Môn học: Chuyên đề hệ thống phân tán

Tổng quan Hệ thống phân tán

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Hệ thống phân tán

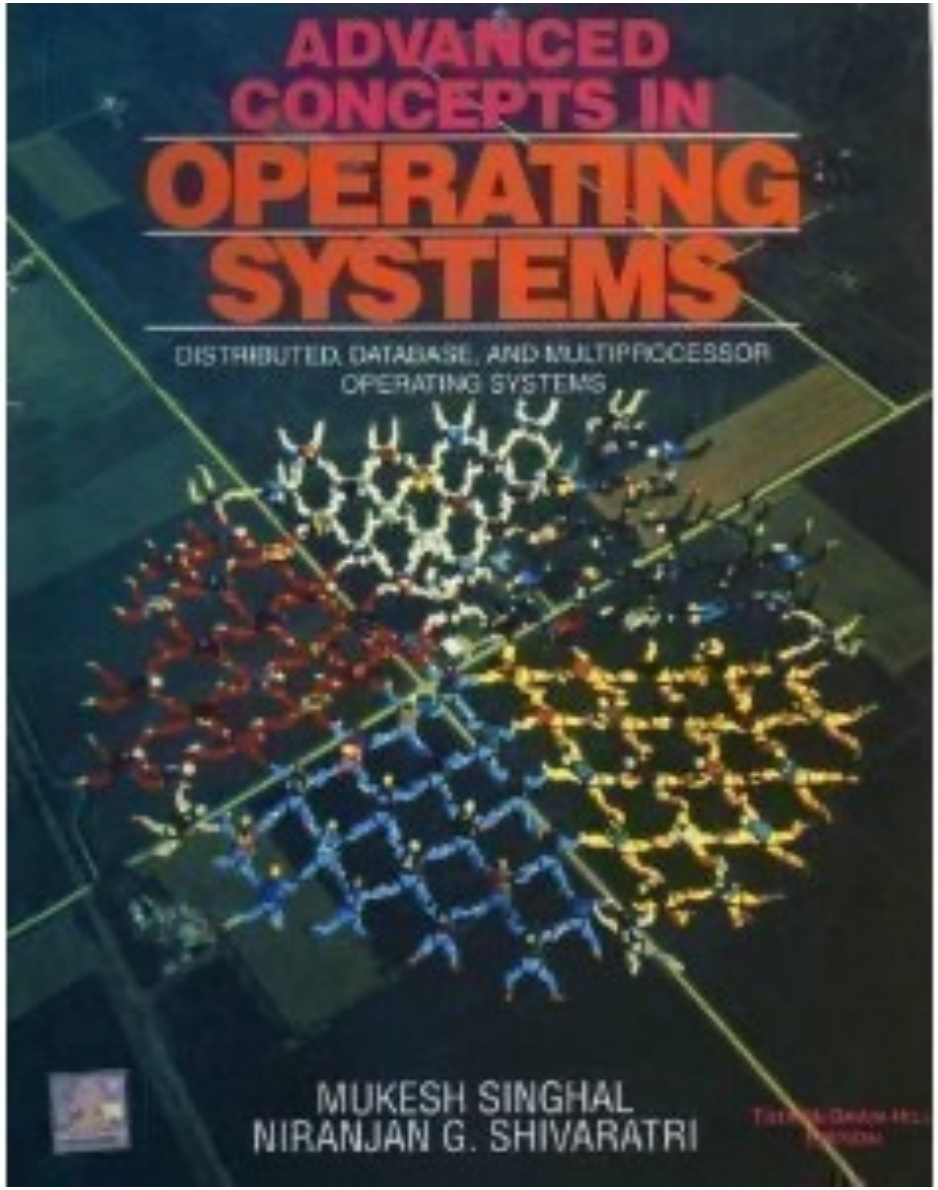
- Tài liệu:
  - Bài giảng, thảo luận trong lớp sẽ là sử dụng để kiểm tra cuối kì.
  - Khuyên bạn nên đọc các bài báo về các thuật toán cụ thể.
  - Các sách khác thảo luận về các thuật toán sẽ giúp ích bạn.

GV: Tran trung dung, **ttdung@fit.hcmus.edu.vn**

TG: TBA

Hệ thống phân tán

- Đồ án :
  - C, C++,Java


ADVANCED
CONCEPTS IN
OPERATING
SYSTEMS
DISTRIBUTED, DATABASE, AND MULTIPROCESSOR
OPERATING SYSTEMS
MUKESH SINGHAL
NIRANJAN G. SHIVARATRI

Tóm tắt môn học

Tóm tắt dự kiến. Có thể thay đổi tùy thời gian và tiến độ thực tế của học kì:

- Giới thiệu về hệ điều hành, quá trình liên lạc.
- Hệ điều hành phân tán
  - kiến trúc
  - Đồng bộ đồng hồ, thứ tự
  - Loại trừ lẫn nhau phân tán
  - Phát hiện deadlock phân tán
  - Các giao thức thỏa thuận
- Quản lý tài nguyên phân tán
  - Hệ thống tập tin phân tán
  - Chia sẻ bộ nhớ phân tán
  - Lập lịch phân tán

Tóm tắt...

- Khôi phục và khả năng chịu lỗi
- Điều khiển đồng thời/ an ninh hệ thống
  - Phụ thuộc vào thời gian thực tế

Các chủ đề mới sẽ được cung cấp tài liệu và thời gian đủ để các bạn cảm thụ được tốt.

Đánh giá

- 1 thi giữa kì: tại lớp 75 phút
- 1 thi cuối kì: 80-90 phút
- Bài tập tại lớp
- Đồ án lập trình:
  - 1 khởi động
  - 2 đồ án dựa trên thuật toán học trên lớp.

Điểm

- Giữa kì + cuối kì: 70%.
  - 35% mỗi lần thi
  - Nếu ko thi giữa kì thì các bài tập trên lớp chiếm 25%.
- Đồ án: 30%
- Bài trên lớp: ++10%

Kế hoạch

- Thi giữa kì dự kiến: tuần thứ 8
- Thi cuối kì: theo lịch chung

Đồ án lập trình

- Không chấp nhận copy bất kì một phần nào. Từ internet hay từ ai đó
- Nộp bài qua Moodle.
- Cần demo
- Bài có copy sẽ bi 0 điểm

Bài giảng trên lớp

- Thảo luận đồ án:
  - Sẽ có các thảo luận nhỏ ở lớp
  - Trình bày chi tiết trong seminar + QA trên moodle
- Lý thuyết:
  - Học toàn bộ ở lớp

Moodle

- Thông báo
- Sinh viên phải chịu trách nhiệm cập nhật thông tin trên moodle mỗi ngày
- Nếu Moodle ko hoạt động và sv ko cập nhật đc thông tin thì lỗi tại sv ?!

