Chương 3
Mô hình Client-Server

Giảng viên: Nguyễn Đức Thái
thai@cse.hcmut.edu.vn

Nội dung

- Nhắc lại bài trước: tổng quan về lập trình mạng
  - Lập trình mạng trong Unix
  - Lập trình mạng trong Windows
  - Cấu trúc socket và địa chỉ của nó

- Giới thiệu mô hình Client-Server
- Giới thiệu giao tiếp sockets
- Giải thuật cho các ứng dụng với mô hình Client-Server

Giới thiệu mô hình Client-Server

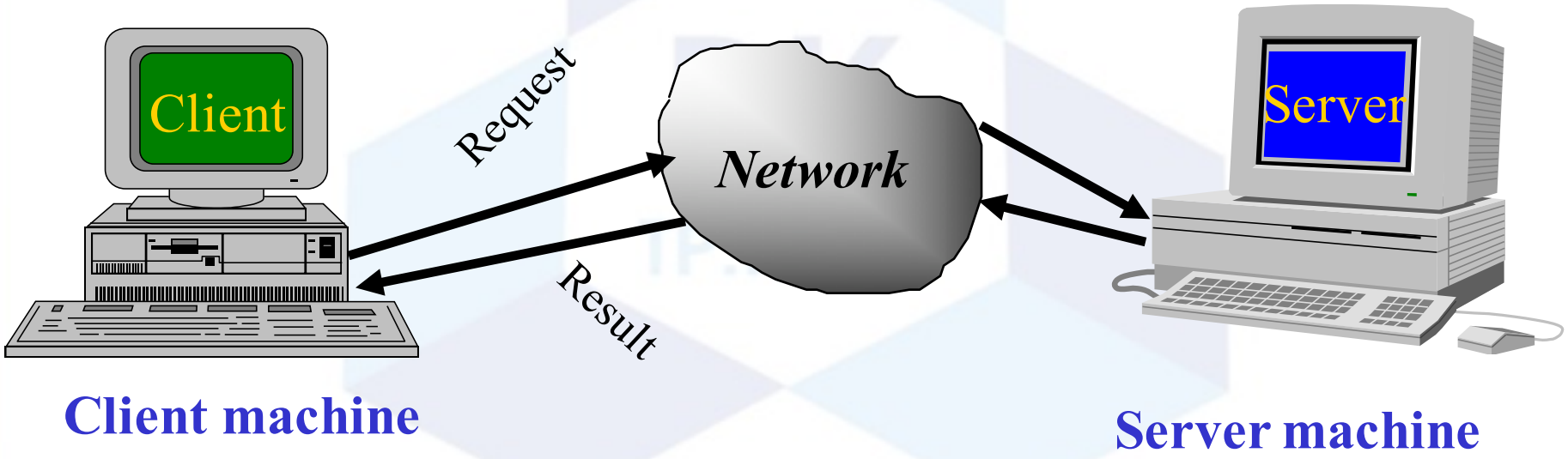
Kiến trúc Client-Server (1)

- Kiến trúc Client-Server là kiến trúc, trong đó mỗi máy tính hoặc quy trình (process) trong mạng có thể đóng vai trò hoặc là client hoặc là server
- Server nhận yêu cầu từ phía Client và trả kết quả lại cho Client

Kiến trúc Client-Server (2)

Thành phần:

- Clients
- Server
- Communication networks



Clients

- Clients là ứng dụng (application) chạy trên máy tính
- Clients dùng tài nguyên của servers:
  - Files
  - Devices
  - Processing power
- **Ví dụ**: E-mail client
  Là ứng dụng giúp chúng ta gởi và nhận mails

Clients là ứng dụng

Clients không nhất thiết là người

Servers

- Servers là những computer hoặc những process có chức năng quản lý tài nguyên mạng (network resource):
  - Disk drivers (file servers)
  - Printers (print servers)
  - Network traffic (network servers)

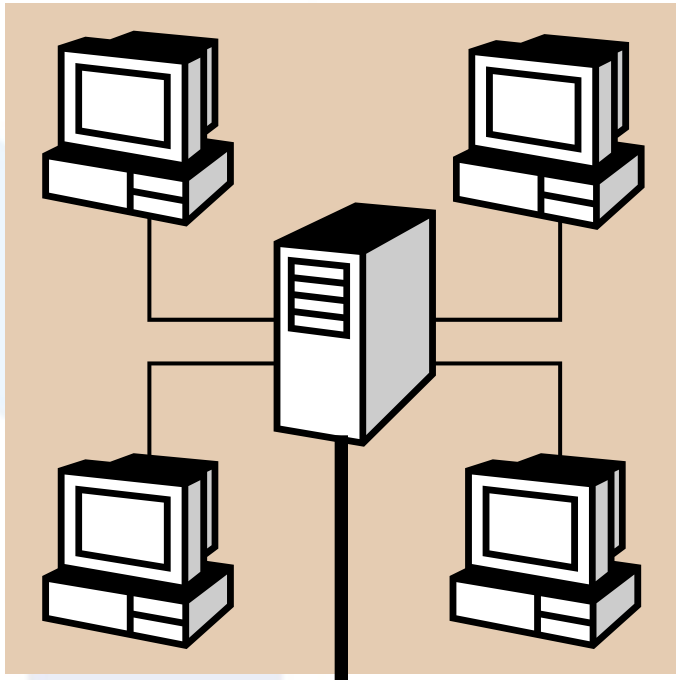
Servers Manage Resources

- **Ví dụ**: Database server

  Là hệ thống giúp chúng ta xử lý các câu truy vấn

Communication Networks

Networks kết nối
Clients and Servers



Mô hình Client-Server (1)

