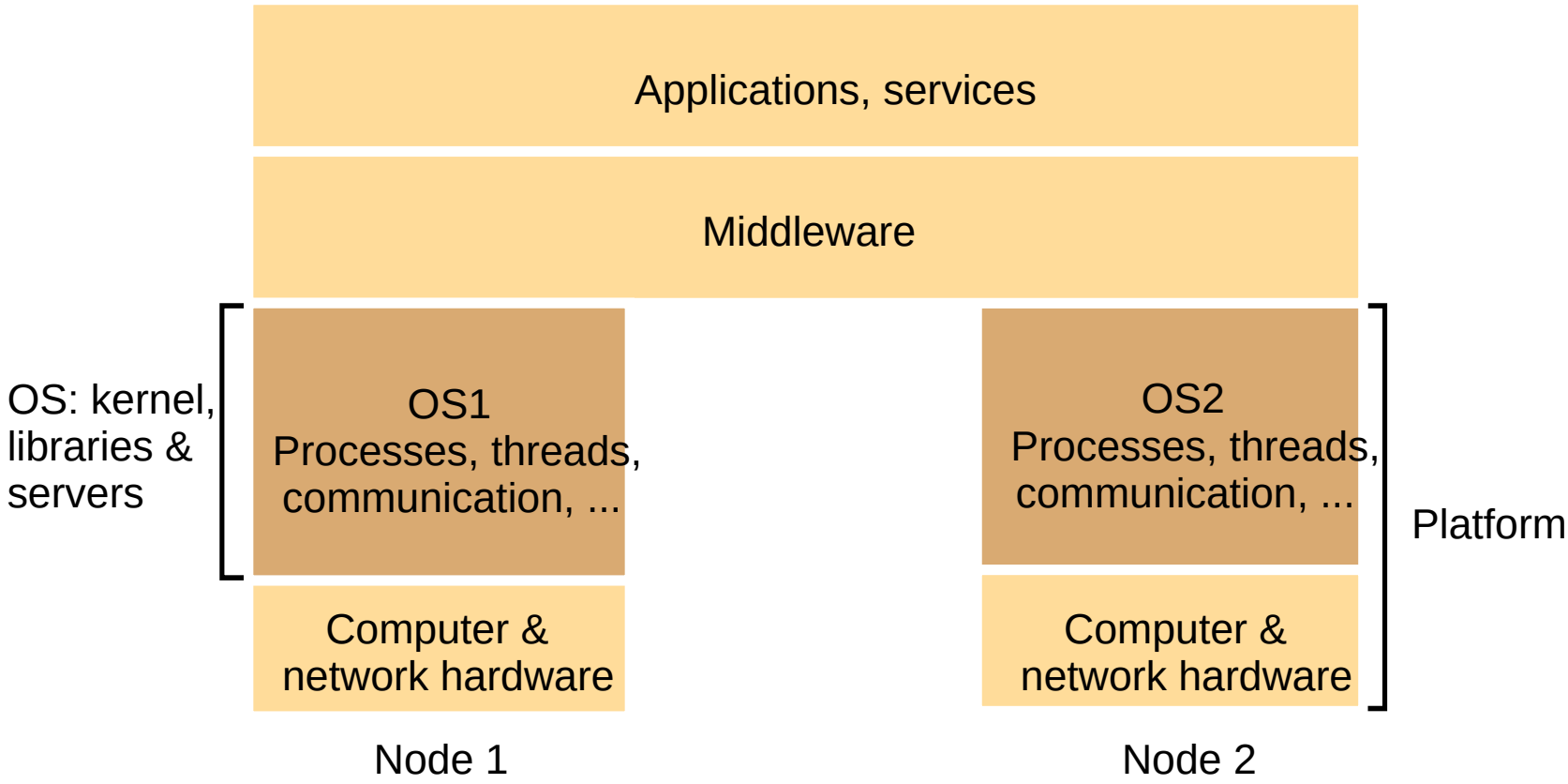
Sistem Terdistribusi

Sistem Operasi Terdistribusi

Peran sistem operasi

- Menyediakan abstraksi kontrol sumber daya fisik bagi user
- Manajemen resource
- Menyediakan sistem call terhadap sumber daya baik fisik / non fisik
- Jenis:
  - Desktop OS
  - Network OS
  - Distributed OS

