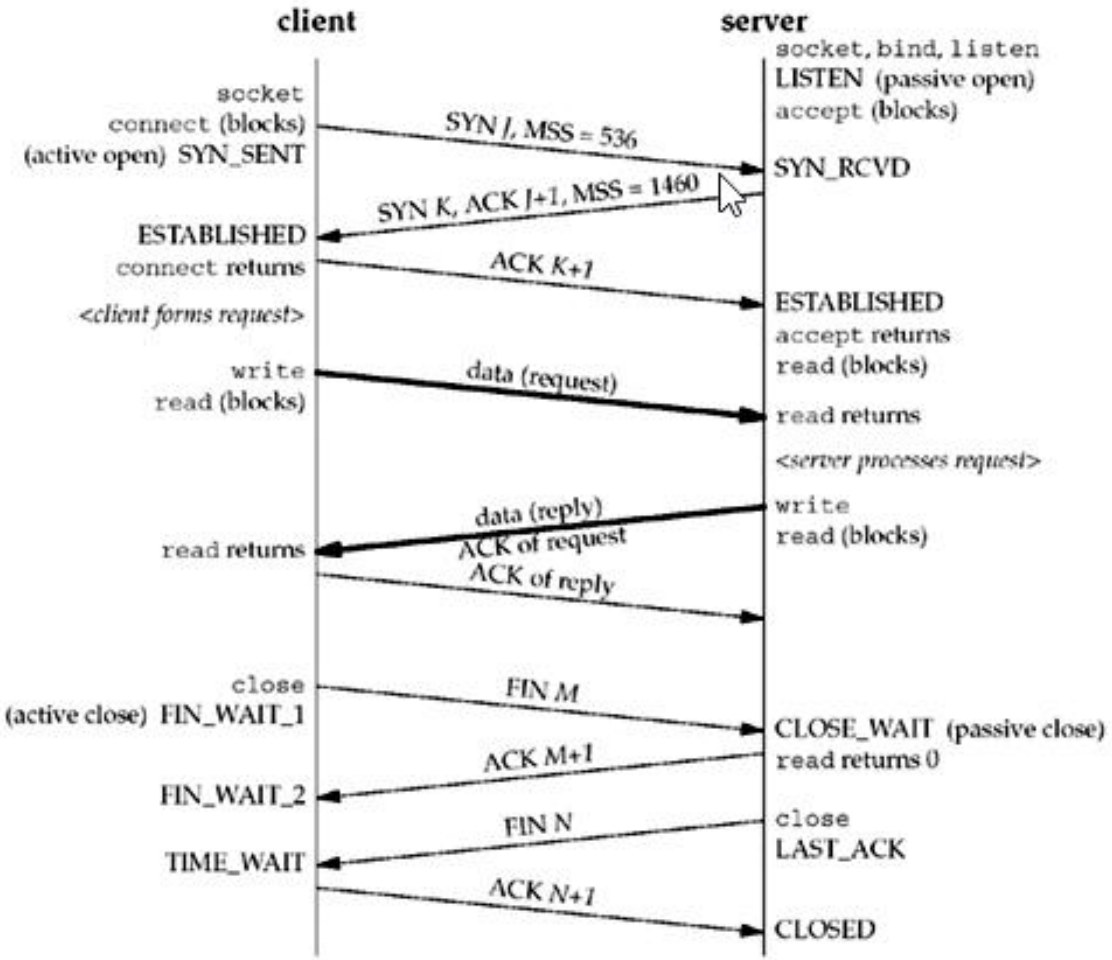
Lưu ý: Cả hai bên đều có thể chủ động đóng liên kết