- Chương trình ứng dụng Client chạy trên *máy địa phương* (local), yêu cầu dịch vụ từ chương trình ứng dụng khác, là sever, chạy trên *máy khác ở xa*
- *Tổng quát*, server cung cấp dịch vụ cho nhiều client, không chỉ riêng một vài client
- Chương trình client được chạy *khi cần thiết*, trong khi chương trình server cần phải *chạy liên tục*
- Client *mở kênh giao tiếp*, đến địa chỉ IP và port của server
- Request-response có thể được lặp lại *vài lần*, số lần này là hữu hạn
- Client sau đó đóng kênh giao tiếp

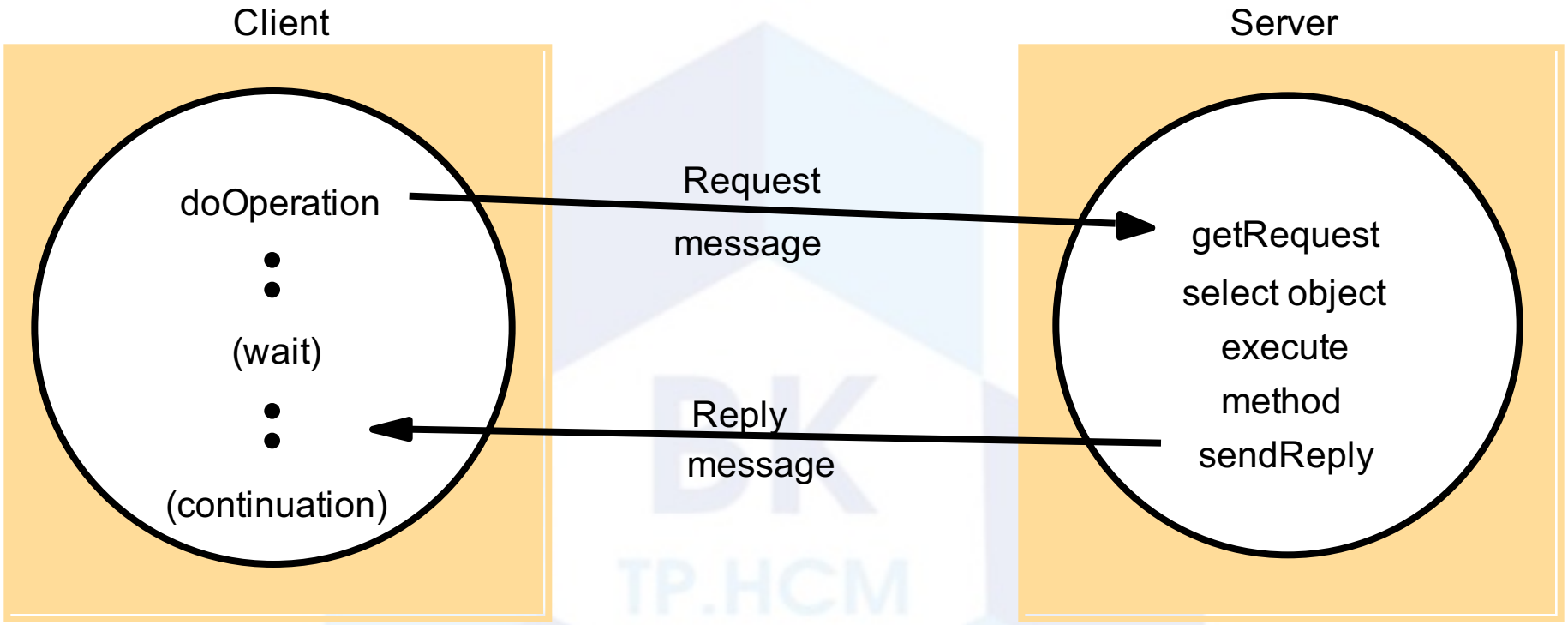
Mô hình Client-Server (2)

- Server *chờ đợi* yêu cầu đến từ client, nhưng không bao giờ chủ động cung cấp dịch vụ nếu không được yêu cầu
- Server *chạy liên tục*, nếu không có sự cố gì xảy ra
- Vấn đề concurrency *phía client*: hai hoặc nhiều clients chạy cùng lúc trên 1 máy, hoặc là một client khởi động, chạy và kết thúc trước khi client khác có thể chạy
- Vấn đề concurrency *phía server*: server chỉ xử lý một yêu cầu tại một thời điểm nhất định, trong khi concurrent server có thể xử lý nhiều yêu cầu tại cùng thời điểm

Giao tiếp Client-Server (1)

- Client "sometimes on"
  - Gởi yêu cầu đến server
  - Ví dụ: web browser
  - Không giao tiếp trực tiếp với client khác
  - Cần phải biết địa chỉ server mới có thể kết nối được
- Server "always on"
  - Đáp ứng những yêu cầu của clients
  - Ví dụ: Web server
  - Không chủ động kết nối với client, mà chờ client gởi request
  - Cần phải có địa chỉ cố định

Giao tiếp Client-Server (2)



So sánh: Client-Server và Peer-to-Peer

- Mạng client-server có chứa nhiều clients kết nối vào một server
- File server trong mạng client-server là một máy tính có công suất và tốc độ cao với dung lượng đĩa cứng lớn
- Ngược lại, mạng peer-to-peer chứa 2 hoặc nhiều máy tính, đóng góp tài nguyên cá nhân, như ổ đĩa, CD-ROM và máy in, v.v.
- Những tài nguyên này có thể được dùng bởi tất cả các máy tính trong mạng, khi 2 trong số chúng giao tiếp nhau trong session.
- Trong mạng peer-to-peer, mỗi máy tính đóng cả 2 vai trò client và server nghĩa là tất cả máy tính trong mạng đều ngang bằng
- Điểm mạnh của mạng peer-to-peer là dễ kiểm soát không cần đòi hỏi thực thể nào khác và không bị trễ (delay) vì không định tuyến qua server.
- Tuy nhiên, có thể xảy ra đụng độ session cao hơn, so với client-server