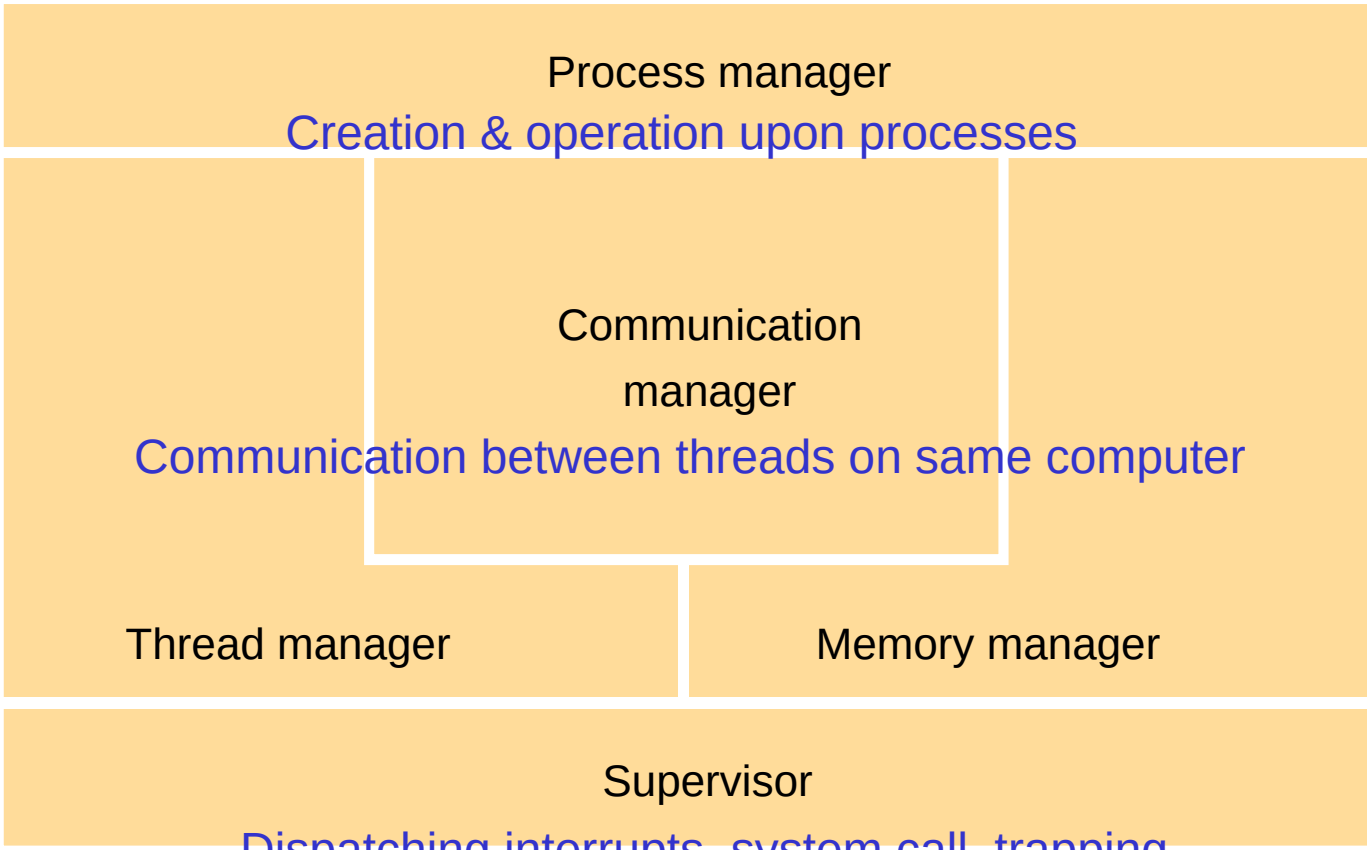
System layers



OS Tasks

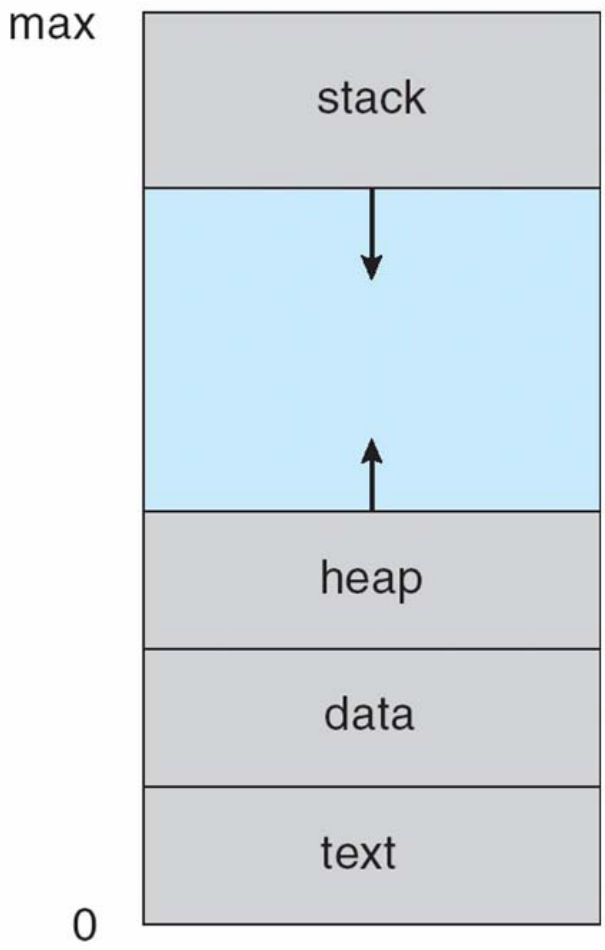
- raise the **programming interface** for these resources to a more useful level:
  - By providing **abstractions / encapsulation** of the basic resources such as:
    processes, unlimited virtual memory, files, communication channels
  - **Protection** of the resources used by applications
  - **Concurrent processing** to enable applications to complete their work with minimum interference from other applications
- provide the resources needed for (distributed) services and applications to complete their task:
  - **Communication -** network access provided
  - **Scheduling -** processors scheduled at the relevant computers

Core OS functionality



*

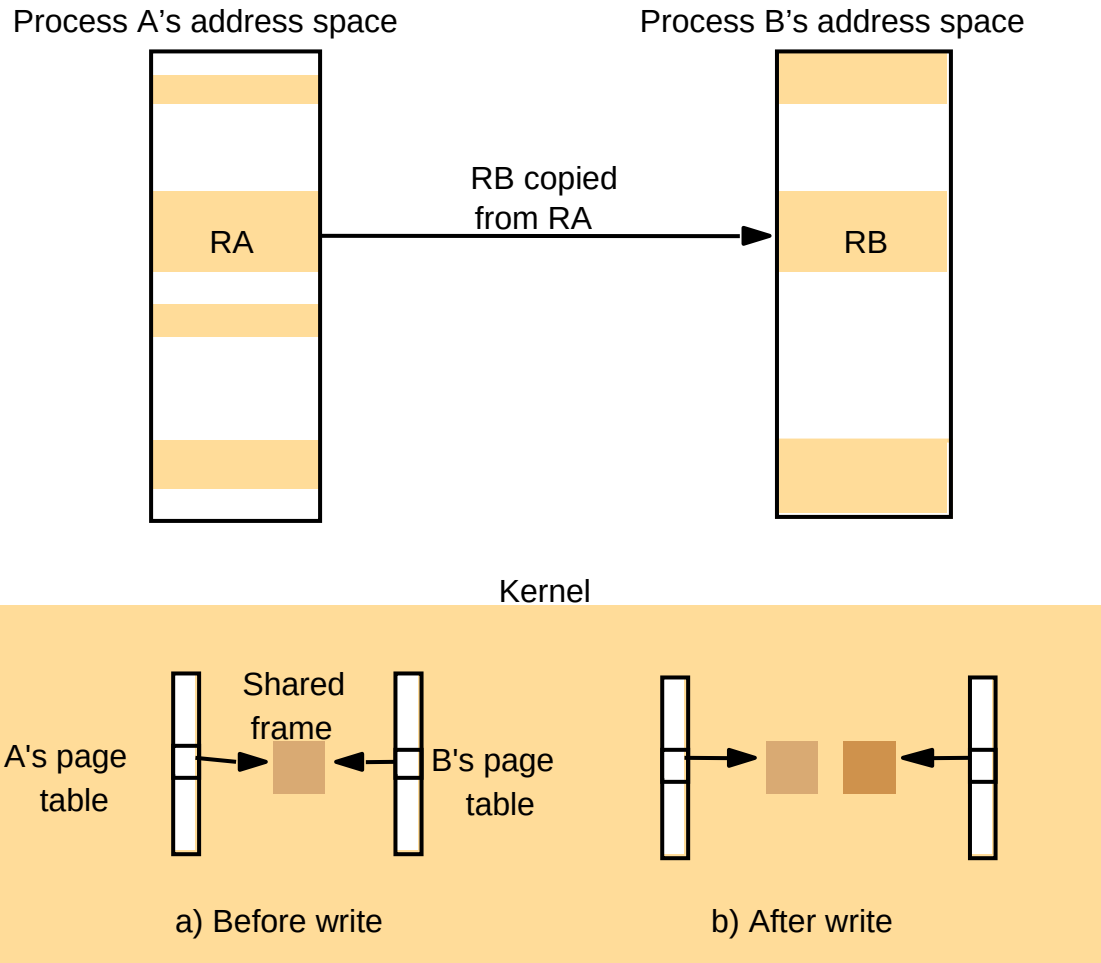
Process address space



- Regions can be shared
  - kernel code
  - libraries
  - shared data & communication
  - **copy-on-write system**
- Files can be mapped to memory
  - Virtual memory

