Distributed Systems

1. Introduction

Literature

- **Andrew Tanenbaum**
  - **Distributed Systems, Prentice Hall, 2002**
- **George Coulouris, Jean Dollimore, Tim Kindberg**
  - **Verteilte Systeme – Konzepte und Design, 3. Auflage**
- Randy Chow and Theodore Johnson
  - Distributed Operating Systems & Algorithms
- Doug Lea
  - Concurrent Programming in Java
- Henry Bal
  - Programming Distributed Systems
- Gregory Andrews
  - Concurrent Programming
- Laszlo Böszörmenyi and Carsten Weich
  - Programming in Modula-3 (chapter 16)
- Andrew Tanenbaum
  - Distributed Operating Systems

What is Distributed?

- "A distributed system is a collection of independent computers that appear to the users of the system as a single coherent system." – *Tanenbaum & Enslow*
- "A distributed system is a system designed to support the development of applications and services which can exploit a physical architecture consisting of multiple, autonomous processing elements that do not share primary memory but cooperate by sending asynchronous messages over a communication network" – *Blair & Stefani*
- "A distributed system is one that stops you getting any work done when a machine you've never even heard of crashes" – *Leslie Lamport*

Why Distributed?

- Resource and Data Sharing
  - Printers, databases, multimedia servers etc.
- Availability, Reliability
  - The loss of some instances can be hidden
- Scalability, Extensibility
  - System grows with demands (e.g. extra servers)
- Performance
  - Huge power (CPU, memory etc.) available
  - *Horizontal* distribution (same logical level is distr.)
- Inherent distribution, communication
  - Organizational distribution, e-mail, video conference
  - *Vertical* distribution (corresponding to org. struct.)

Problems of Distribution

- Concurrency, Security
  - Clients must not disturb each other
- Partial failure
  - We often do not know, where is the error (e.g. RPC)
- Location, Migration, Replication
  - Clients must be able to find their servers
- Heterogeneity
  - Hardware, platforms, languages, management
- Convergence
  - Between distributed systems and telecommunication