Gian lận

- Trung thực là yêu cầu tiên quyết
- Gian lận sẽ được chuyển lên cho Khoa.
- Gian lận bất kì bài tập hay đồ án, thi … sẽ bị điểm 0 toàn môn

Đồ án

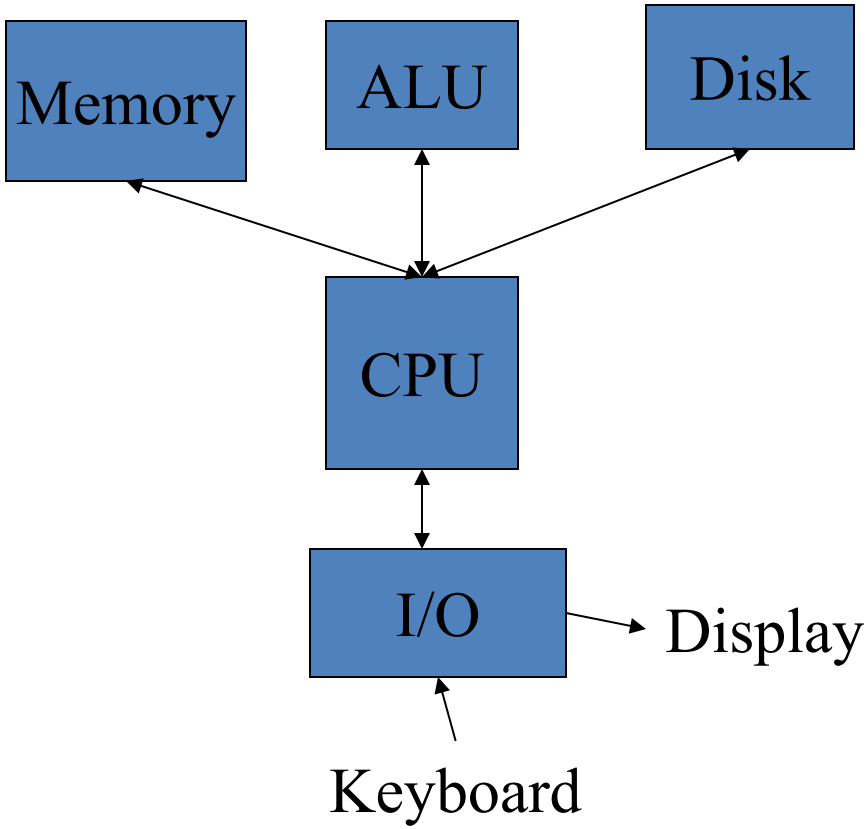
- Liên quan đến các bài tập như thứ tự sự kiện, phát hiện deadlock, cân bằng tải, tin nhắn, và thực hiện các thuật toán phân phối (ví dụ như lập kế hoạch, vv.)
- Hệ điều hành: Linux / Windows, C / C + + / Java. Lập trình mạng sẽ là cần thiết. Nhiều hệ thống sẽ được sử dụng.
- Chi tiết và thời hạn cụ thể sẽ được công bố trong các lớp học và moodle.
- Gợi ý: Tìm hiểu lập trình socket mạng và chủ đề, nếu bạn không biết. Thử các chương trình đơn giản như truyền tập tin, nói chuyện, vv

Bài tập về nhà

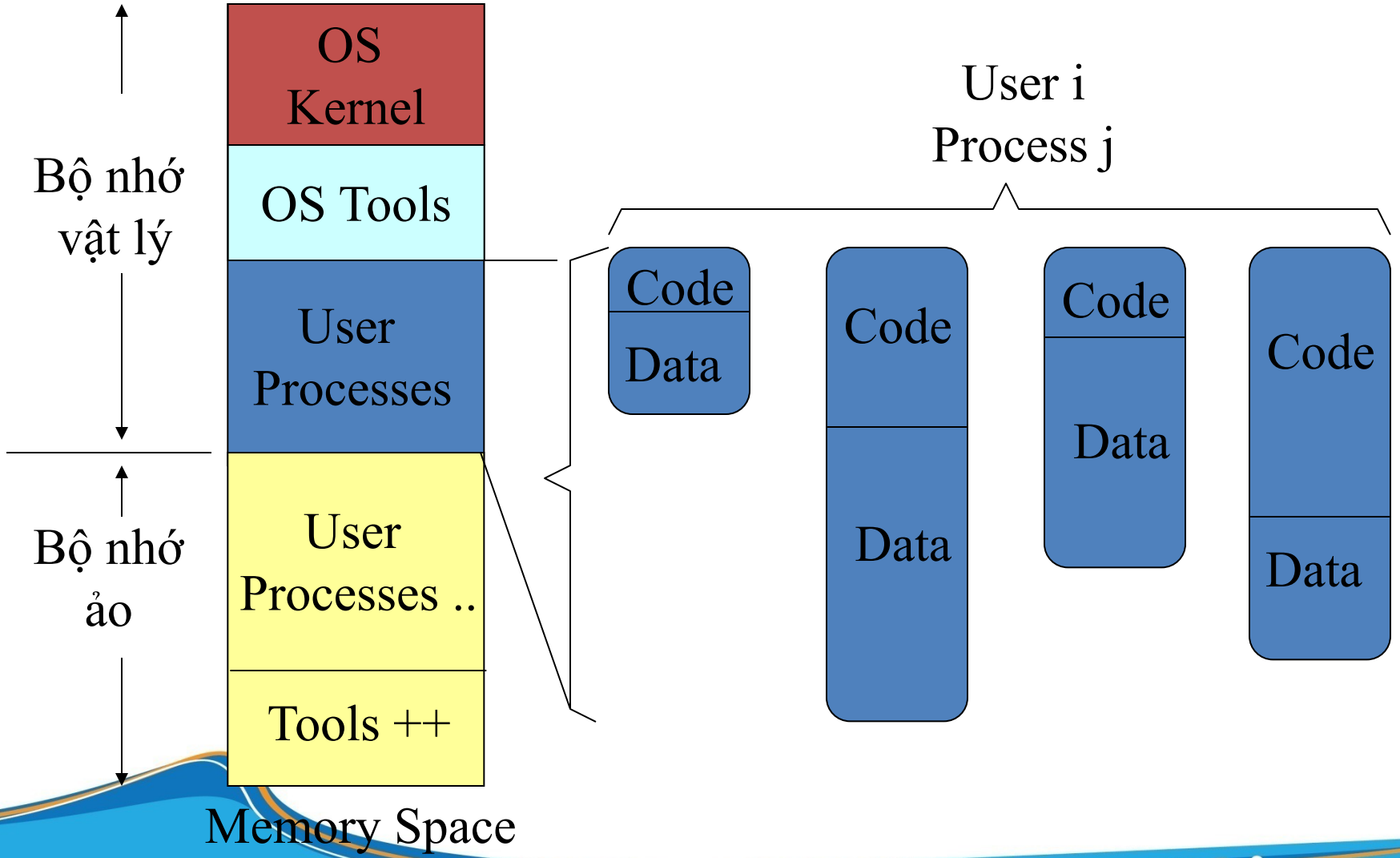
- Thông báo tại lớp, chủ yếu là để cộng điểm và bài mẫu ôn thi