*

Copy-on-write



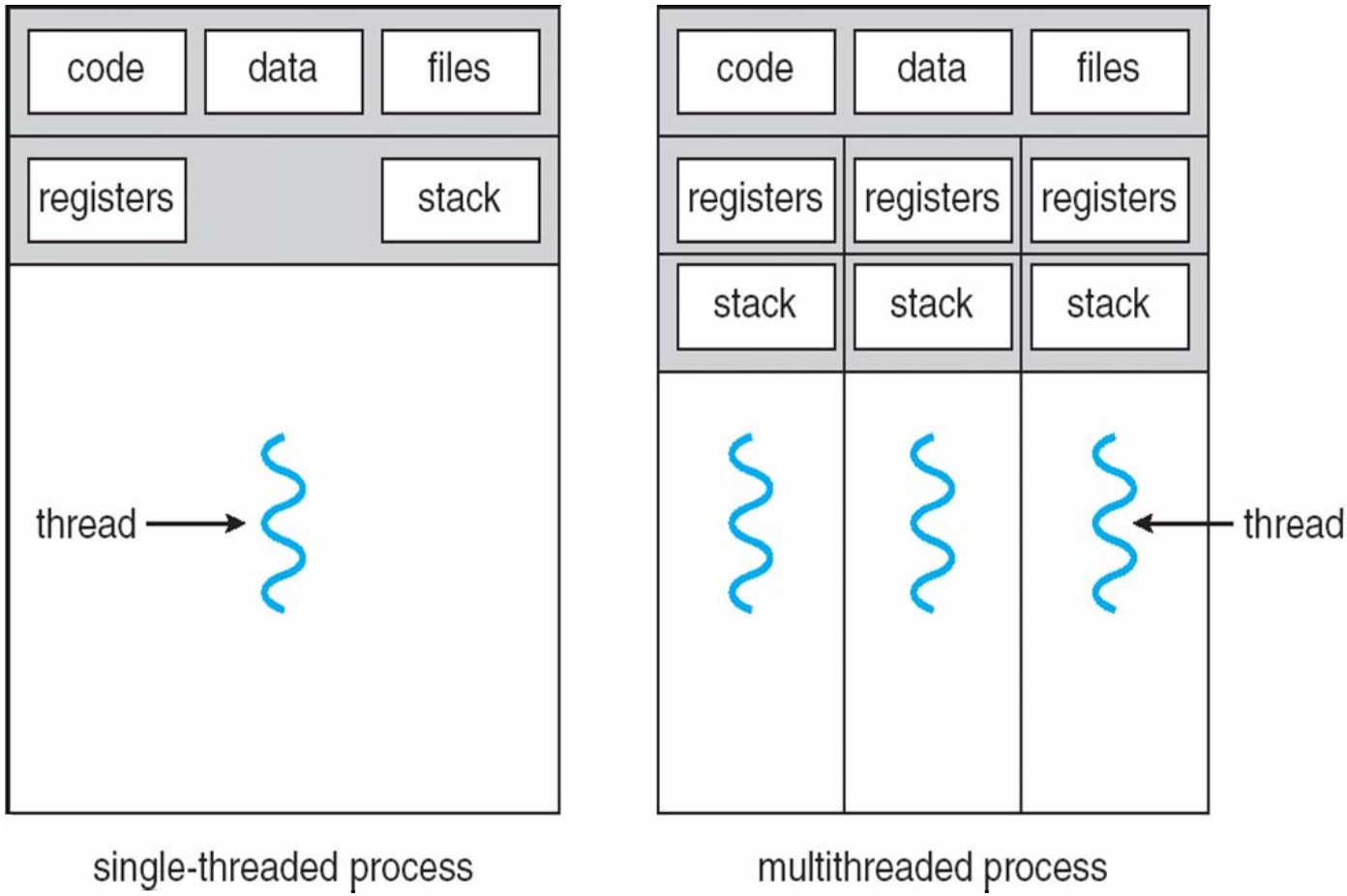
Copy on Write

- Region RA dan RB merupakan sharing memori dengan teknik copy-on write antara dua proses A dan B
- Ketika proses B hendak menulis ke share memory, maka akan terjadi **memory protection page fault,** sehingga akan mengalokasikan frame baru dari hasil duplikasi frame asli sehingga akan terdapat 2 page frame

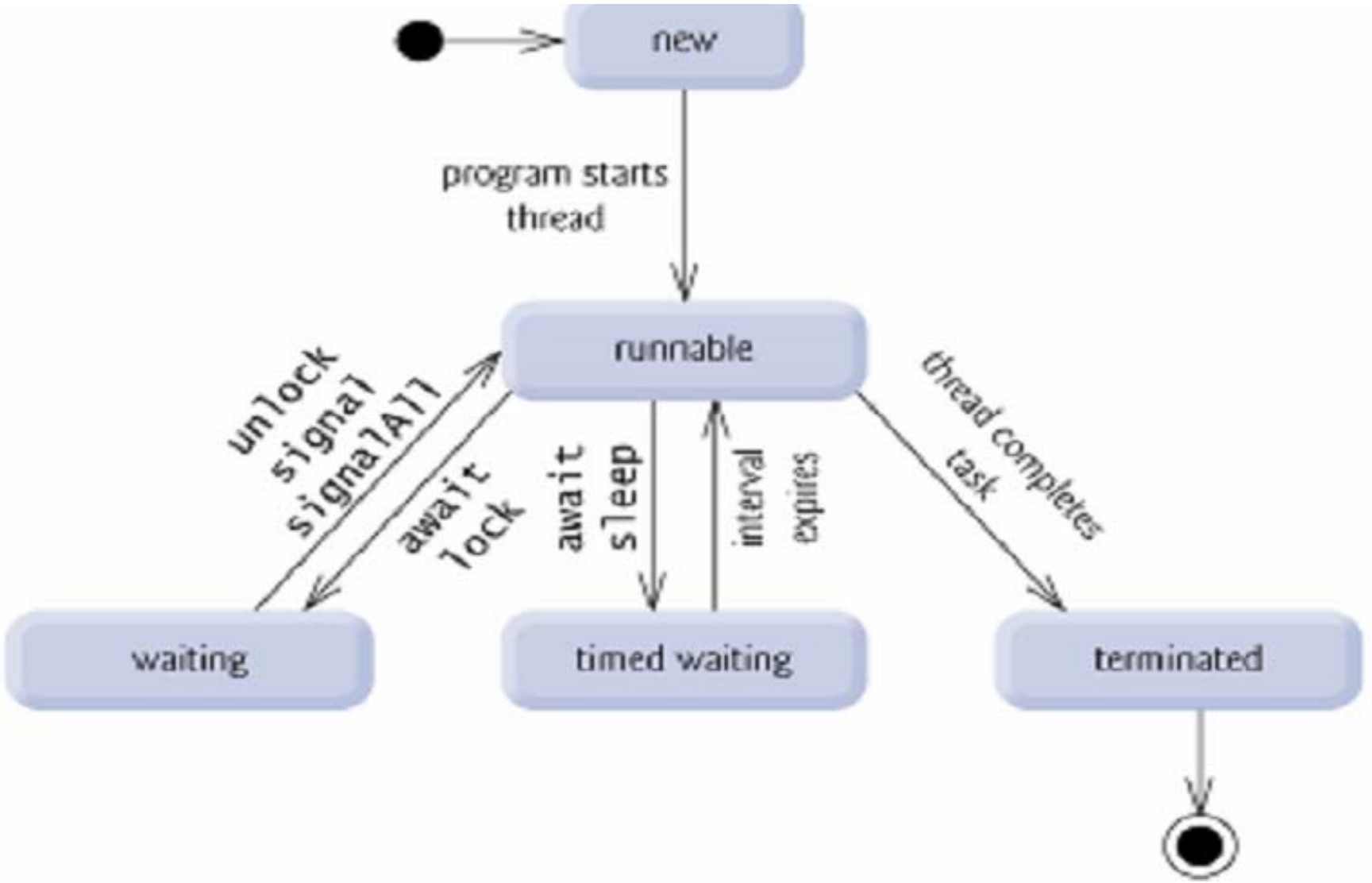
Thread

- Thread = Lightweight Process
- Thread = satuan dasar penggunaan CPU
- Pembuatan Thread dilakukan oleh:
  - Kernel Thread – lebih lambat
  - User Thread – lebih cepat, berbasis API
- Kernel digunakan dalam Distributed OS
  - Multithreading