Giao tiếp sockets

Giao thức TCP và UDP

- TCP (Transmission Control Protocol) là một giao thức hướng kết nối, hỗ trợ việc điều tiết dòng dữ liệu giữa hai máy tính một cách đáng tin cậy

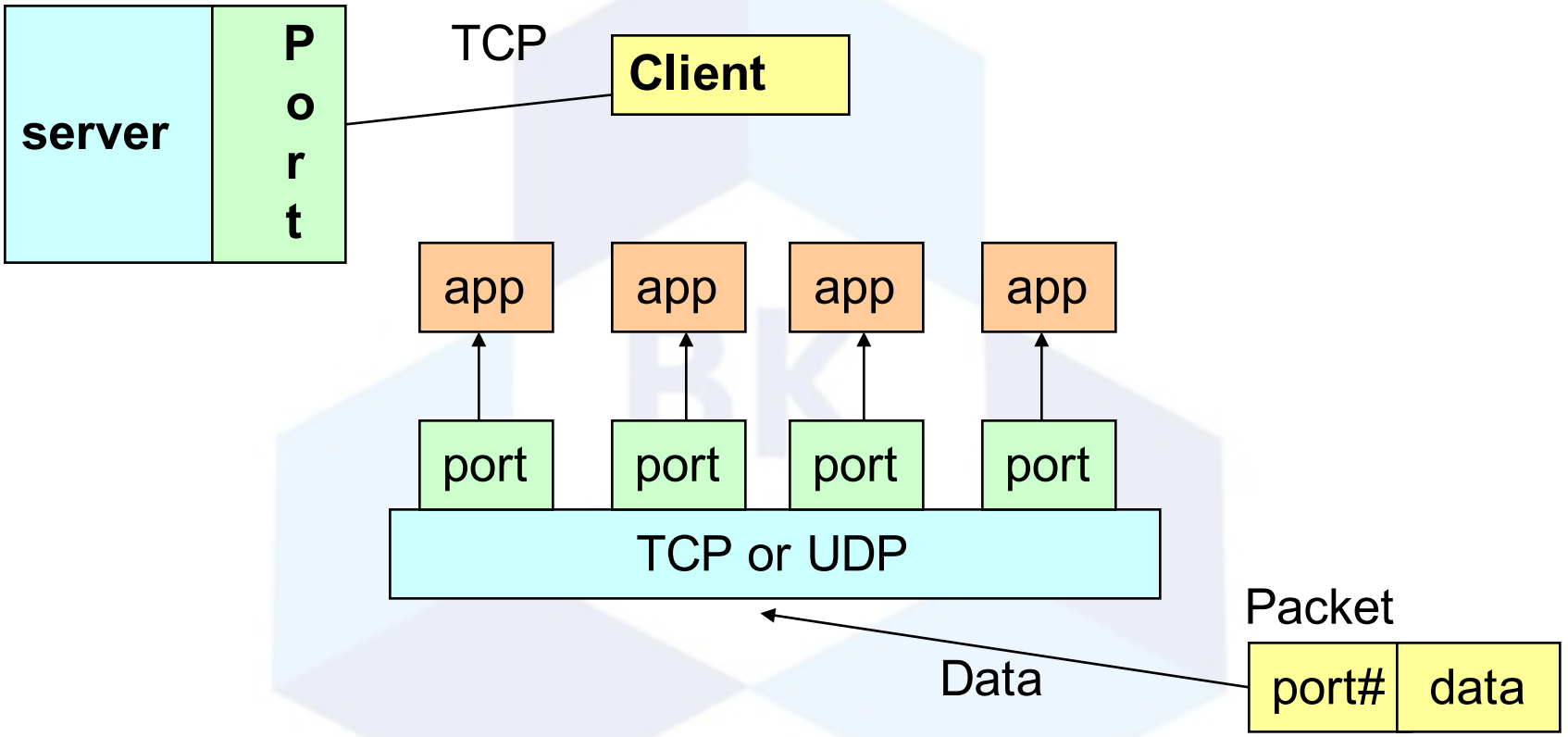
  Ví dụ: những ứng dụng dùng HTTP, FTP, Telnet, …

- UDP (User Datagram Protocol) là một giao thức, gởi packets dữ liệu độc lập, được gọi là *datagrams*, từ một máy tính đến máy tính khác mà không biết chắc có đến đích hay không