Biểu đồ chuyển trạng thái của TCP



Các gói tin trao đổi trong TCP



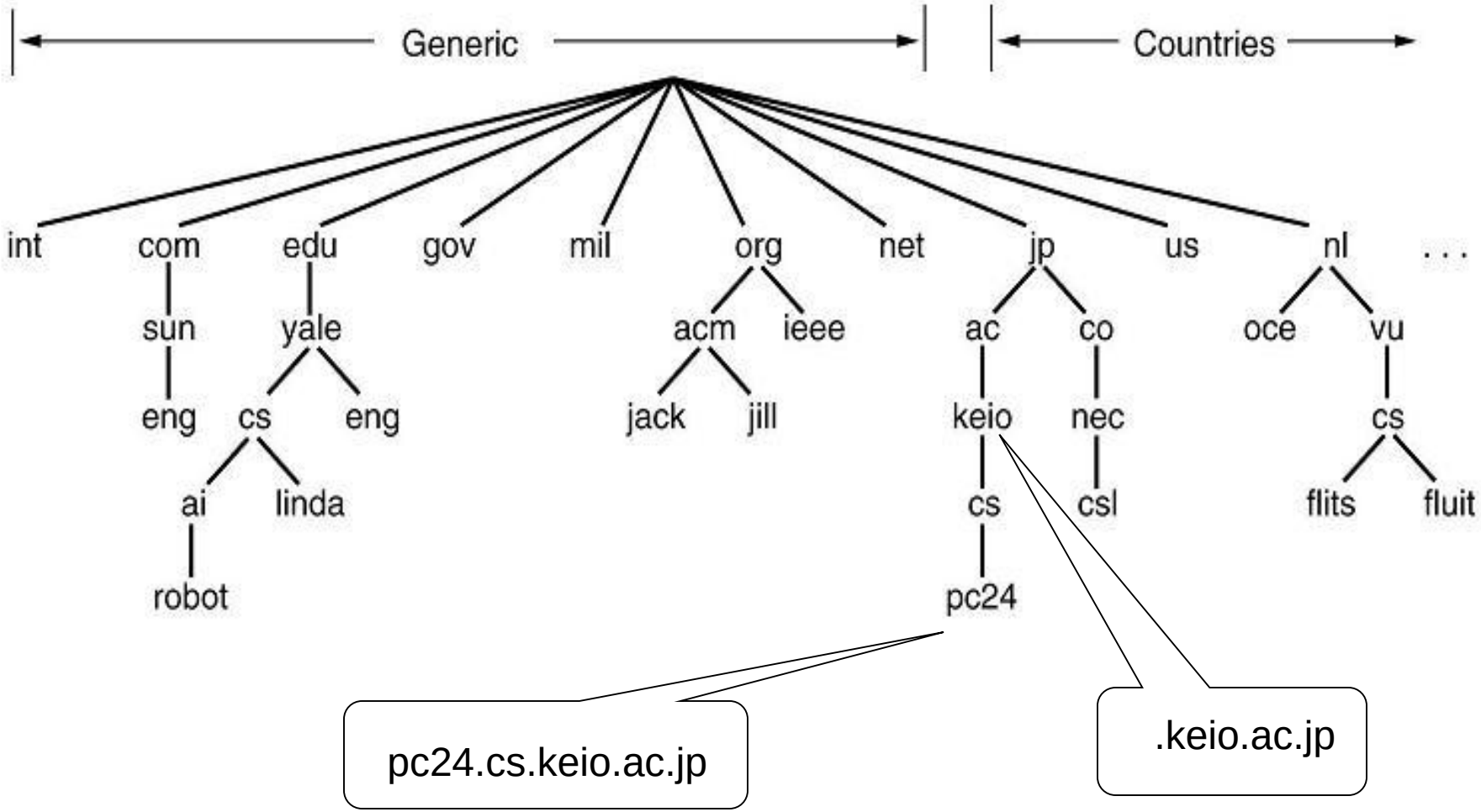
Nội dung

- Khái niệm mạng máy tính
- OSI – TCP/IP
- Giao thức IP
- Tầng giao vận
- Hệ thống tên miền (DNS)
- Mô hình ứng dụng

Tên miền

- Domain Name
  (FQDN: Fully Qualified Domain Name)
  - Tên miền là tên của một máy tính hay của một mạng máy tính, sử dụng tên (chữ cái, chữ số)
    - www.keio.ac.jp
    - www.hedspi.hut.edu.vn
    - soict.hut.edu.vn

Không gian tên miền



Tên và địa chỉ

- Trước khi truyền tin, máy trạm phải được xác định
  - Bởi một địa chỉ IP, hoặc
  - Bởi một tên miền (thuận tiện cho NSD)
- Tên
  - Độ dài thay đổi
  - Dễ nhớ cho con người
  - Không liên quan tới vị trí vật lý của máy
- Địa chỉ
  - Độ dài cố định
  - Dễ cho máy tính để xử lý
  - Liên quan tới vấn đề chọn đường