Tổ chức máy tính cơ bản

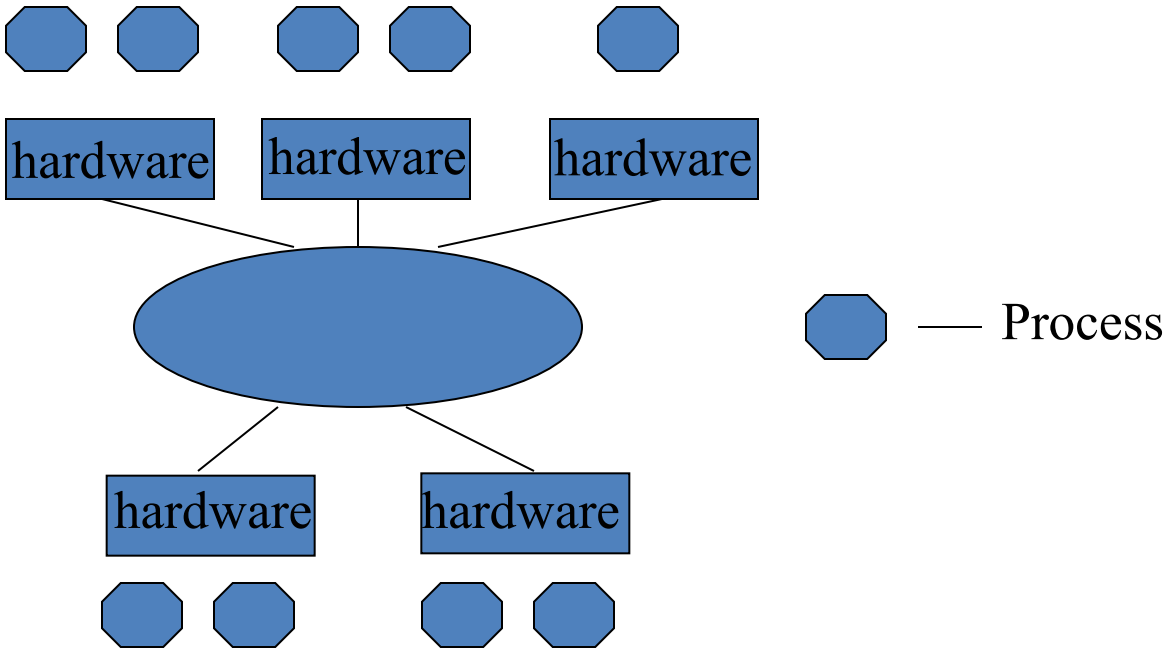
- Input Unit
- Output Unit
- CPU
- Memory
- ALU (Arithmetic & Logic Unit)
- Secondary Storage