  Ví dụ: những ứng dụng dùng PING, TFTP, …

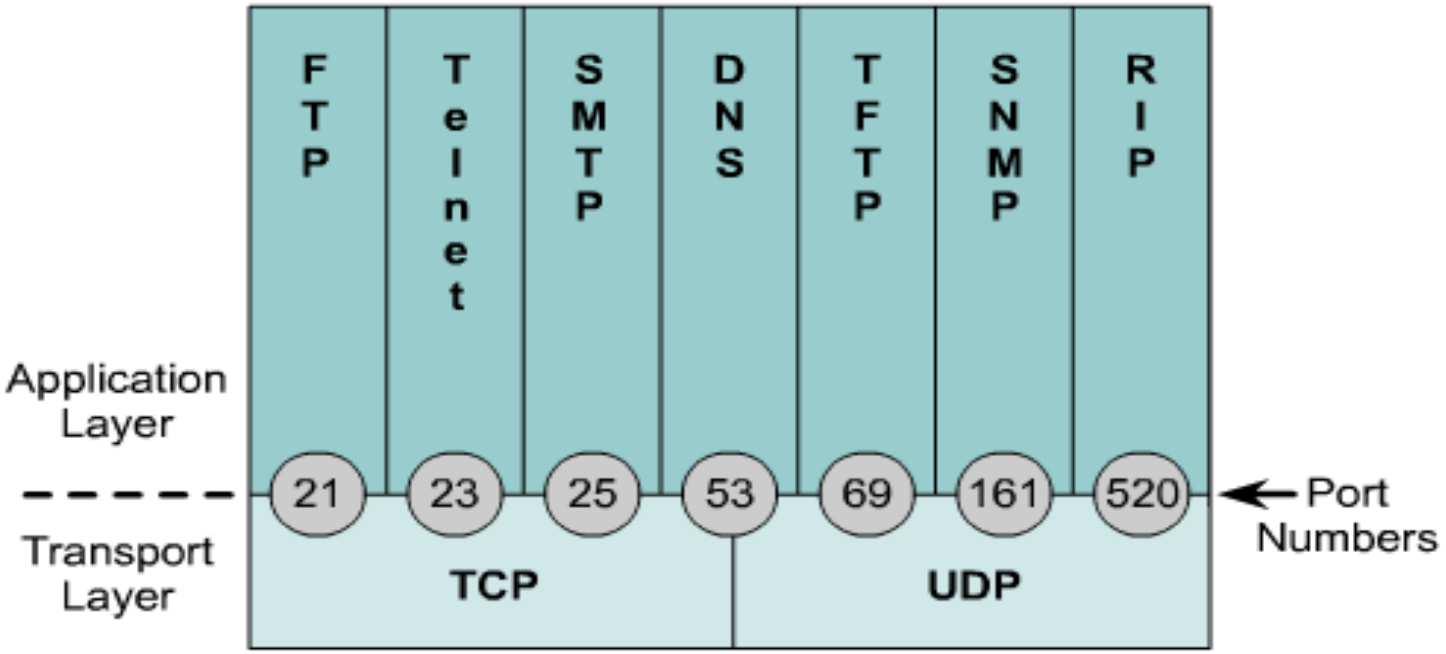
Ports

- Giao thức TCP và UDP dùng ports để map dữ liệu đến, vào một process đang chạy trên một máy tính



Ports

- Port được biểu diễn bằng một số nguyên có giá trị tối đa là 65,535 (16 bits)
- Những process của người dùng thông thường sử dụng ports có giá trị từ 1024 trở lên



Sockets

- Sockets cung cấp giao diện cho lập trình mạng ở tầng vận chuyển
- Giao tiếp mạng dùng sockets tương tự như thực hiện file I/O
  - Thực tế, socket handle được thực hiện giống như file handle
  - Streams dùng trong file I/O operation cũng có thể được áp dụng cho socket-based I/O
- Giao tiếp dựa trên socket không phụ thuộc vào ngôn ngữ lập trình
  - Có nghĩa là, một chương trình socket được viết trong Java có thể giao tiếp với một chương trình viết trong Java hoặc non-Java

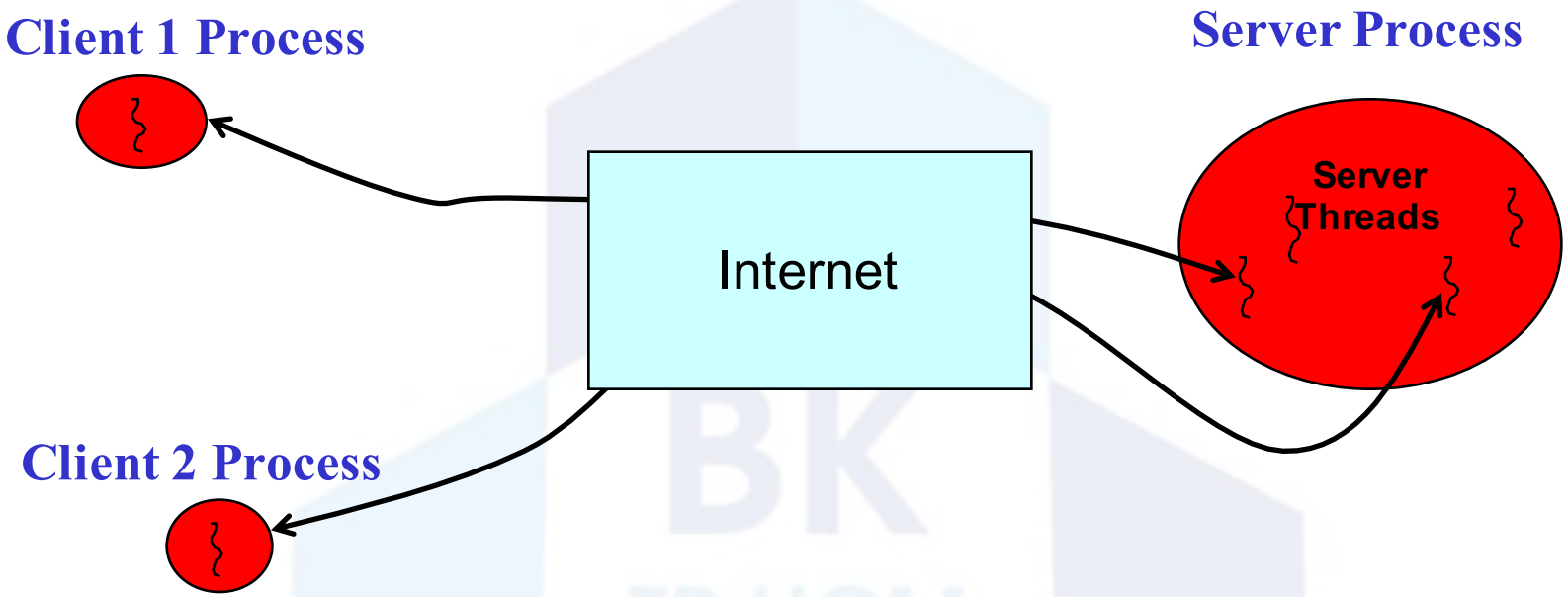
Giao tiếp Socket (1)

- Socket của chương trình server được gán một port nhất định.
- Server lắng nghe và chờ đợi một socket của client để thiết lập kết nối.