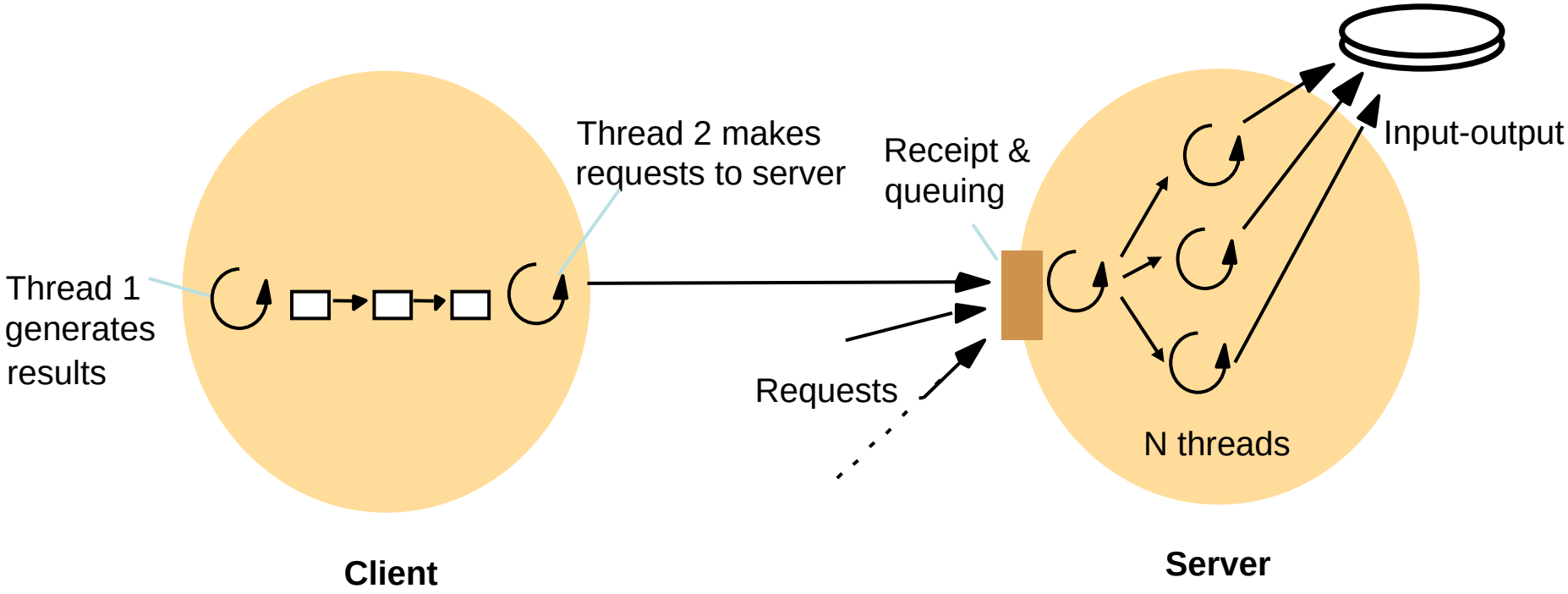
Single and Multithreaded Processes



Thread Life Cycles

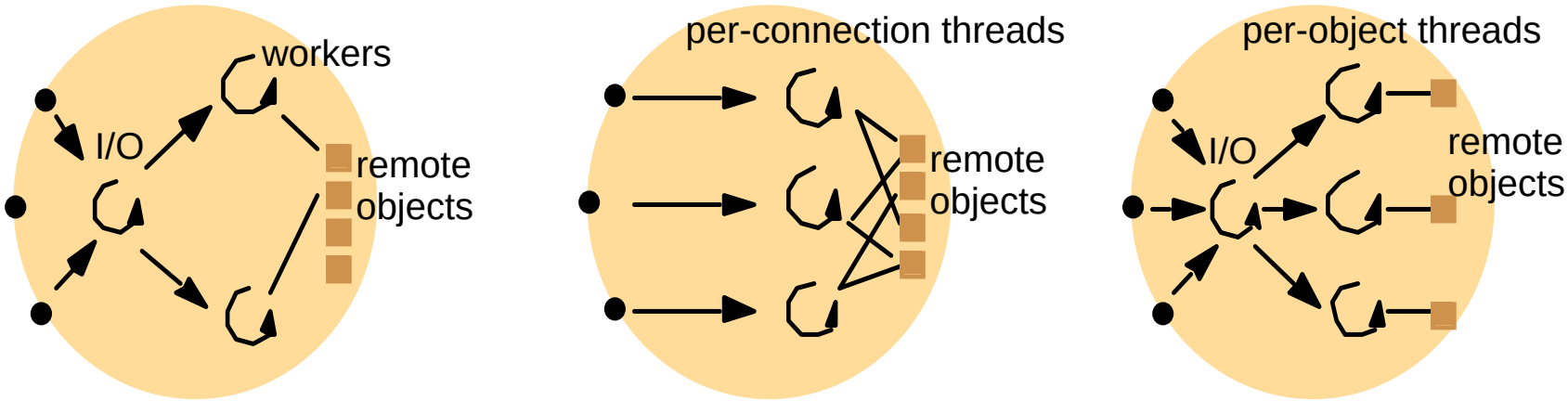


Client and server with threads



The 'worker pool' architecture

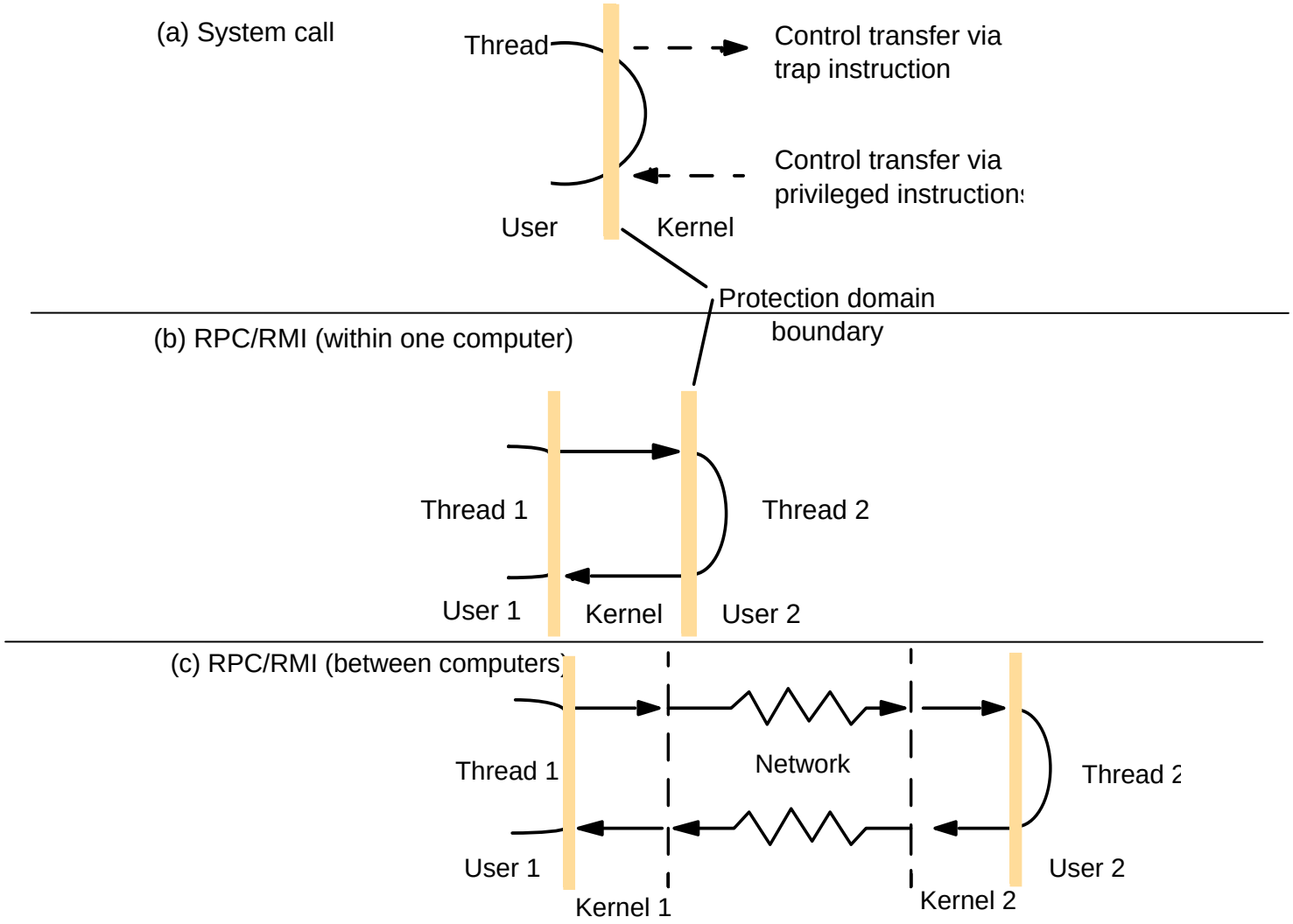
Alternative server threading architectures