Chuyển đổi địa chỉ và ví dụ

- Máy tính thích dùng số
- Người thích dùng tên

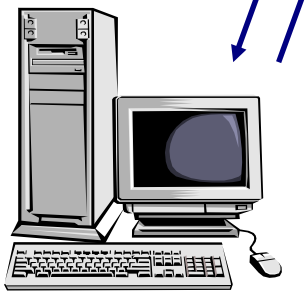
User

Tôi muốn vào địa chỉ
www.hedspi.hut.edu.vn

Cần có chuyển đổi địa chỉ

Mời truy cập vào
202.47.142.40

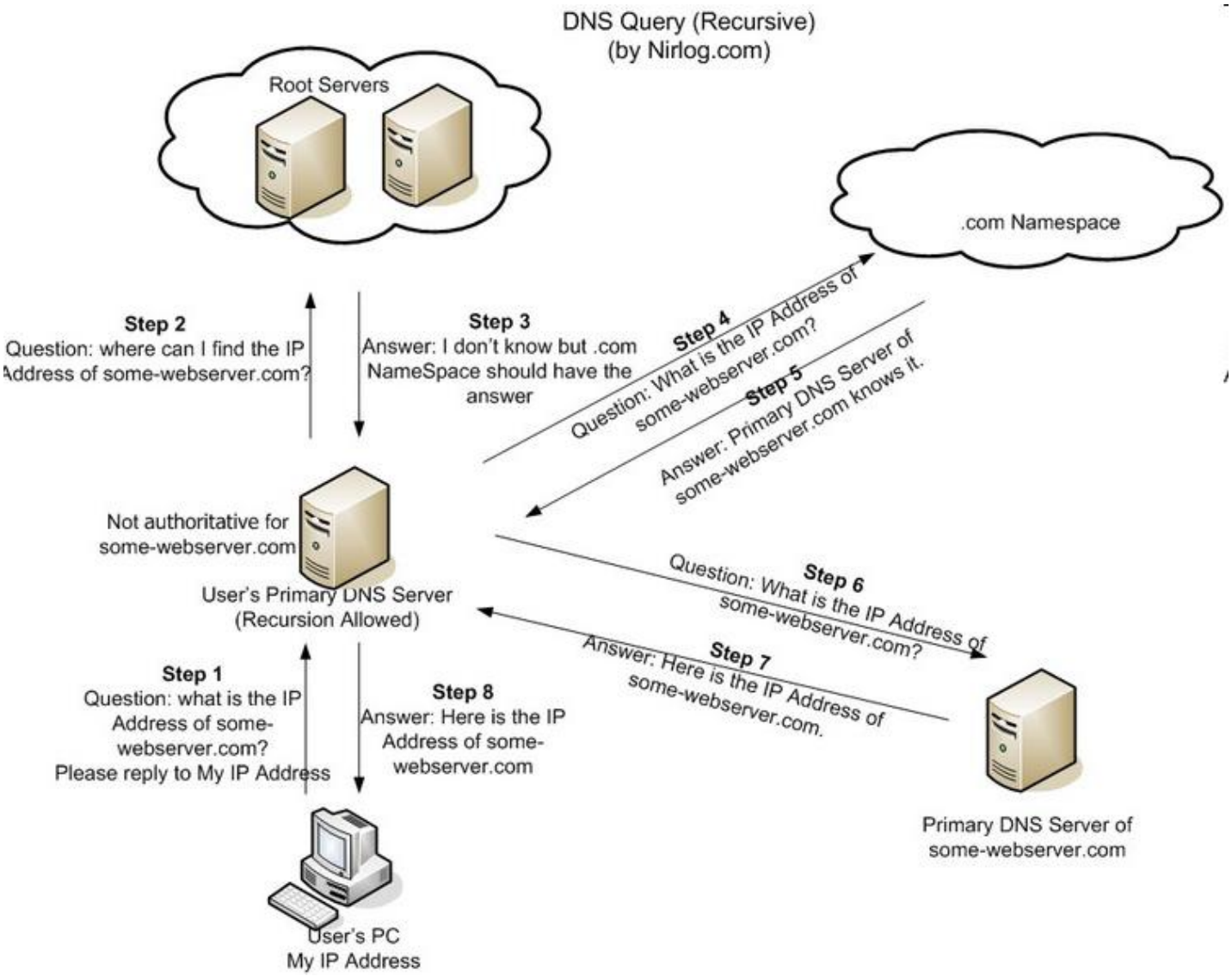
Máy chủ tên miền



Máy chủ web
202.47.142.40

Bạn cũng có thể nhập địa chỉ trực tiếp

Cơ chế phân giải địa chỉ tên miền



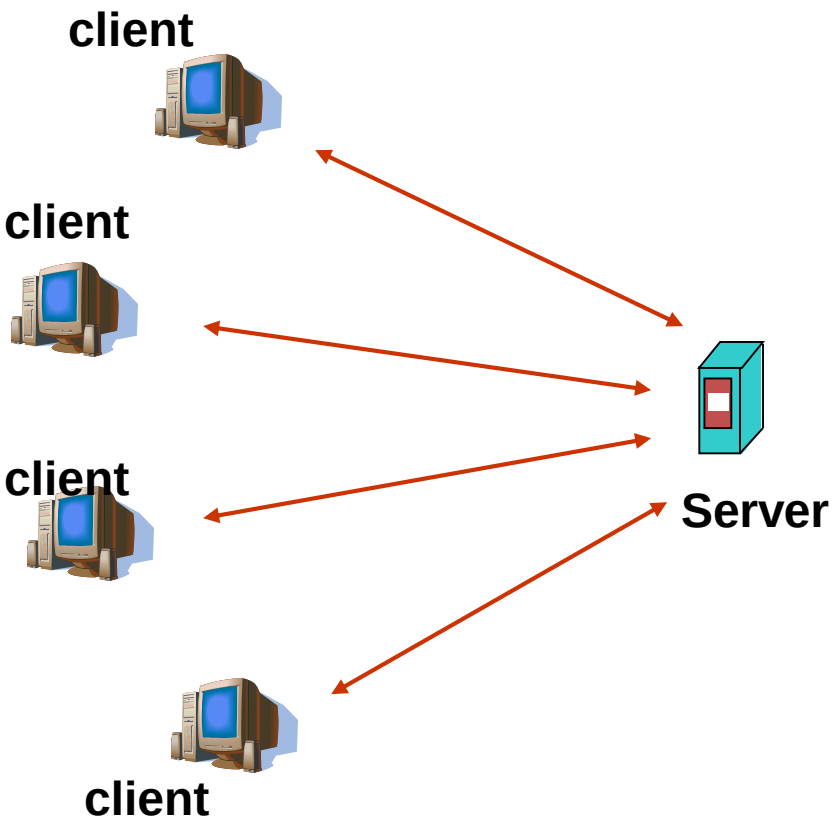
Cơ chế phân giải địa chỉ tên miền (2)

- Bước 1 : máy khách hỏi máy phân giải tên miền cục bộ (ngay trên máy cục bộ)
  - Nếu có bản ghi → trả lời lại máy khách
  - Nếu không → hỏi máy ROOT
- Bước 2 : hỏi máy ROOT địa chỉ IP của tên miền tương ứng
- Bước 3 : máy chủ trả về kết quả hoặc máy quản lý tên miền cấp lớn hơn
- Bước 4 : hỏi máy chủ do máy ROOT trả về → tiếp tục cho đến khi đến được máy quản lý tên miền hỏi

Nội dung

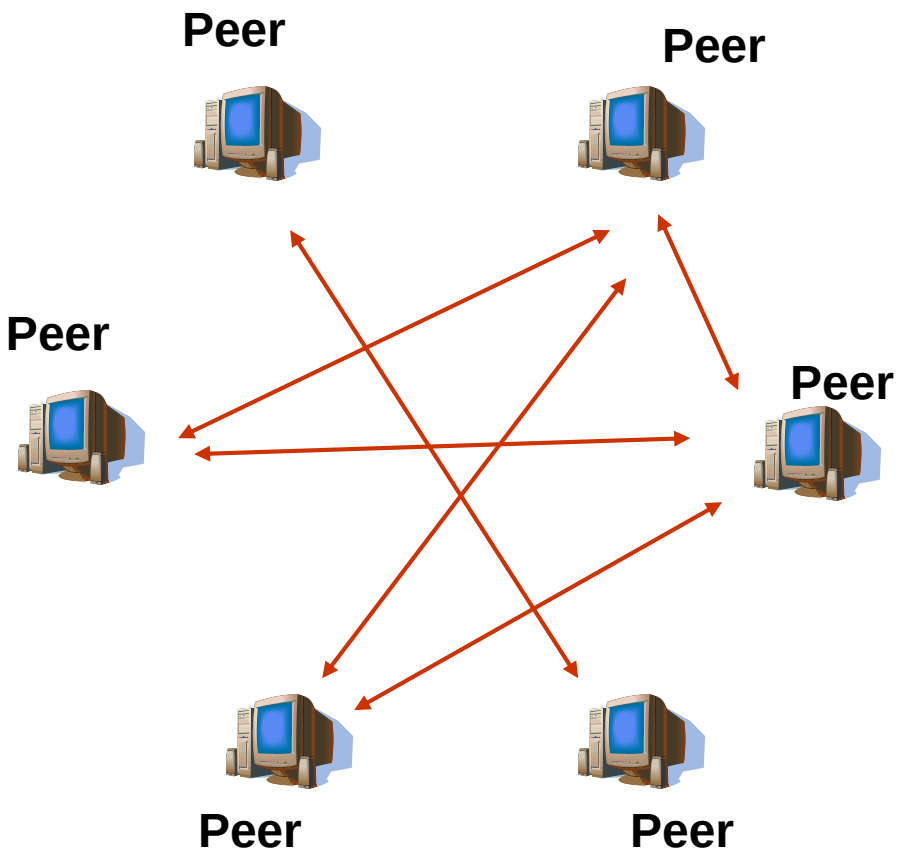
- Khái niệm mạng máy tính
- OSI – TCP/IP
- Giao thức IP
- Tầng giao vận
- Hệ thống tên miền (DNS)
- Mô hình ứng dụng