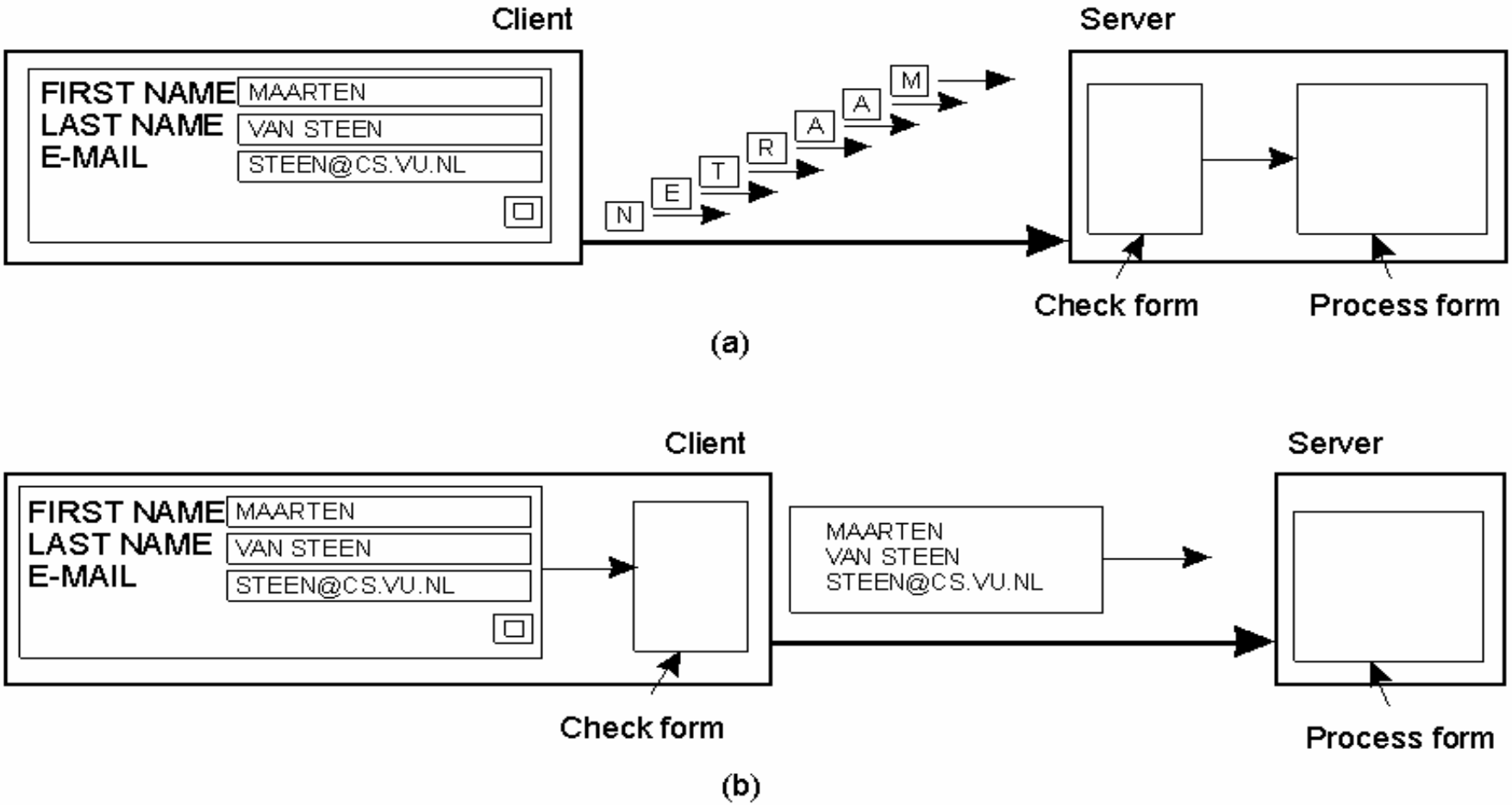
Distribution Transparencies

- Access
  - Hide differences in data representation (big/little “endians” etc.)
- Location
  - Resources are found by name, regardless from the location
- Migration
  - Resources can move and still be found by name
- Relocation
  - Resources can move *while in use* and still be found by name
- Replication
  - Arbitrary num. of copies can exist, automatic consistency is guaranteed
- Persistence
  - Hide whether a (software) resource is in memory or on disk
- Failure
  - Hide failure and recovery of a resource – fairly difficult
- Parallelism
  - Automatically distribute work among processing units – very difficult