a. Thread-per-request b. Thread-per-connection c. Thread-per-object

– Implemented by the server-side ORB in CORBA

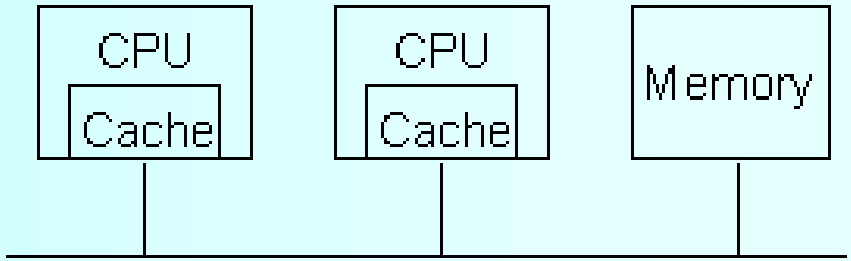
Invocations between address spaces



Arsitektur Model

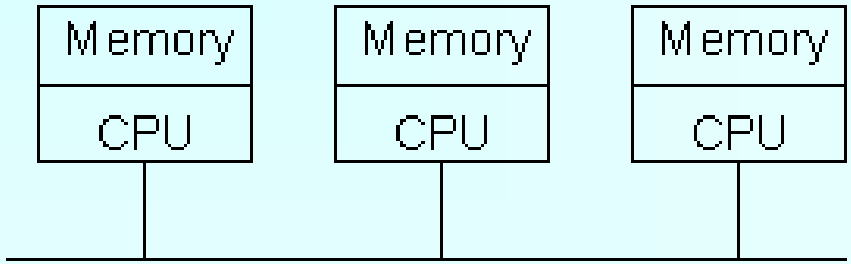
- Multiprocessors
  - Tightly coupled.
  - Shared memory.

Parallel architecture



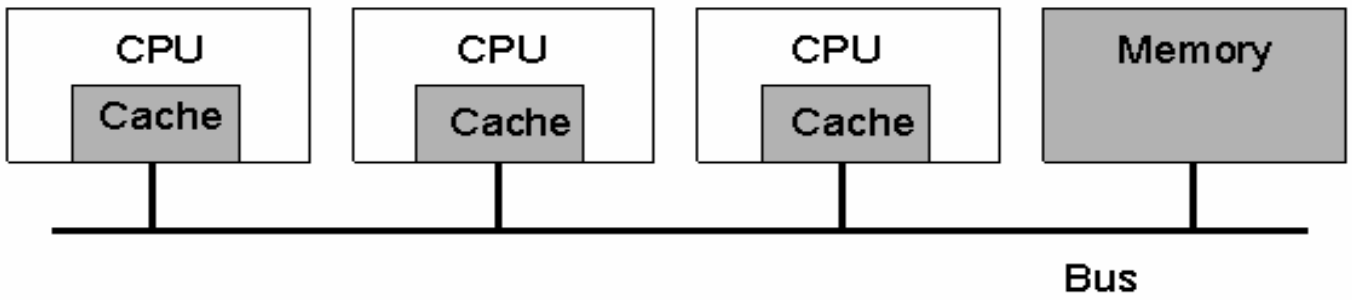
- Multicomputers.
  - Loosely coupled.
  - Private memory.
  - Autonomous.

Distributed architecture



Univprocessor vs Multiprocessors

- Uniprocessor has only one processor for OS
- Multiprocessor dimensions
  - Memory: could be shared or be private to each CPU
- A bus-based multiprocessor.



Uniprocessor Operating Systems

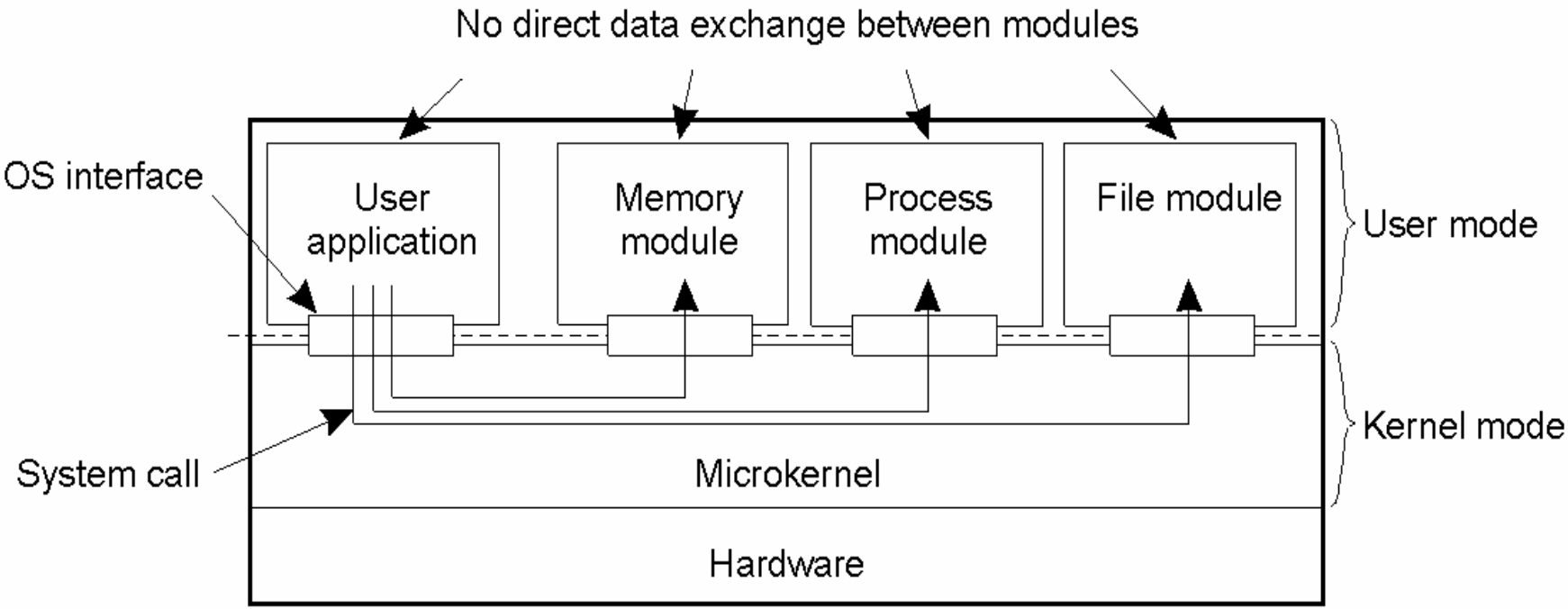
- An OS acts as a resource manager or an arbitrator
  - Manages CPU, I/O devices, memory
- OS provides a **virtual interface** that is easier to use than hardware