Góc nhìn giản đơn về hđh



Hệ thống phân tán



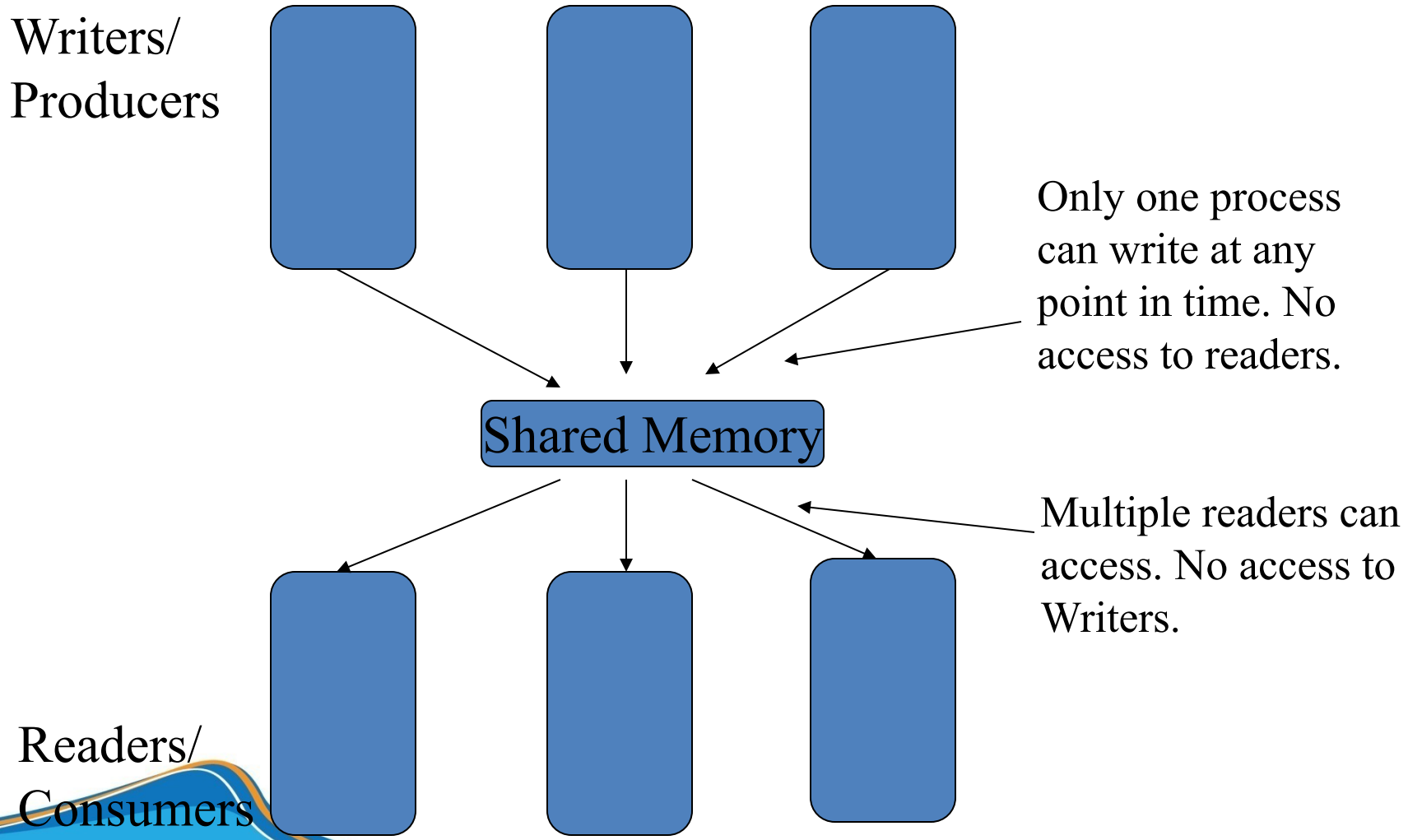
Giao tiếp giữa các tiến trình

- Cần cho trao đổi dữ liệu / thông điệp giữa các quá trình thuộc cùng nhóm và khác nhóm.
- Cơ chế:
  - Chia sẻ bộ nhớ: Chỉ định và sử dụng một số dữ liệu / bộ nhớ chia sẻ. Sử dụng bộ nhớ chia sẻ để trao đổi dữ liệu.
    - Đòi hỏi phải có cơ sở để kiểm soát quyền truy cập vào dữ liệu chia sẻ.
  - Truyền thông điệp: Sử dụng "cao hơn“ cấp cơ bản (chỉ để "gửi" và "nhận" dữ liệu.)
    - Đòi hỏi phải có hệ thống hỗ trợ cho việc gửi và nhận msg.
  - Cấu trúc ngôn ngữ hướng đối tượng
    - Đáp ứng yêu cầu
    - Tương tự như truyền thông điệp với phản hồi bắt buộc
    - Có thể được thực hiện bằng cách sử dụng bộ nhớ chia sẻ .

Ví dụ IPC

- Tính toán song song/phân tán: sắp xếp mãng thì chia sẻ bộ nhớ sẽ thích hợp hơn
- client-server: truyền thông điệp hoặc RPC có thể phù hợp hơn với
  - Chia sẻ bộ nhớ có thể là hữu ích, nhưng các chương trình rõ ràng hơn với truyền thông điệp
- RPC vs truyền thông điệp: nếu ko cần phản hồi ngay lập tức, truyền thông điệp đơn giản là đủ.