Giao tiếp Socket (2)

- Server tiếp nhận kết nối từ phía client. Sau đó server tạo một socket mới, được gán vào port khác.
- Nó cần một socket mới và port mới để đáp ứng client vừa được thiết lập connection, trong khi vẫn dùng socket ban đầu để lắng nghe những connection requests mới

Multithread Server: Phục vụ nhiều clients

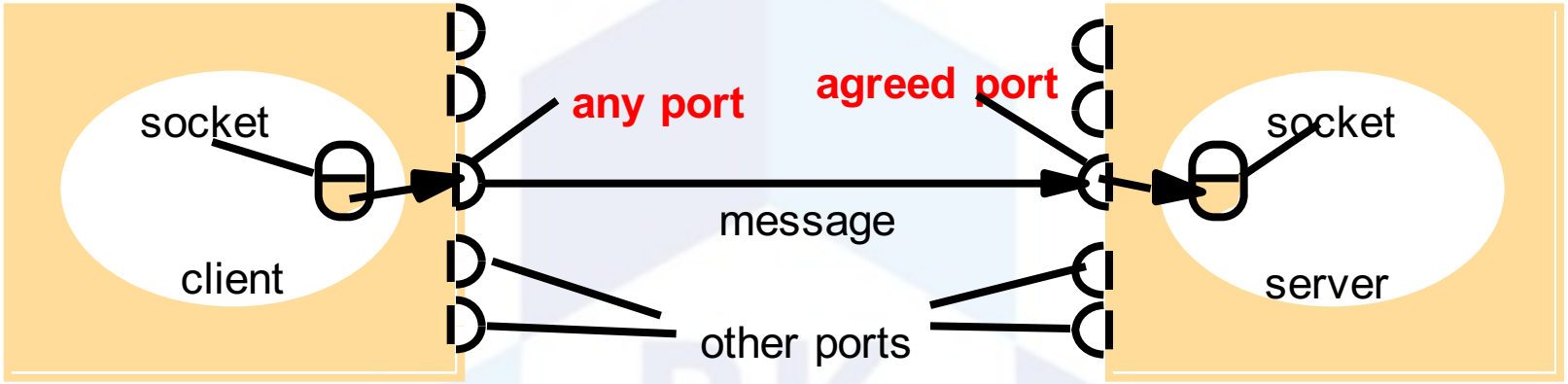


Giao tiếp Socket (3)

- Một socket là một *điểm cuối* (endpoint) của giao *tiếp hai chiều* giữa hai programs chạy trong network
- Một socket được *gán cho một port* để tầng vận chuyển có thể nhận ra (identify) được đích đến là ứng dụng nào mà dữ liệu được gởi đến

- Gói java.net package cung cấp 2 classes:
  - Socket – dùng cho client
  - ServerSocket – dùng cho server

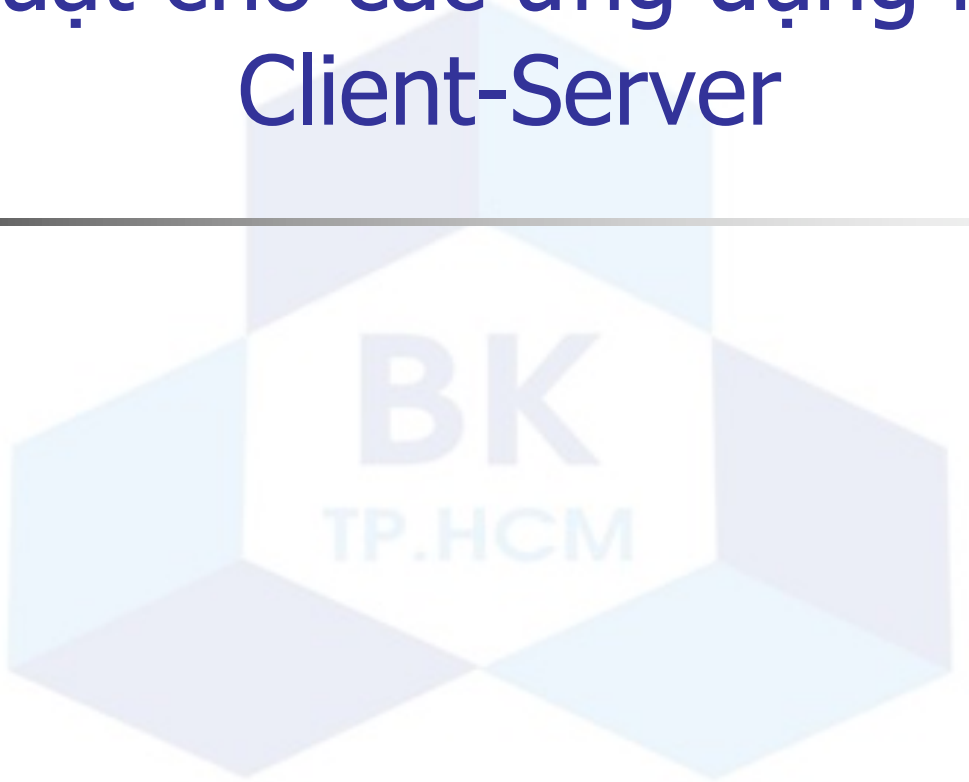
Giao tiếp Socket (4)