Mô hình khách chủ



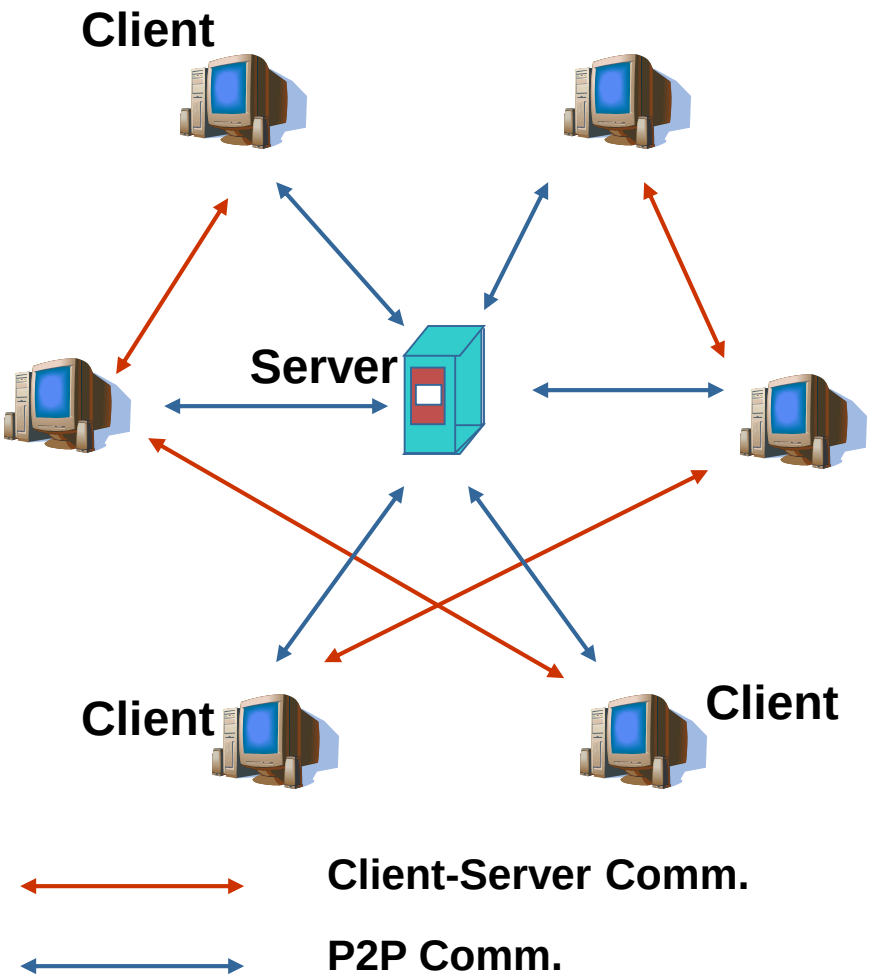
- Khách
  - Gửi yêu cầu truy cập dịch vụ đến máy chủ
  - Về nguyên tắc, không liên lạc trực tiếp với các máy khách khác
- Chủ
  - Thường xuyên online để chờ y/c đến từ máy trạm
  - Có thể có máy chủ dự phòng để nâng cao hiệu năng, phòng sự cố
- e.g. Web, Mail, …

Mô hình điểm-điểm thuần túy



- Không có máy chủ trung tâm
- Các máy có vai trò ngang nhau
- Hai máy bất kỳ có thể liên lạc trực tiếp với nhau
- Không cần vào mạng thường xuyên
- E.g. Gnutella

Mô hình lai



- Một máy chủ trung tâm để quản lý NSD, thông tin tìm kiếm…
- Các máy khách sẽ giao tiếp trực tiếp với nhau sau khi đăng nhập
- E.g. Skype
  - Máy chủ Skype quản lý các phiên đăng nhập, mật khẩu…
  - Sau khi kết nối, các máy sẽ gọi VoIP trực tiếp cho nhau

Processes communicating

Process: program running within a host.