Chia sẻ bộ nhớ



Chia sẻ bộ nhớ: khả năng

- Locks (unlocks)
- Semaphores
- Monitors
- Serializers
- Path expressions

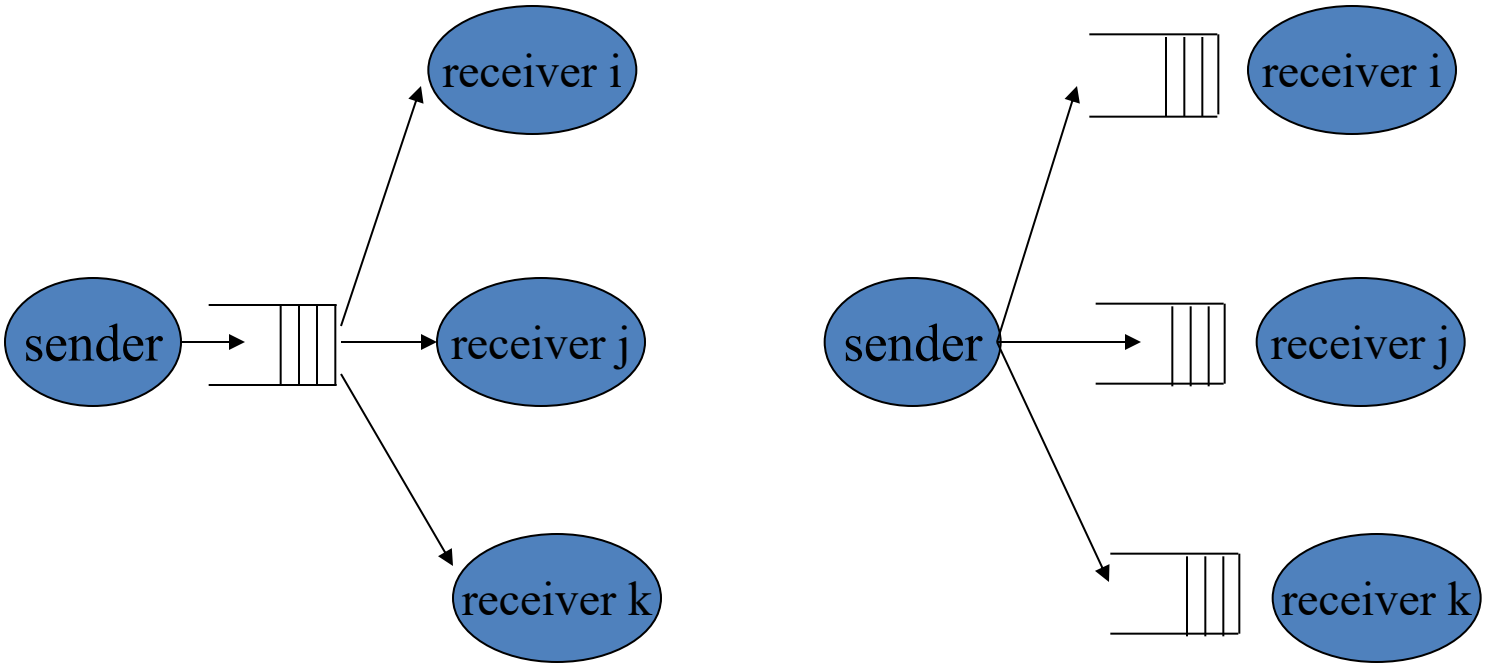
Truyền thông điệp

- Blocked gửi/nhận: Cả hai quá trình gửi và nhận được chặn cho đến khi được thông báo là hoàn tất. Đồng bộ
- Unblocked gửi/nhận: Cả gửi và người nhận không bị chờ đợi cho đến khi hoàn tất. Không đồng bộ
- Unblocked gửi/ Blocked nhận: người gửi không bị chặn. Nhận đợi đến khi nhận được tin nhắn hoàn tất
- Blocked Gửi / Unblocked Nhận: Hữu ích ?
- Có thể được thực hiện bằng cách sử dụng bộ nhớ chia sẻ. Truyền thông điệp: Một mô hình dễ dàng hơn cho con người.

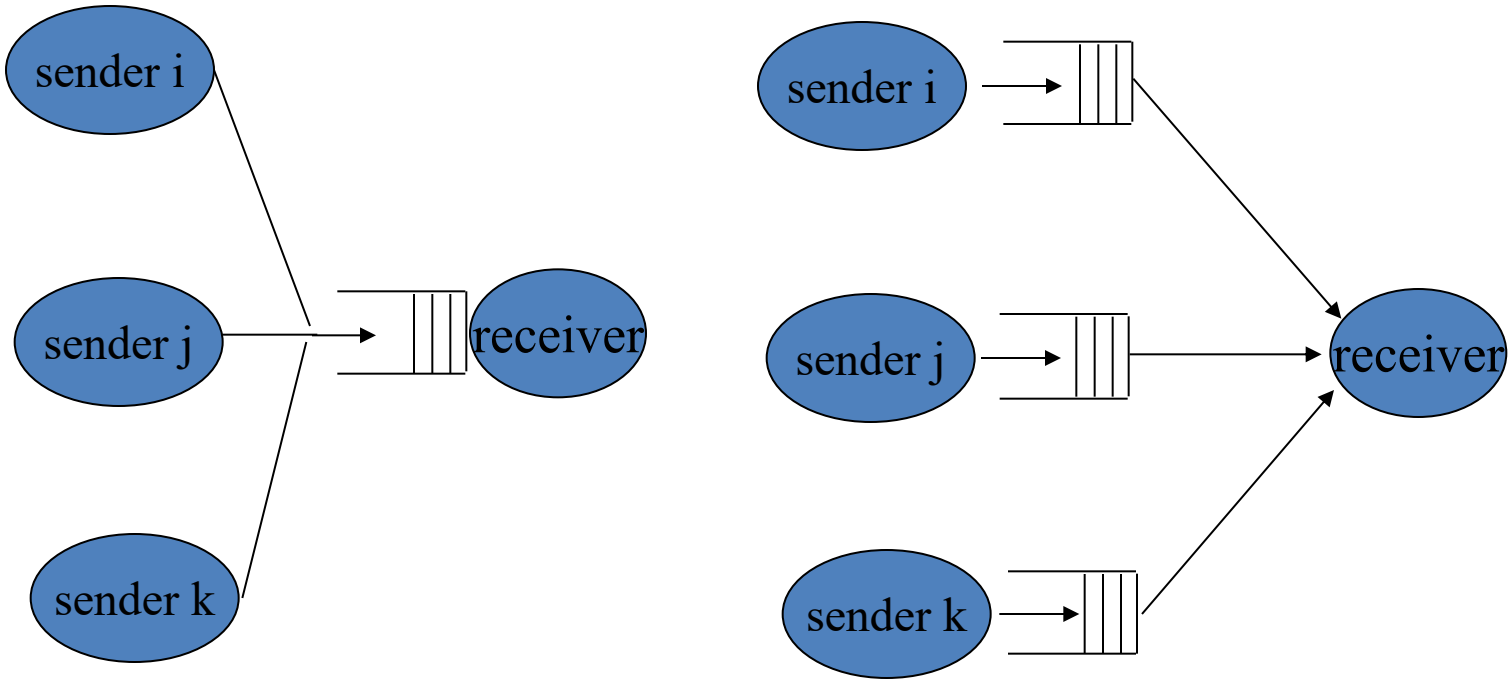
Un/blocked

- Blocked message
  - Dễ: hiểu, cài đặt và kiểm tra tính đúng đắn
  - Không mạnh mẽ, có thể không hiệu quả khi cả gửi và nhận cùng phải đợi
- Unblocked message
  - Hiệu quả hơn, không lãng phí bởi thời gian chờ đợi
  - Cần hàng đợi
  - Khó xác minh tính đúng đắn của chương trình

Message Passing: Possibilities



Message Passing: Possibilities...



Định danh - Naming

- Định danh trực tiếp - Direct Naming
  - Xác định chính xác id của từng thực thể.
  - Đơn giản nhưng mà không mạnh, vì mỗi bên gửi/nhận phải biết id cụ thể của đối tác, sẽ khó trong việc quản trị hệ thống id
  - Không phù hợp cho mô hình client-server tổng quát
- Định danh cổng - Port Naming
  - Bên nhận sử dụng một cổng để nhận tất cả msg, mô hình client-server.
  - Tốn chi phí phân loại msg, xác nhận
- Định danh toàn cục - Global Naming (mailbox)
  - Phù hợp với client-server, khó triển khai trên nền phân tán
  - Cấu trúc ngôn ngữ và việc xác nhận phức tạp
- Định danh gián tiếp - Indirect Naming
  - Sử dụng server định danh, e.g., ví dụ RPCs.