Internet address = 138.37.94.248

Internet address = 138.37.88.249

Giải thuật cho các ứng dụng mô hình Client-Server

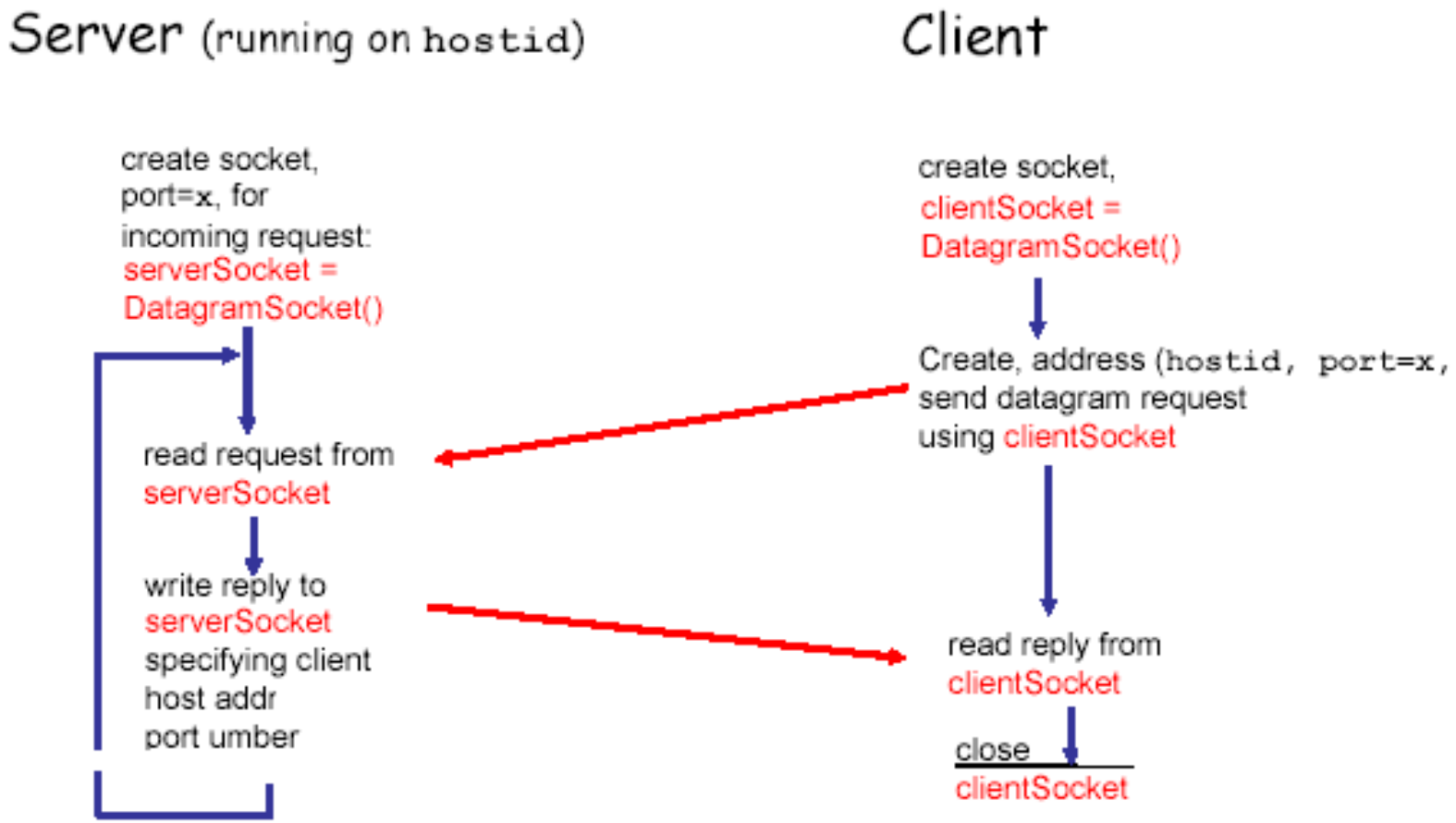


Ứng dụng Client-Server với UDP (1)