- Structure of uniprocessor operating systems
  - Monolithic (e.g., MS-DOS, early UNIX)
    - One large kernel that handles everything
  - Micro Kernel
    - Only essential kernel function, otherwise in user space

Uniprocessor Operating Systems

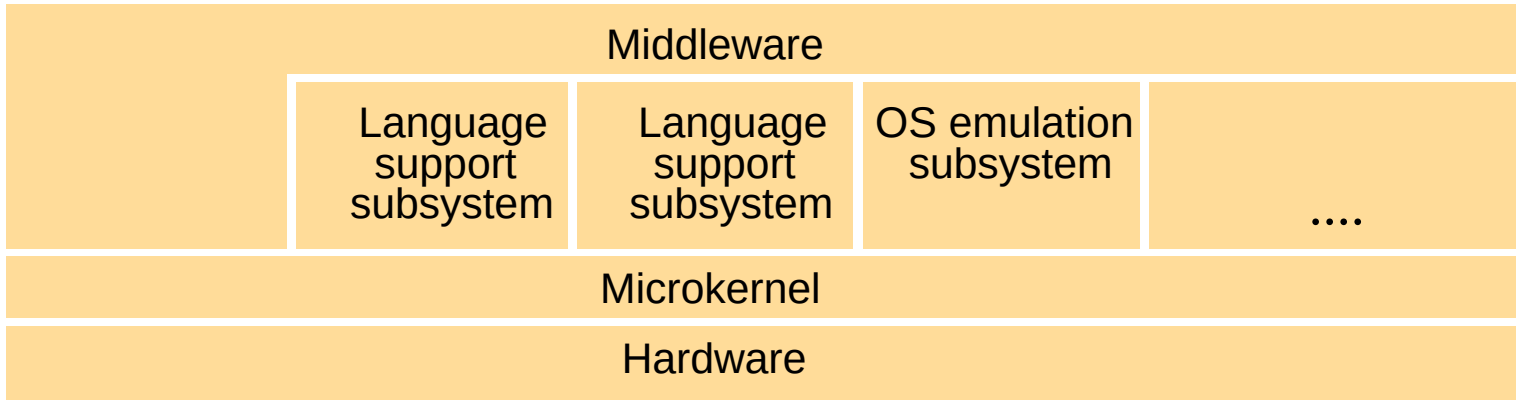
Microkernel architecture

- Small kernel: interprocess coomunication, low level I/O, memory, process management & scheduling
- user-level servers implement additional functionality



Monolithic support middleware

Figure 6.16



The microkernel supports middleware via subsystems

flexibility and extensibility

- services can be added, modified and debugged
- small kernel -> fewer bugs
- protection of services and resources is still maintained

service invocation expensive

- extra system calls by services for access to protected resources

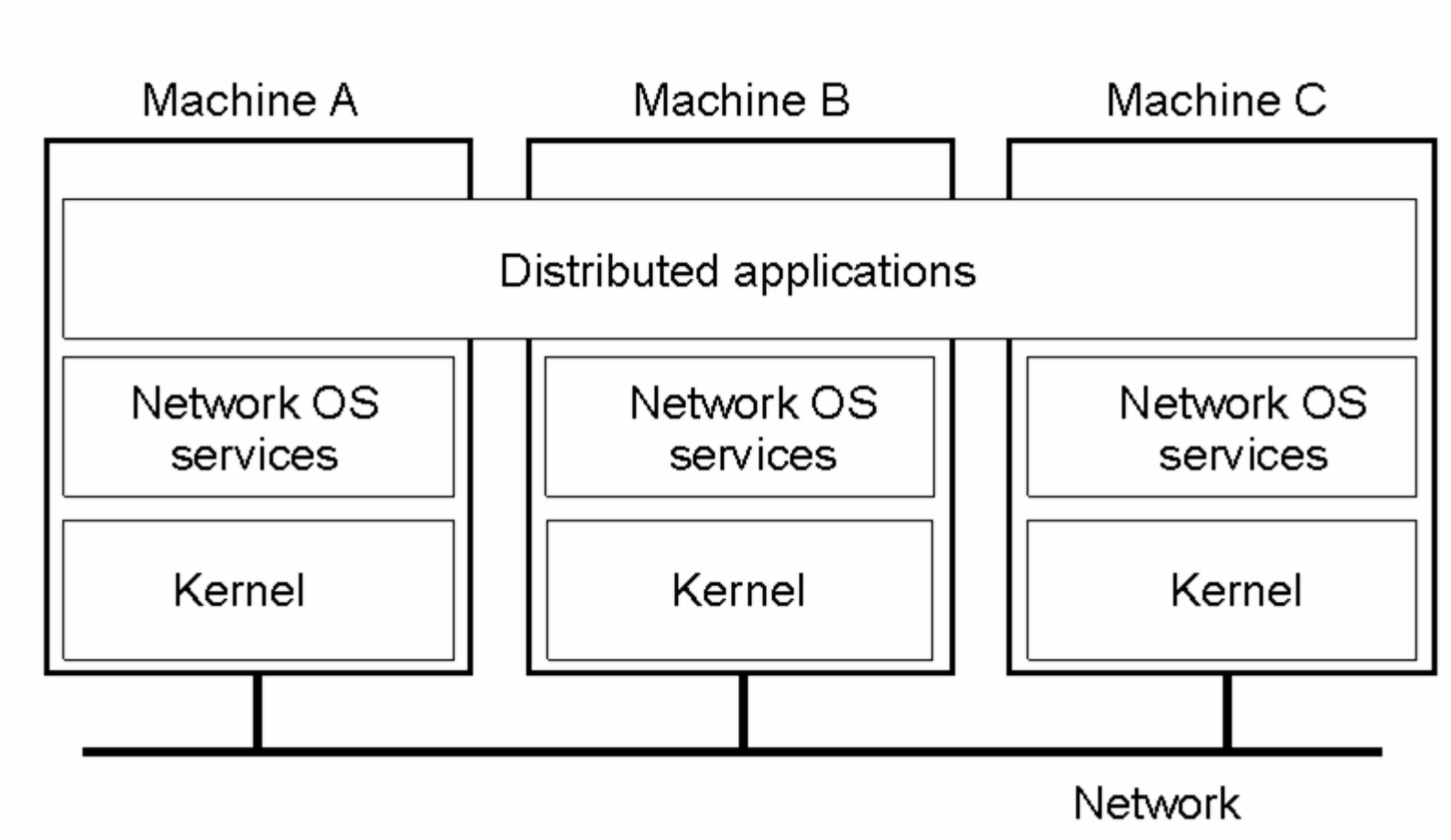
Jenis OS selain Uniprocessor OS

- Multiprocessor OS
  - Looks like a virtual uniprocessor, contains only one copy of the operating system, communication via shared memory, single run-queue
- Network OS
  - Does not look like a virtual uniprocessor, contains n copies of the operating system, communication via shared files, n run-queues
- Distributed OS
  - Looks like a virtual uniprocessor (more or less), contains n copies of the operating system, communication via messages, n run-queues