Giao tiếp tuần tự

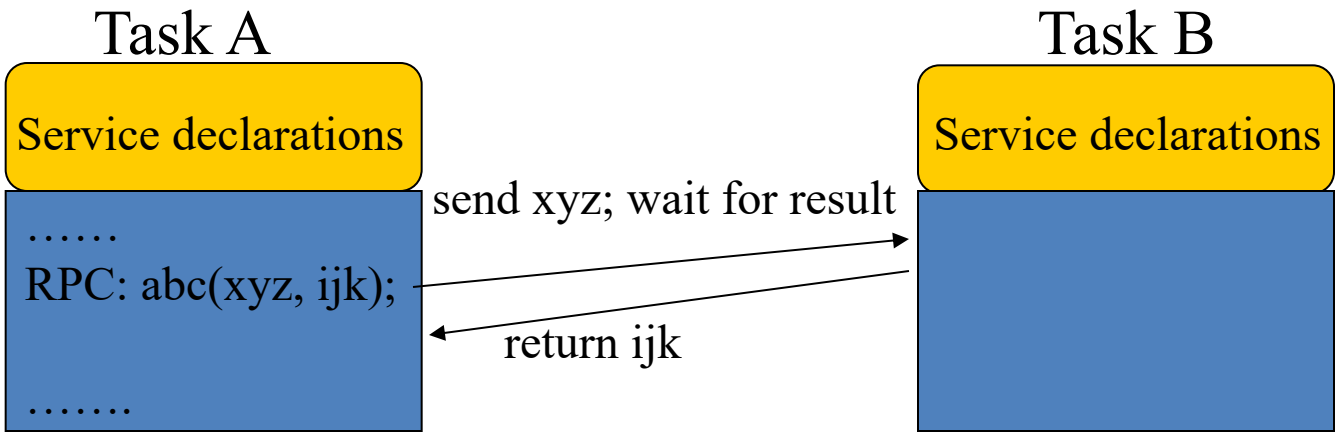
```
process reader-writer
    OKtoread, OKtowrite: integer (initially = value);
    busy: boolean (initially = 0);
     *[    busy = 0; writer?request() ->
                    busy := 1; writer!OKtowrite;
           □
           busy = 0; reader?request() ->
                    busy := 1; reader!OKtoread;
           □
           busy = 1; reader?readfin() -> busy := 0;
           □
           busy = 1; writer?writefn() -> busy := 0;
     ]
```

Giao tiếp tuần tự: hạn chế

- Yêu cầu chính xác tên của tiếp trình trong lệnh I/O.
- Không có buffer msg, input/output bị blocked → làm chậm trễ và không hiệu quả

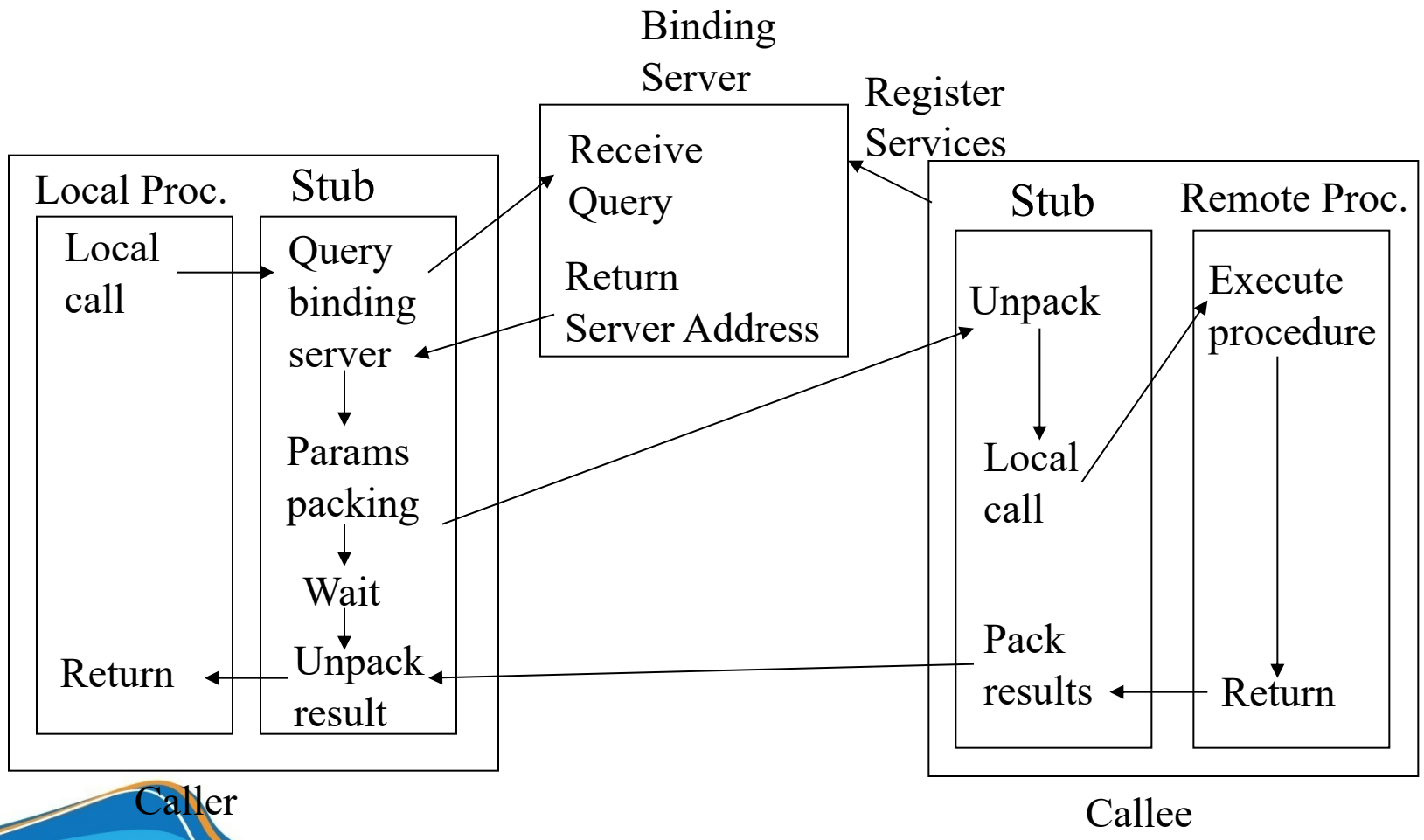
Operation oriented constructs

Remote Procedure Call (RPC):



- Mô tả dịch vụ: mô tả các tham số in/out
- Có thể cài đặt bằng cách gửi nhận msg
- Caller: bị blocked khi gọi RPC.
- Callee cài đặt như sau:
  - luôn đợi nhận cuộc gọi
  - có thể bị ngắt bởi cuộc gọi mới
  - tạo tiểu trình phục vụ cho cuộc gọi mới