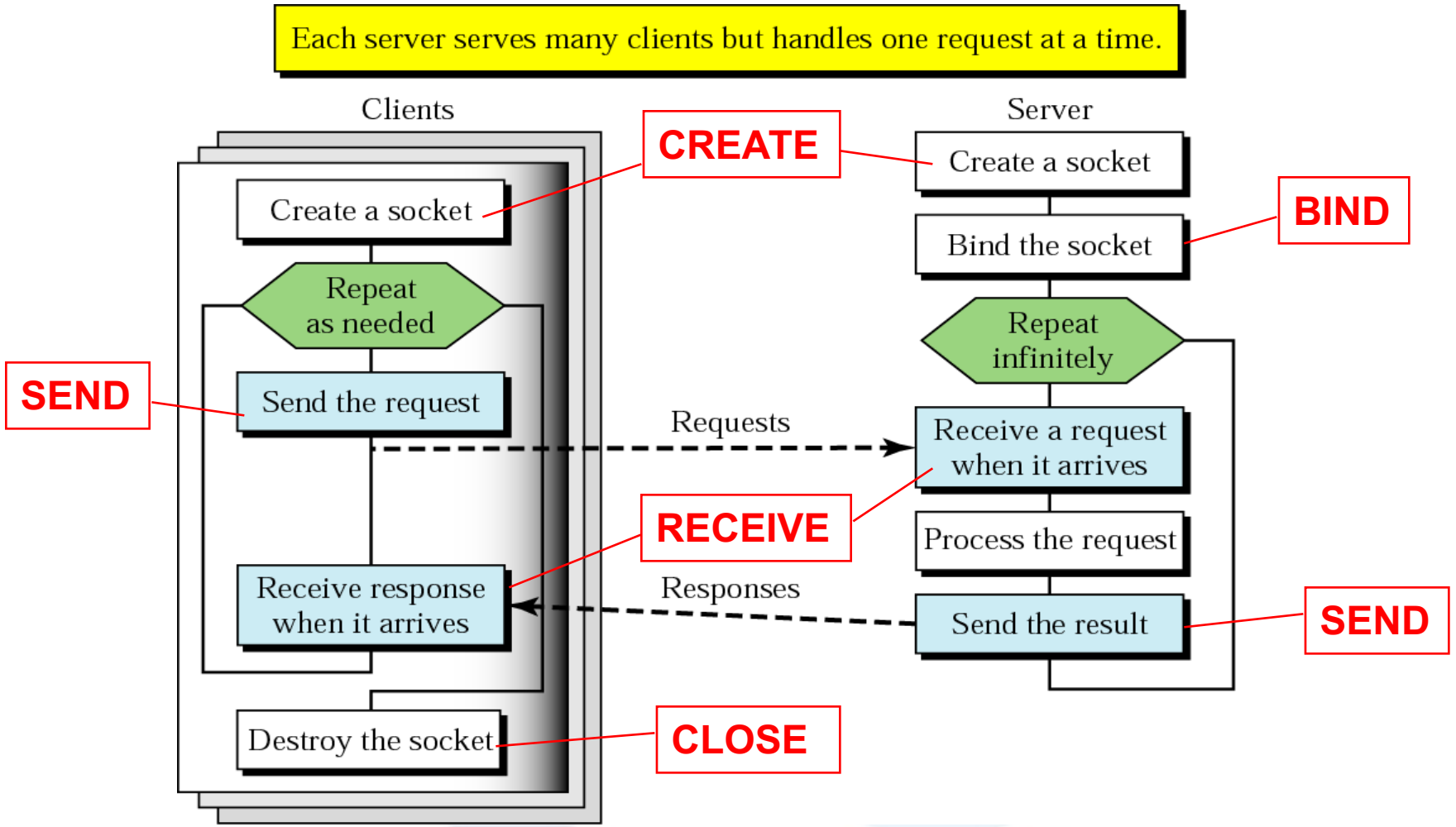
- Client operations
  - Nhận biết địa chỉ IP của server và port
  - Tạo UDP socket
  - Gởi/nhận dữ liệu
  - Đóng socket

- Server operations
  - Tạo socket và đăng ký (register) socket với hệ thống
  - Nhận thông điệp từ client và trả lời cho client

Ứng dụng Client-Server với UDP (2)



Ứng dụng Client-Server với UDP (3)



Ứng dụng Client-Server với TCP (1)

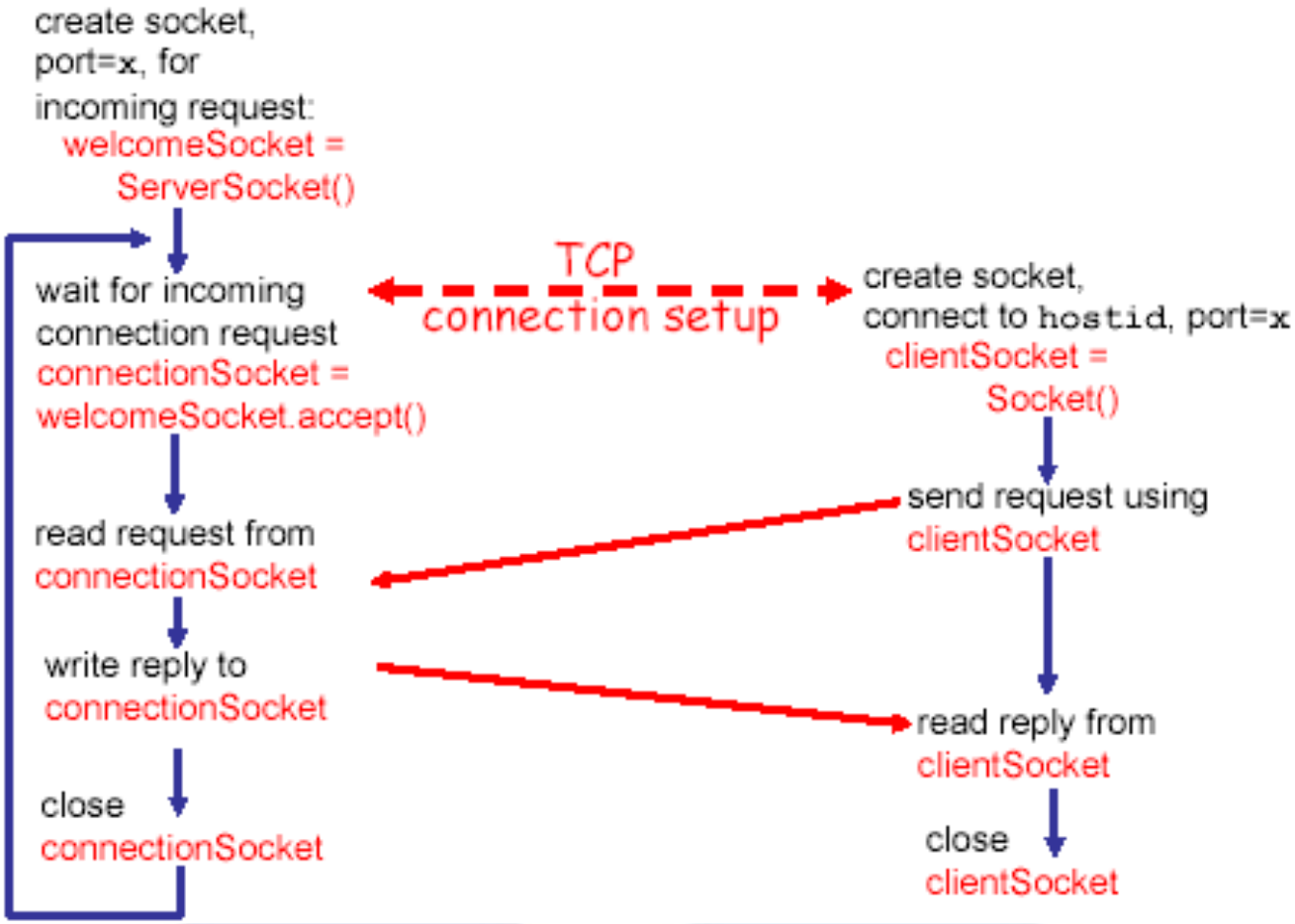
- Client operations
  - Nhận biết địa chỉ IP của server và port
  - Tạo UDP socket
  - Setup connection to server
  - Gởi/nhận dữ liệu
  - Đóng connection