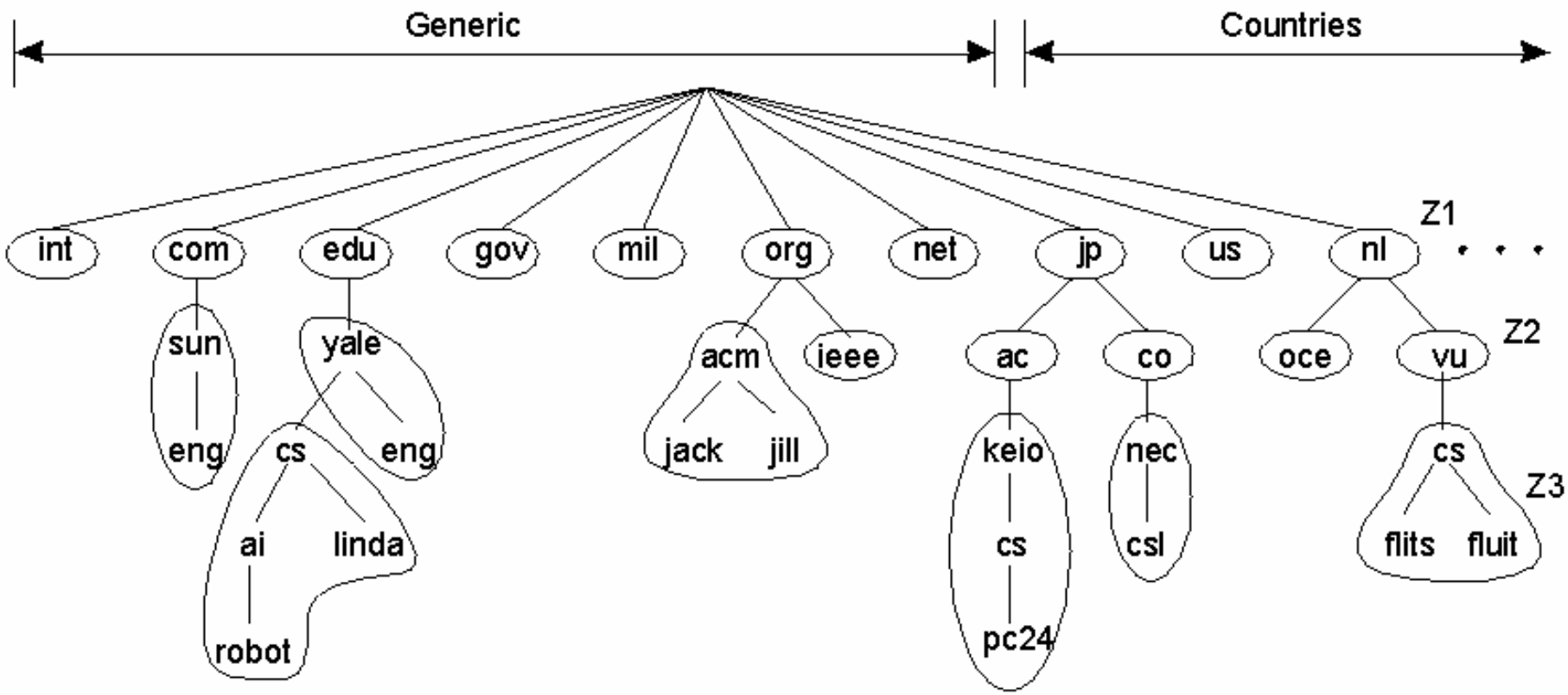
Concurrent, Distributed, Parallel Progr.

- **Common features**
  - The program consists of more than one thread of control
  - No explicit assumption about time (as opposed to *real-time* prog.)
- **Concurrent Programming**
  - Main goal: Inherent parallelism (concurrency)
  - Based on (generally quasi-parallel) lightweight threads, normally uses common memory, often well supported by general-purpose programming languages (e.g. Concurrent Pascal, Java)
- **Parallel Programming**
  - Main goal: Speed-up and efficiency ($S_n = T_1/T_n$, $E_n = S_n/n$)
  - Mostly based on threads, uses common or distributed memory, support by hardware and special languages (High-Speed Fortran)
- **Distributed Programming**
  - Main goal: Physical distribution (for several reasons)
  - Sometimes supported by special languages (Orca), uses distributed shared memory, socket or RPC-based communication

Scalability – example (1)



Scalability – example (2)