- within same host, two processes communicate using inter-process communication (defined by OS).
- processes in different hosts communicate by exchanging messages

Client process: process that initiates communication

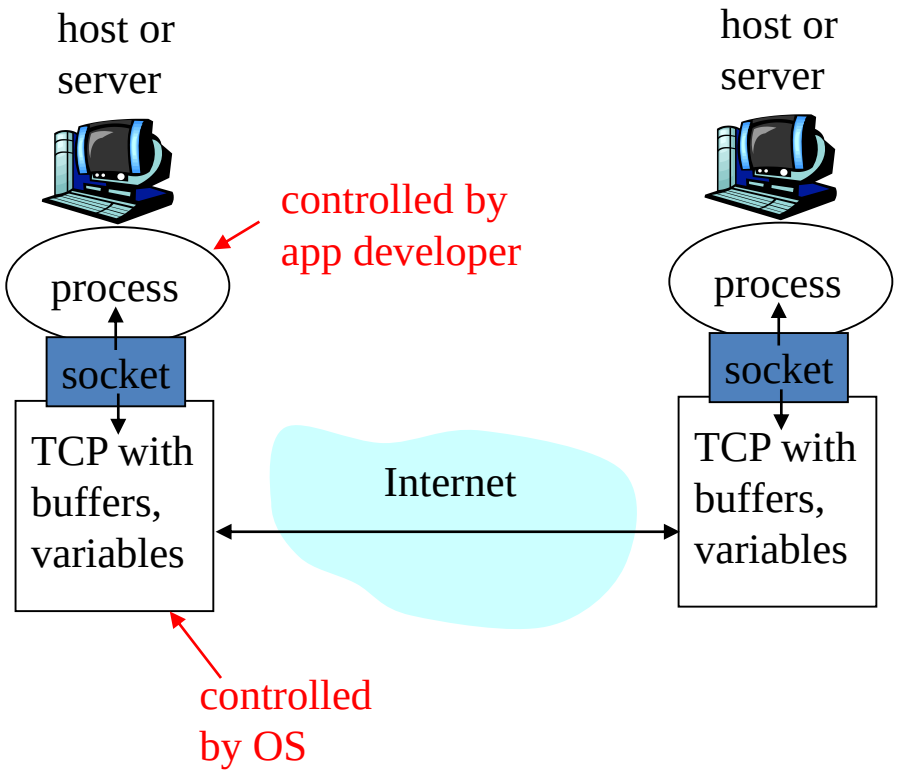
Server process: process that waits to be contacted

- Note: applications with P2P architectures have client processes & server processes

Sockets

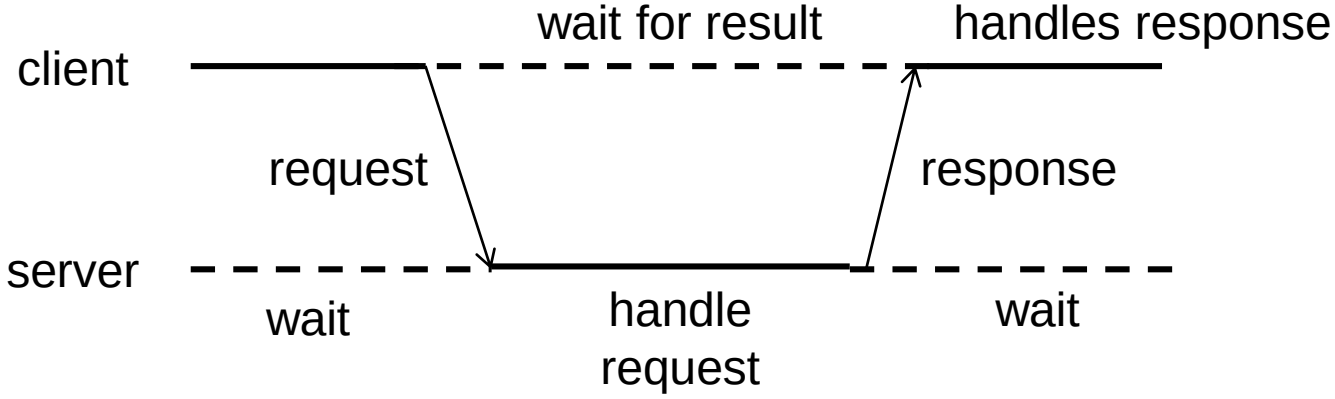
- process sends/receives messages to/from its socket
- Defined by
  - Port number
  - IP Address
  - TCP/UDP

  Socket address
- API: (1) choice of transport protocol; (2) ability to fix a few parameters



Processes communicating

- Client process: sends request
- Server process: replies response
- Typically: single server - multiple clients
- The server does not need to know anything about the client
- The client should always know something about the server
  - at least the socket address of the server



App-layer protocol defines

- Types of messages exchanged, eg, request & response messages
- Syntax of message types: what fields in messages & how fields are delineated
- Semantics of the fields, ie, meaning of information in fields
- Rules for when and how processes send & respond to messages

What transport service does an app need?

Data loss

- some apps (e.g., audio) can tolerate some loss
- other apps (e.g., file transfer, telnet) require 100% reliable data transfer

Timing

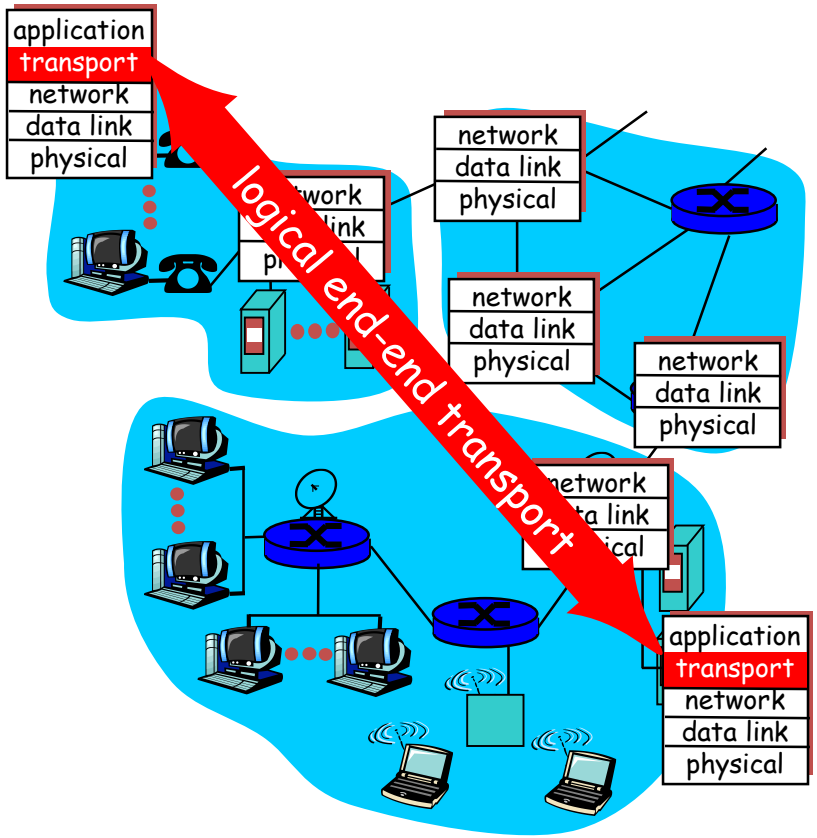
- some apps (e.g., Internet telephony, interactive games) require low delay to be "effective"