Network OS

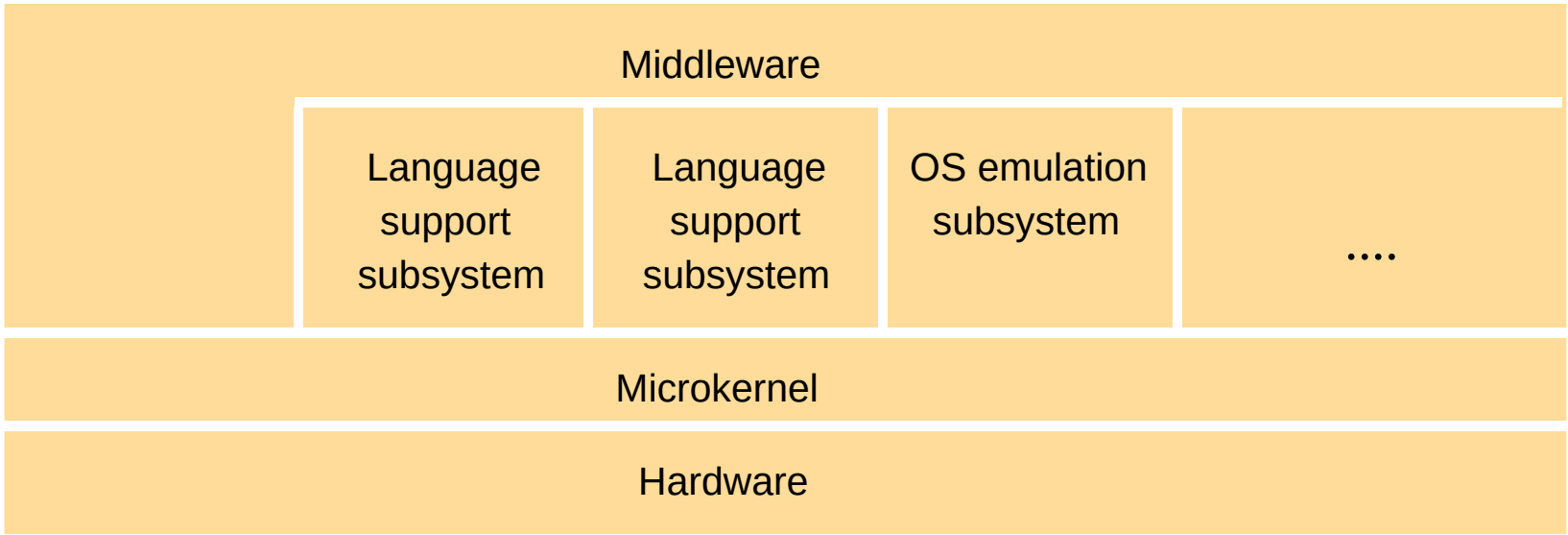
- Setiap host menjalankan Sistem Operasi untuk mengatur sumber daya yang dimilikinya termasuk mengakses sumber daya di jaringan
- Untuk mengakses resource jaringan:
  - NFS (Network File System)
  - Samba – implementasi protokol SMB di Win & Linux
- Pengguna dapat mengakses suatu proses di komputer lain dengan login ke **telnet/ssh**

Network Operating System



Network Operating System

- As client-server model – micro kernel
  - Minimal OS kernel
  - Additional functionality as user processes



The microkernel supports middleware via subsystems

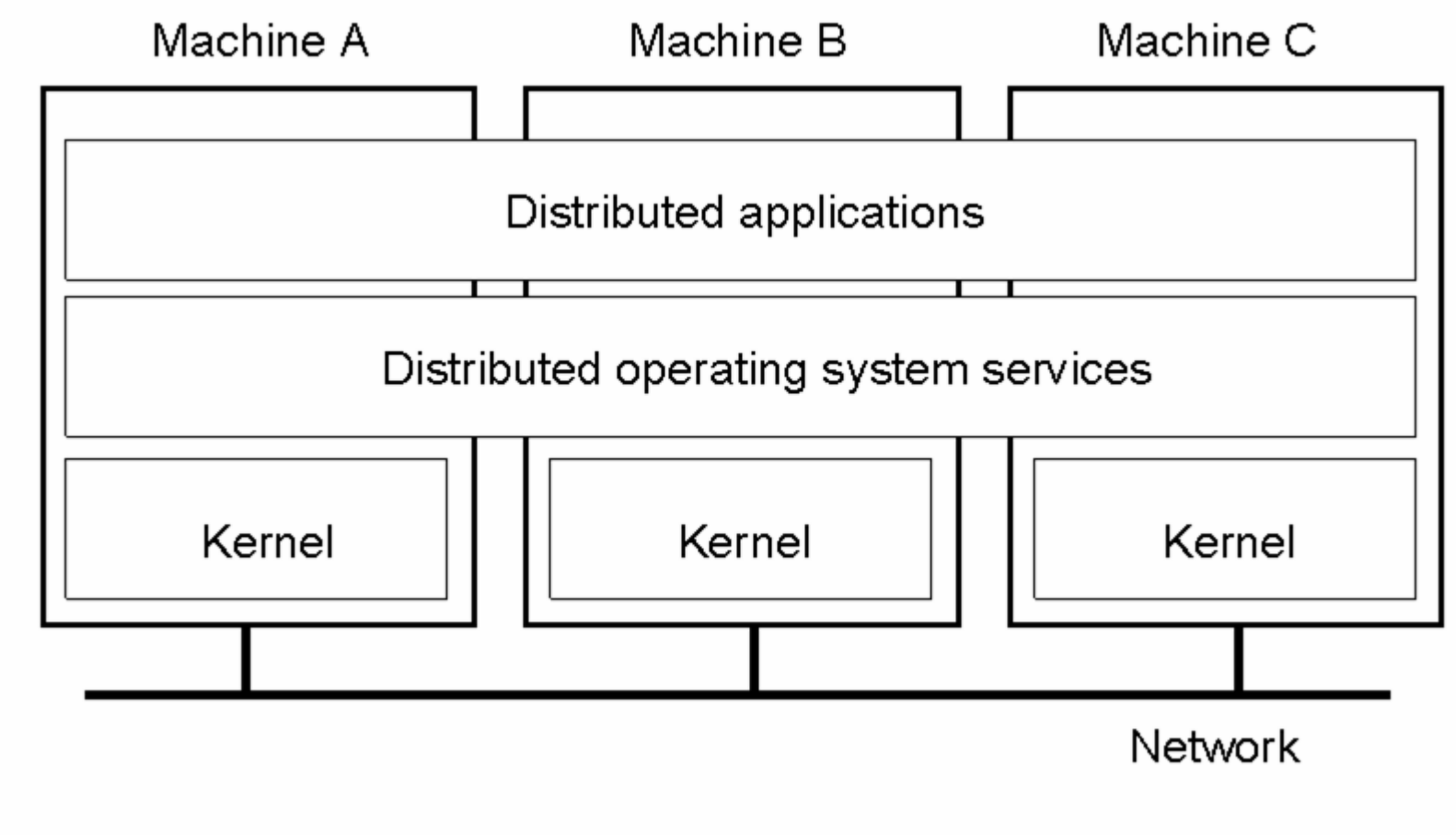
Distributed Operating System

- Dapat memanajemen komputer-komputer dan membuat “mereka” tampak sebagai sinlge komputer
- Dapat menjalankan proses di komputer lain tanpa mengetahui siapa yang meresponnya
- Manages resources in a distributed system
  - Seamlessly and transparently to the user
- Looks to the user like a centralized OS
  - But operates on multiple independent CPUs
- Provides transparency
  - Location, migration, concurrency, replication,…
- Presents users with a virtual uniprocessor