- Server operations
  - Tạo socket và đăng ký (register) socket với hệ thống
  - Lắng nghe và chờ kết nối từ phía clients
  - Tiếp nhận (accept) connection
  - Gởi/nhận dữ liệu
  - Đóng connection

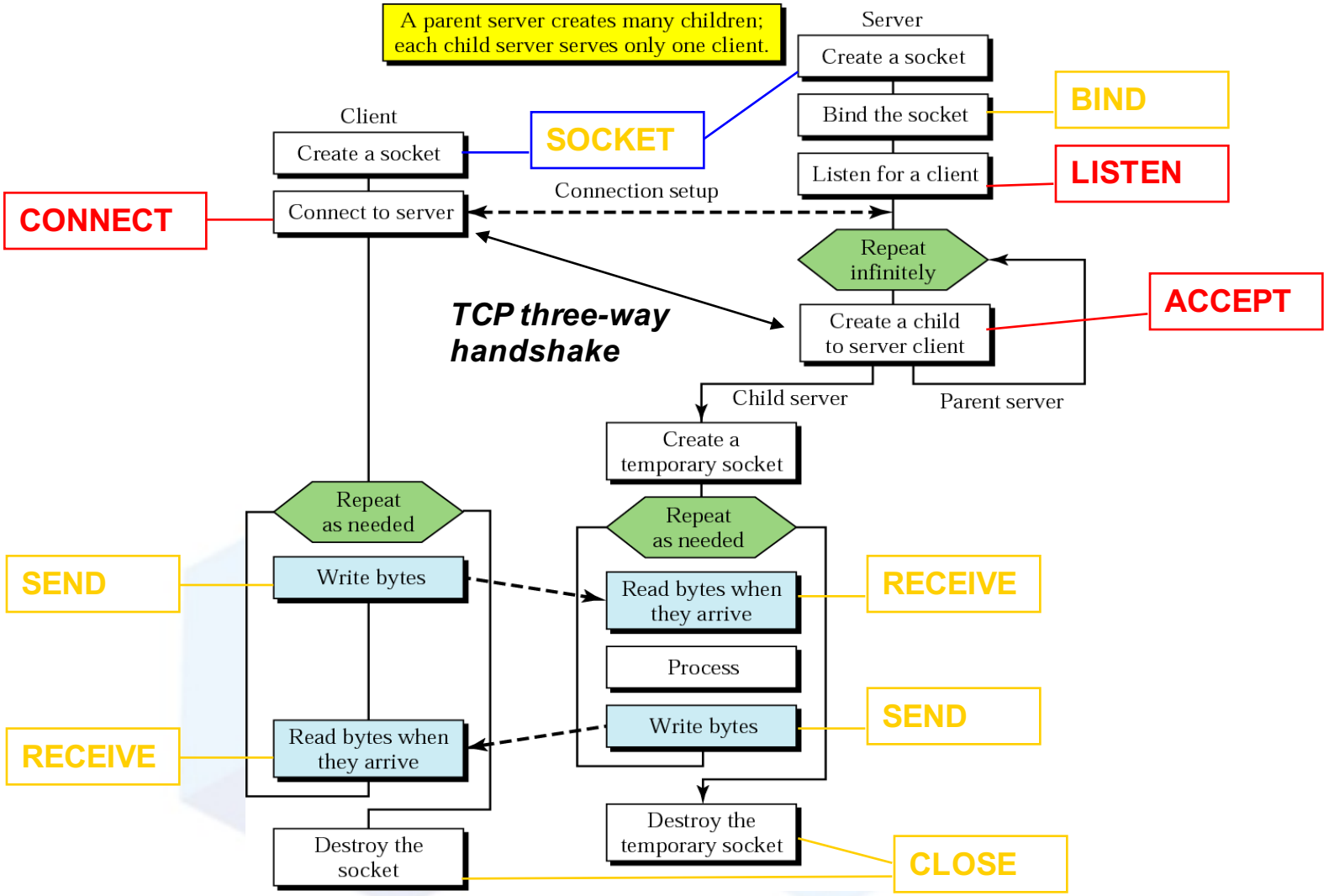
Ứng dụng Client-Server với TCP (2)

- Concurrent server operations
  - Tạo và đăng ký (register) socket với hệ thống
  - Lắng nghe và chờ kết nối từ phía clients
  - Tiếp nhận kết nối và tạo thread mới để giữ kết nối
  - Lắng nghe và chờ kết nối mới
  - Sự vận hành của thread:
    - Gởi/nhận dữ liệu thông qua kết nối
    - Đóng kết nối

Ứng dụng Client-Server với TCP (3)



Ứng dụng Client-Server với TCP (4)



Tài Liệu Tham Khảo

- [1] Bộ Slides cũ môn Lập Trình Mạng, Khoa KH&KTMT.