Bandwidth

- some apps (e.g., multimedia) require minimum amount of bandwidth to be "effective"
- other apps ("elastic apps") make use of whatever bandwidth they get

Transport services and protocols

- provide *logical communication* between app processes running on different hosts
- transport protocols run in end systems
  - send side: breaks app messages into segments, passes to network layer
  - rcv side: reassembles segments into messages, passes to app layer
- more than one transport protocol available to apps
  - Internet: TCP and UDP



Internet transport protocols services

TCP service:

- *reliable transport* between sending and receiving process
- *flow control:* sender won't overwhelm receiver
- *congestion control:* throttle sender when network overloaded
- *does not provide:* timing, minimum throughput guarantee, security
- *connection-oriented:* setup required between client and server processes

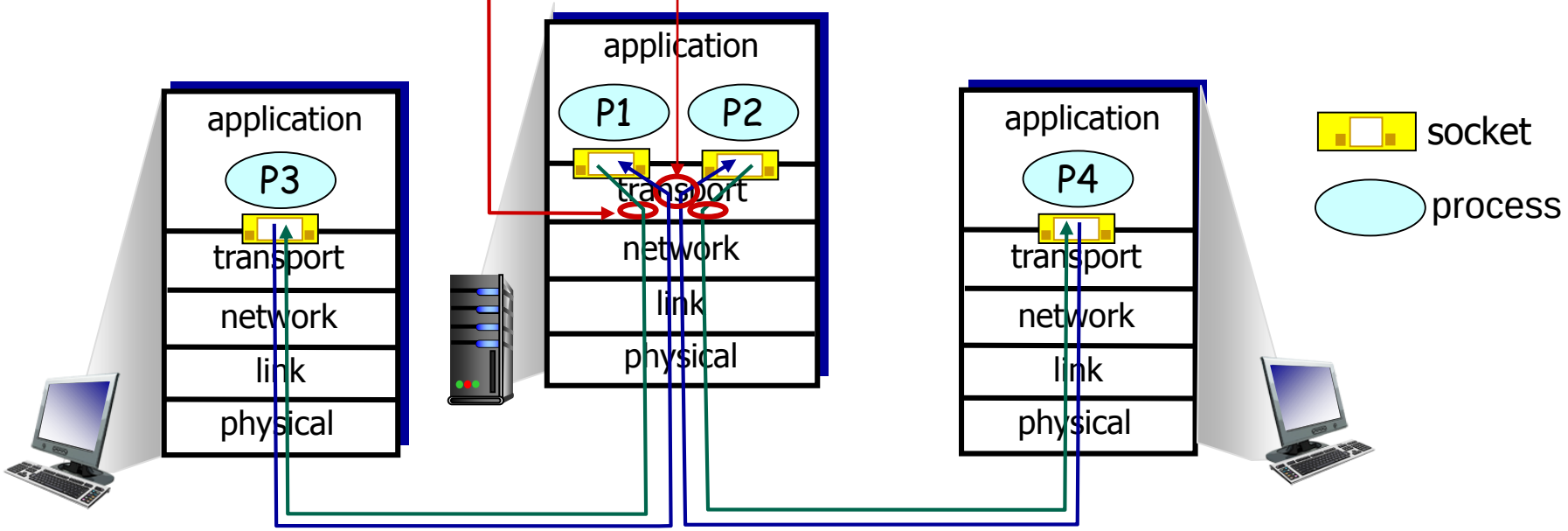
UDP service:

- *unreliable data transfer* between sending and receiving process
- *does not provide:* reliability, flow control, congestion control, timing, throughput guarantee, security, orconnection setup,

Multiplexing/demultiplexing

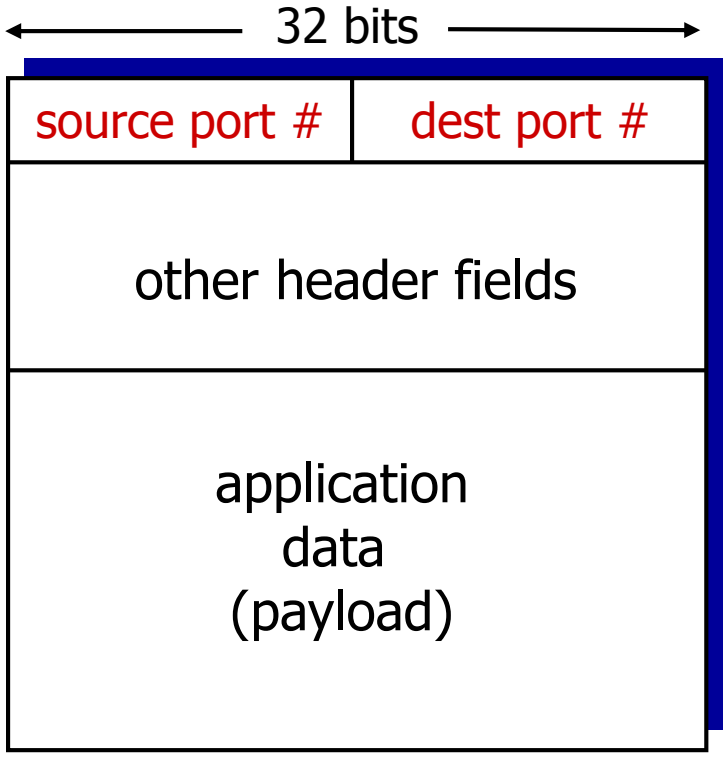
multiplexing at sender:
handle data from multiple sockets, add transport header (later used for demultiplexing)

demultiplexing at receiver:
use header info to deliver received segments to correct socket