Distributed OS

- Presents users (and applications) with an integrated computing platform that hides the individual computers.
- Has control over all of the nodes (computers) in the network and allocates their resources to tasks without user involvement.
  - In a distributed OS, the user doesn't know (or care) where his programs are running.
- Examples:
  - Amoeba
  - WebOS

Distributed Operating Systems



Manfaat dan Hardware DOS

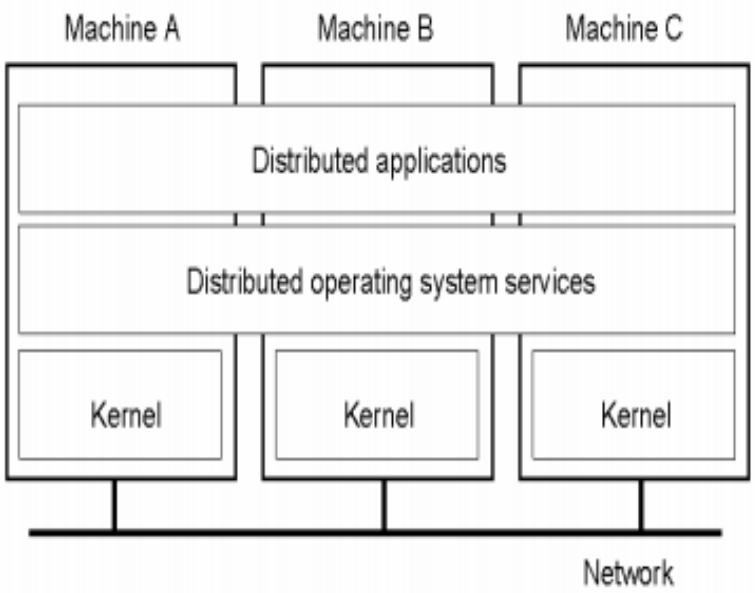
Manfaaat:

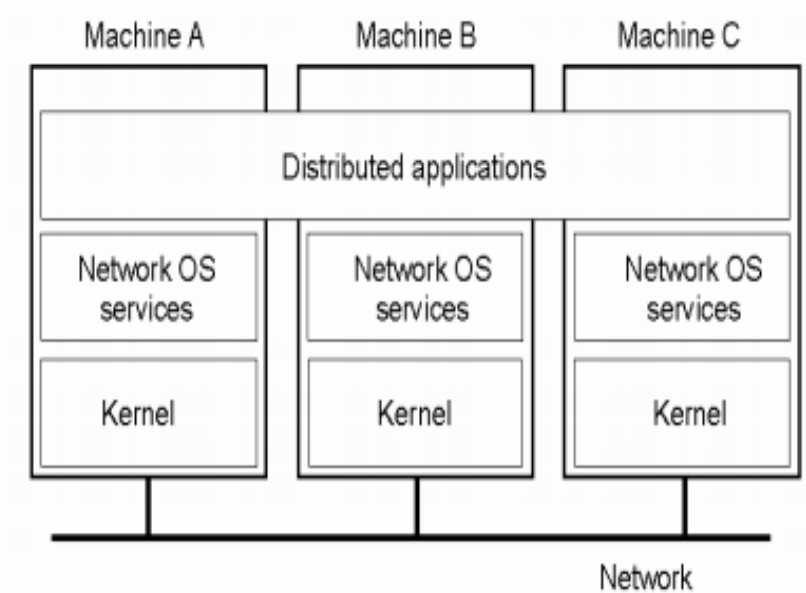
- Sharing resources
- Waktu komputasi
- Reliabilitas
- Komunikasi

Hardware

- Workstation,
- Gateway
- Server
- Processor pool.

The differences





- User is not aware of the multiple CPUs.
- Each machine runs a part of the Distributed Operating System.
- The system is fault-tolerant.

- User is aware of the existence of multiple CPUs.
- Each machine has its own private Operating System.
- The system is not fault-tolerant.

What is Amoeba?

- Amoeba is a **distributed operating system**
- Runs on a **simple micro-kernel**
- Developed by Andrew Tanenbaum
- Has user transparency
  - The user logs into the system **not a specific** machine
  - When a program is initiated, the system decides what machine will run it.

The History of Amoeba

- Developed by Andrew Tanenbaum at the *Vrije Universiteit* in conjunction with *Centrum voor Wiskunde en Informatica*
- First prototype was released in 1983
- The last official update was in 1996
- Others have developed their own versions
  - Fireball Amoeba by Fireball Software Distribution

Goals of Amoeba

- There are four main goals
  - Distribution
    - Connecting together many machines
  - Parallelism
    - Allowing individual jobs to use multiple CPUs easily
  - Transparency
    - Having the collection of computer act like a single system
  - Performance
    - Achieving all of the above in an efficient manner

Key Concepts

- Micro-kernel
  - A simple micro-kernel is the basis for Amoeba
  - All computers in the network run this kernel
  - It handles the memory management, I/O, communication, object primitive, and basic processes
- Remote Procedure Calls (RPC)
  - Used for communication between client and server
  - Accessed by stubs which are created by Amoeba Interface Language

- Threads
  - Each process has its own address space and contains multiple threads
  - These threads have their own stack and program counter, but share the global data and code of the process
- FLIP
  - Fast Local Internet Protocol
  - Developed by Andrew Tanenbaum
  - Designed to optimize the speed of RPCs

- Objects
  - The abstract data type used by Amoeba
  - Can be either software or hardware
    - But software is more common
  - Each object has a list of operations that can be preformed and a capability
- Capability
  - 128 bit value
  - Used to verify that the user has permission to access the object
  - Capabilities are encrypted

- Bullet Server
  - Store files in a contiguously fashion
  - Most files can be sent in a single RPC
  - Designed to be a dedicated server
- Directory Server
  - Handles naming of files
  - Knows the physical location of each file

Significant Points

- The system is free
- It has not had an official update in over 10 years
- Can use older/slower CPUs to create a powerful system
- Micro-Kernel allows for other file systems to be created
- Has many UNIX like commands and programs
- Can only hold programs as large as its physical memory

Comparison of DOS

	Cambridge	***Amoeba***	***V Kernel***	***Eden Project***
Developed By	Computing Laboratory@ Univ. of Cambridge	Tanenbaum@ Vrije Universiteit-Amsterdam	David Cheriton@ Stanford University	University of Washington-Seattle
Communication Primitives	RPC	RPC	RPC	RPC
Naming and Protection	Single Name Server	Sparse capabilities with encryption	Three-level naming mechanism	Capabilities without protection
Resources	Processor Bank	Processor Pool	Workstation Model	Workstation Model
Fault tolerance	Small server to startup services	Some fault tolerance through boot server	No fault tolerance	Uses Recorder process.
File Server	Universal file service and Filing Machine	Several file services.	Similar to Unix	No file server. One process for each file

NEXT

- TTS
- Distributed File System or Replication?