Thực thi RPC


Binding
Server
Register
Services
Receive
Query
Return
Server Address
Local Proc.
Stub
Local
call
Query
binding
server
Params
packing
Wait
Unpack
result
Return
Caller
Stub
Remote Proc.
Unpack
Local
call
Execute
procedure
Pack
results
Return
Callee

RPC: các vấn đề

- Truyền con trỏ, biến toàn cục,.. có thể khó khăn.
- Các máy khác nhau, kích thước dữ liệu (số bit của mỗi kiểu dữ liệu) thay đổi cần phải đc mô tả.
  - Abstract Data Types (ADTs) thường được sử dụng để gq vấn đề
  - ADTs là ngôn ngữ có cấu trúc dùng mô tả integer bao nhiêu bit, v.v…
- Nhiều tiến trình có thể cung cấp chung 1 dịch vụ? Định danh cần phải được xác định trước.
- Synchronous/blocked message passing được dùng trong RPC.

Thiết kế RPC

- Cấu trúc
  - Caller: local call + stub
  - Callee: stub + procedure thực hiện
- Kết nối dịch vụ
  - Tìm máy có phục vụ dịch vụ
- Tham số & kết quả
  - Đóng gói: chuyển sang định dạng của máy ở xa
  - Mở gói: chuyển sang định dạng của máy cục bộ

RPC Semantics

- At least once
  - A RPC results in zero or more invocation.
  - Partial call, i.e., unsuccessful call: zero, partial, one or more executions.
- Exactly once
  - Only one call maximum
  - Unsuccessful? : zero, partial, or one execution
- At most once
  - Zero or one. No partial executions.

Triển khai RPC

- Gửi/nhận tham số:
  - Sử dụng giao thức truyền tin cậy? :
  - Sử dụng giao thức truyền không tin cậy – UDP?

Hạn chế của RPC

- Không thể trả về kết quả một cách riêng lẻ: (v.d.,) khong trả ngay về một vài mẫu tin tìm được trong CSDL. Phải đợi đến khi có toàn bộ kết quả.

Hệ điều hành phân tán

Các vấn đề:

- Global Knowledge
- Naming
- Scalability
- Compatibility
- Process Synchronization
- Resource Management
- Security
- Structuring
- Client-Server Model

DOS: các vấn đề ..

- Global Knowledge
  - Thiếu bộ nhớ chia sẻ toàn cục, đồng hồ toàn cục, không dự đoán đc độ trễ của msg
  - Dẫn đến không dự đoán đc global state, khó sắp xếp sự kiện (A sends to B, C sends to D: có thể có quan hệ)
- Naming
  - Cần dịnh vụ định danh (files, databases), users, services (RPCs).
  - Nhân bản thư mục? : bài toán cập nhật?
  - Cần dịch vụ phân giải giữa tên và địa chỉ (IP).
  - Thư mục phân tán: thuật toán tìm kiếm, cập nhật ntn ...

DOS: các vấn đề ..

- Scalability
  - Mở rộng hệ thống có dễ dàng?
  - Bao gồm: chi phí để trao đổi thông điệp tìm kiếm, cập nhật thông tin...
- Compatibility
  - Mức binary: tương thích ở mức lệnh máy
  - Mức Execution: cùng mã nguồn có thể biên dịch và thực thi
  - Mức Protocol: có cơ chế để trao đổi msg, data.

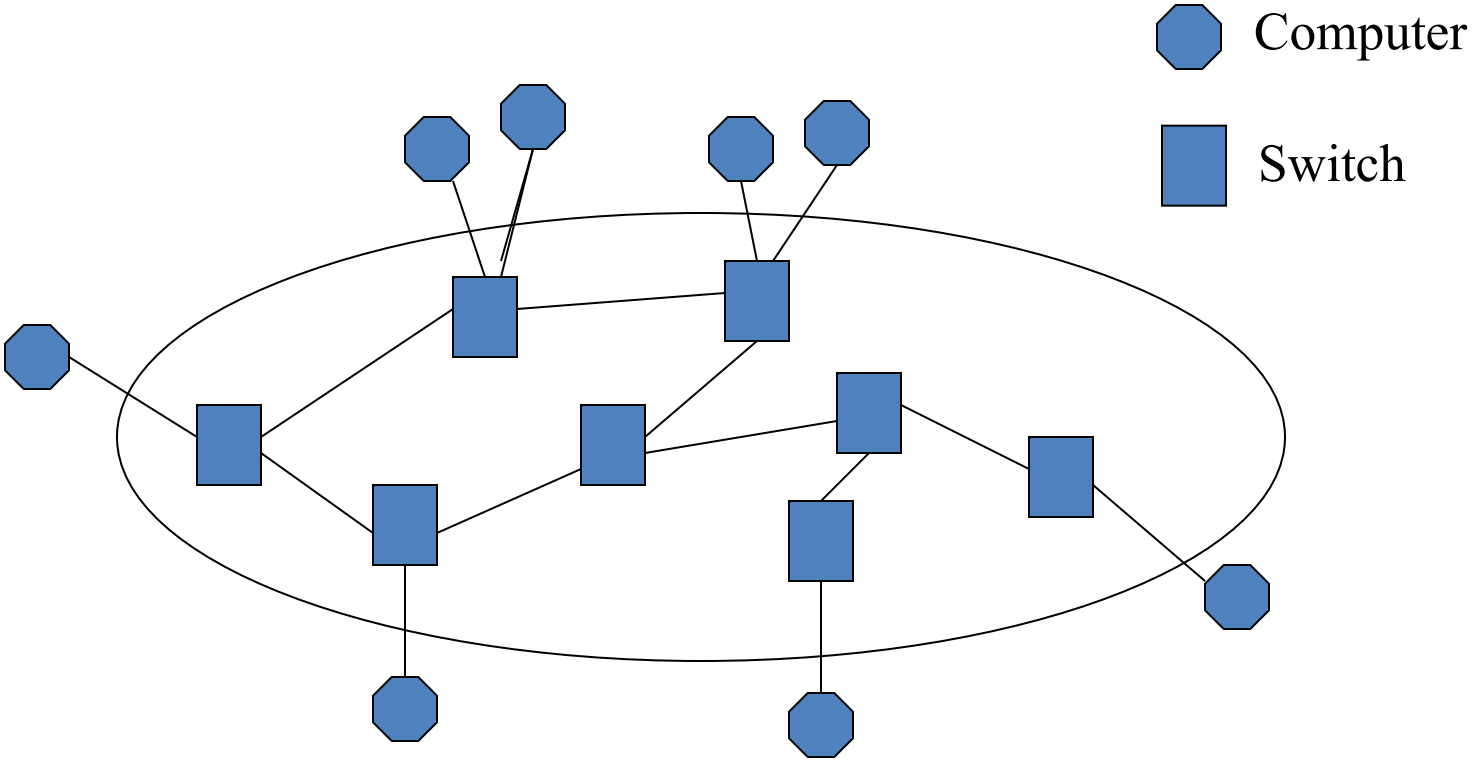
DOS: các vấn đề ..