How demultiplexing works

- host receives IP datagrams
  - each datagram has source IP address, destination IP address
  - each datagram carries one transport-layer segment
  - each segment has source, destination port number
- host uses *IP addresses & port numbers* to direct segment to appropriate socket



TCP/UDP segment format

UDP: User Datagram Protocol [RFC 768]

- "no frills," "bare bones" Internet transport protocol
- "best effort" service, UDP segments may be:
  - lost
  - delivered out of order to app
- *connectionless:*
  - no handshaking between UDP sender, receiver
  - each UDP segment handled independently of others

Why is there a UDP?

- no connection establishment (which can add delay)
- simple: no connection state at sender, receiver
- small segment header
- no congestion control: UDP can blast away as fast as desired

UDP demultiplexing

- Create sockets with port numbers:

```
mySocket = socket(AF_INET,
   SOCK_DGRAM, 0)
```

- UDP socket identified by two-tuple:

(dest IP address, dest port number)

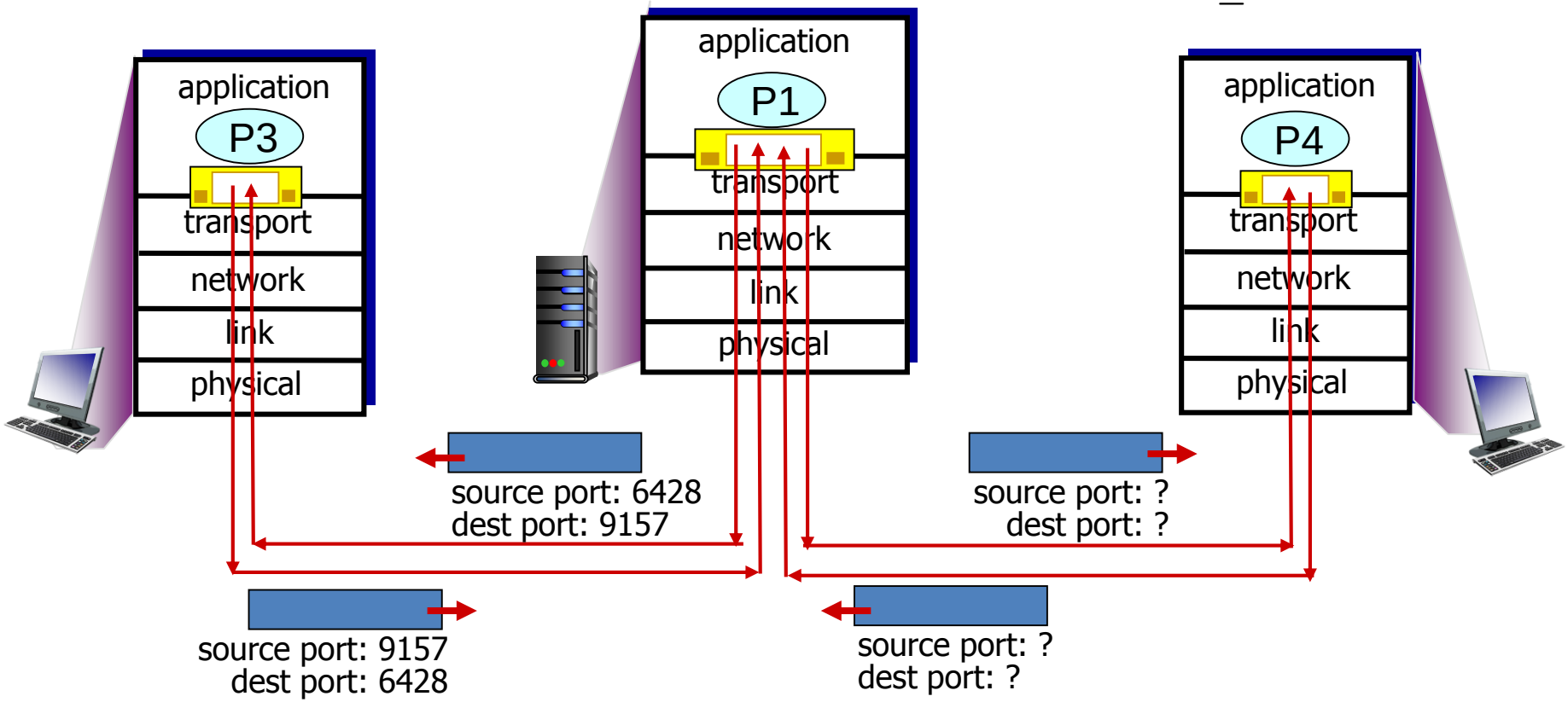
- When host receives UDP segment:
  - checks destination port number in segment
  - directs UDP segment to socket with that port number
- IP datagrams with different source IP addresses and/or source port numbers directed to same socket

UDP demux

```
mySocket =
 socket(AF_INET,
 SOCK_DGRAM, 0);
```

```
serverSocket =
  socket(AF_INET,
  SOCK_DGRAM, 0);
bind(…)
```

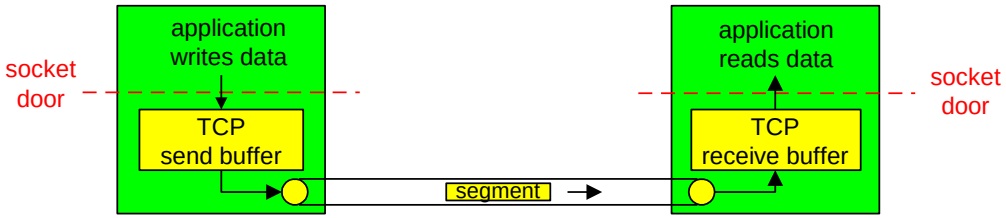
```
mySocket =
 socket(AF_INET,
 SOCK_DGRAM, 0);
```



TCP: Overview

RFCs: 793, 1122, 1323, 2018, 2581

- point-to-point:
  - one sender, one receiver
- reliable, in-order *byte steam:*
  - no “message boundaries”
- pipelined:
  - TCP congestion and flow control set window size
- *send & receive buffers*



- full duplex data:
  - bi-directional data flow in same connection
  - MSS: maximum segment size
- connection-oriented:
  - handshaking (exchange of control msgs) init’s sender, receiver state before data exchange
- flow controlled:
  - sender will not overwhelm receiver

TCP Connection Management: Setup

Recall: TCP sender, receiver establish "connection" before exchanging data segments

- initialize TCP variables:
  - seq. #s
  - buffers, flow control info (e.g. **`RcvWindow`**)
- *client:* connection initiator
  - connect()
- *server:* contacted by client
  - accept()

Three way handshake:

Step 1: client host sends TCP SYN segment to server
- specifies initial seq #
- no data

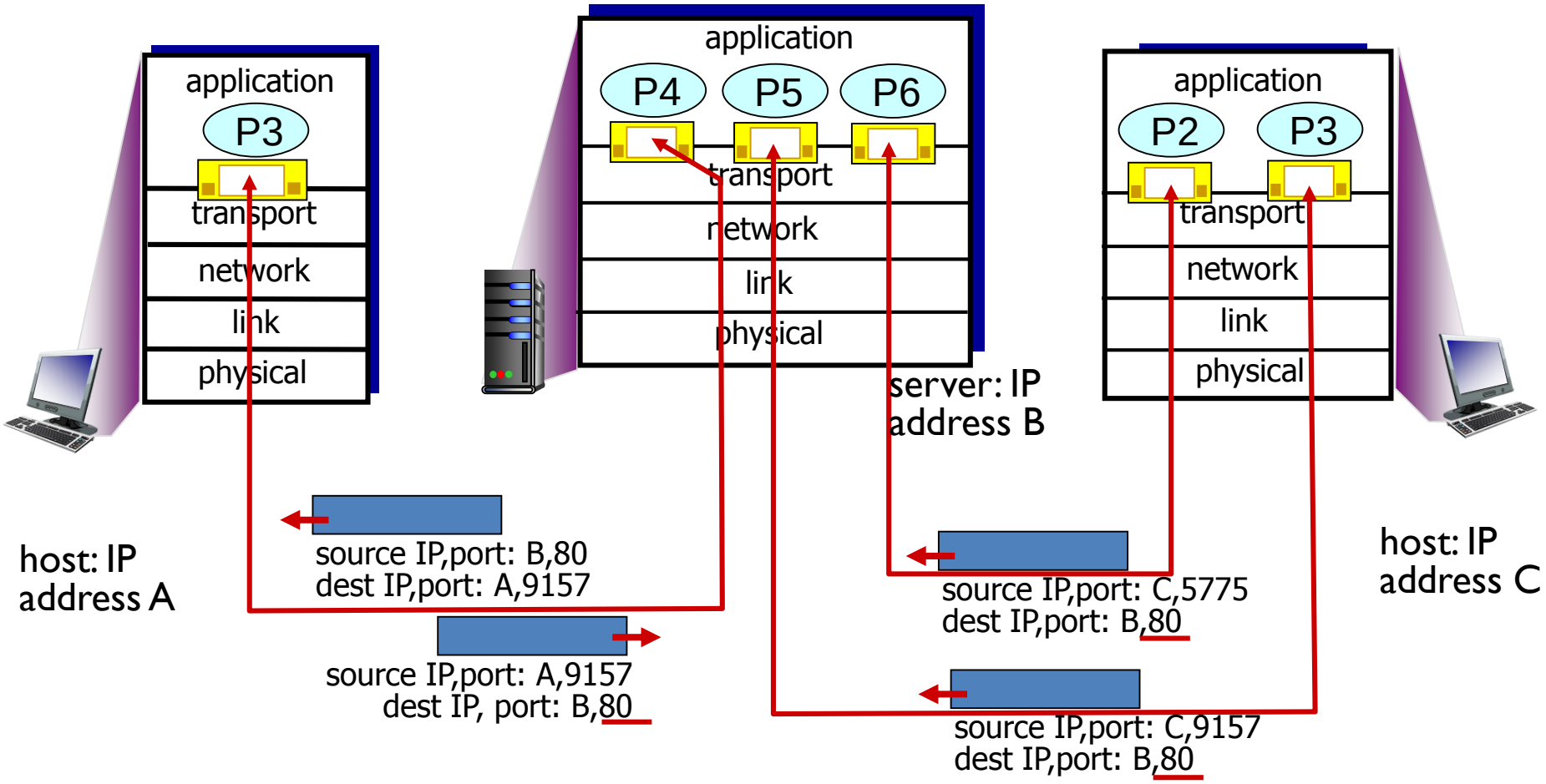
Step 2: server host receives SYN, replies with SYNACK segment
- server allocates buffers
- specifies server initial seq. #

Step 3: client receives SYNACK, replies with ACK segment, which may contain data

Connection-oriented demux

- TCP socket identified by 4-tuple:
  - source IP address
  - source port number
  - dest IP address
  - dest port number
- recv host uses all four values to direct segment to appropriate socket
- Server host may support many simultaneous TCP sockets:
  - each socket identified by its own 4-tuple
- Web servers have different sockets for each connecting client

Connection-oriented demux: example



three segments, all destined to IP address: B,
dest port: 80 are demultiplexed to *different* sockets

Connection-oriented demux: example