- Process Synchronization
  - Chia sẽ bộ nhớ phân tán: khó.
- Resource Management
  - Quản lý Data/object: xử lý việc di chuyển file/object sao cho trong suốt trong hệ phân tán.
  - Bài toán cơ bản: nhất quán, giảm thiểu độ chậm trễ, ..
- Security
  - Xác thực và quyền hạn

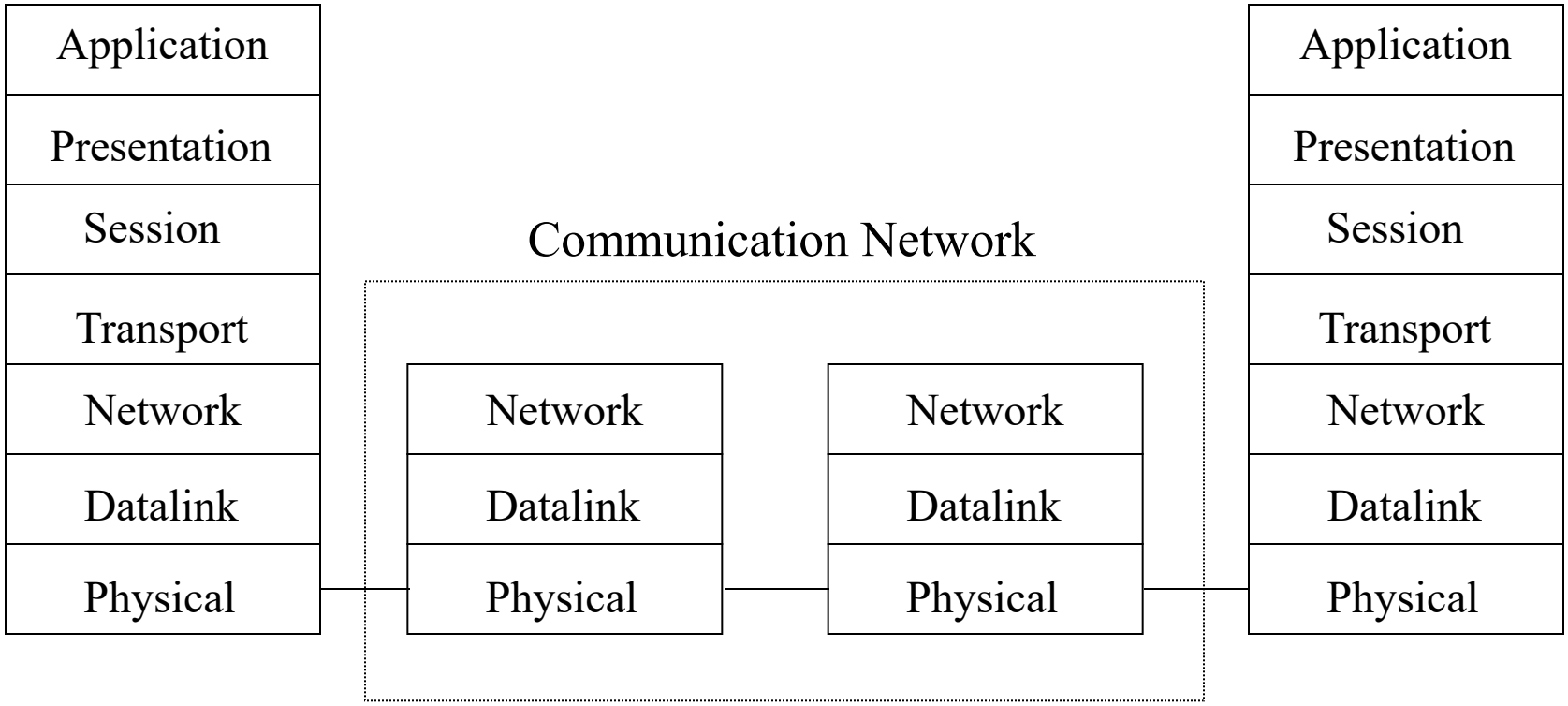
DOS: các vấn đề ..

- Structuring
  - Monolithic Kernel: không nhất thiết(e.g.,) quản lý tập tin ko nhất thiết phải có trên các máy không ổ đĩa.
  - Collective kernel: chức năng phân tán trên toàn hệ thống.
    - Micro kernel + set of OS processes
    - Micro kernel: có các chức năng quản lý bộ xử lý, bộ nhớ, và tác vụ. Thực hiện trên tất cả hệ thống.
    - OS processes: tập các công cụ. Thực thi khi cần.
  - Object-oriented system: hướng đối tượng.
    - Loại đối tượng: process, directory, file, …
    - Hành động trên đối tượng.

DOS: kết nối



ISO-OSI Reference Model



Giao tiếp tin cậy/không tin cậy

- Giao tiếp tin cậy
  - Virtual circuit: một đường dẫn giữa bên gửi và nhận. Tất cả gói tin gửi qua đường này.
  - Dữ liệu nhận đúng theo thứ tự gửi.
  - TCP (Transmission Control Protocol) cung cấp dvu truyền tin cậy.
- Giao tiếp không tin cậy
  - Datagrams: có thể gửi theo nhiều đường dẫn.
  - Dữ liệu có thể bị mất hoặc ko đến đúng thứ tự.
  - UDP (User datagram Protocol) cung cấp dvu không tin cậy