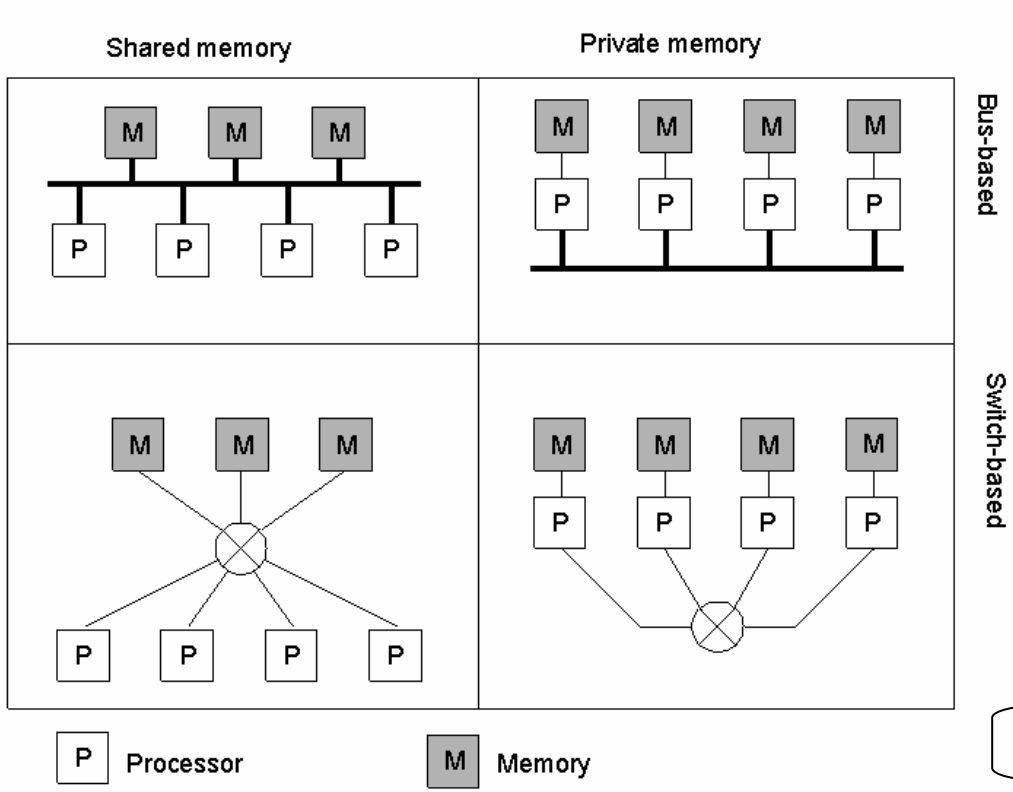


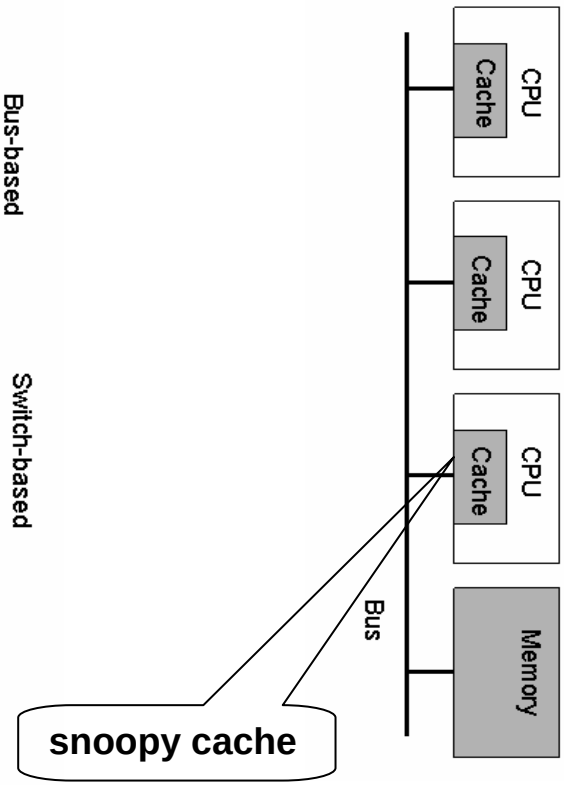
Openness

- Well-defined Interfaces
  1. Black box with no public interfaces
  2. Black box with a well-defined public external interface
  3. White box with well-defined public internal interfaces
- Interoperability
  - Components of different origin can communicate
- Portability
  - Components work on different platforms
- Separation of Concerns
- Standards – a necessity
  - Should allow competition in *non-normative* areas

Hardware Concepts

- Multiprocessors (*Tight* coupling; fast, expensive system-bus or -switch based)
  - Shared memory
  - Gets new dimension with multi-core (e.g. 64 processors)
- Multicomputers (*Loose* coupling; off-the shelf connections, e.g. switched LAN)
  - Message passing – no shared memory, no snoopy cache



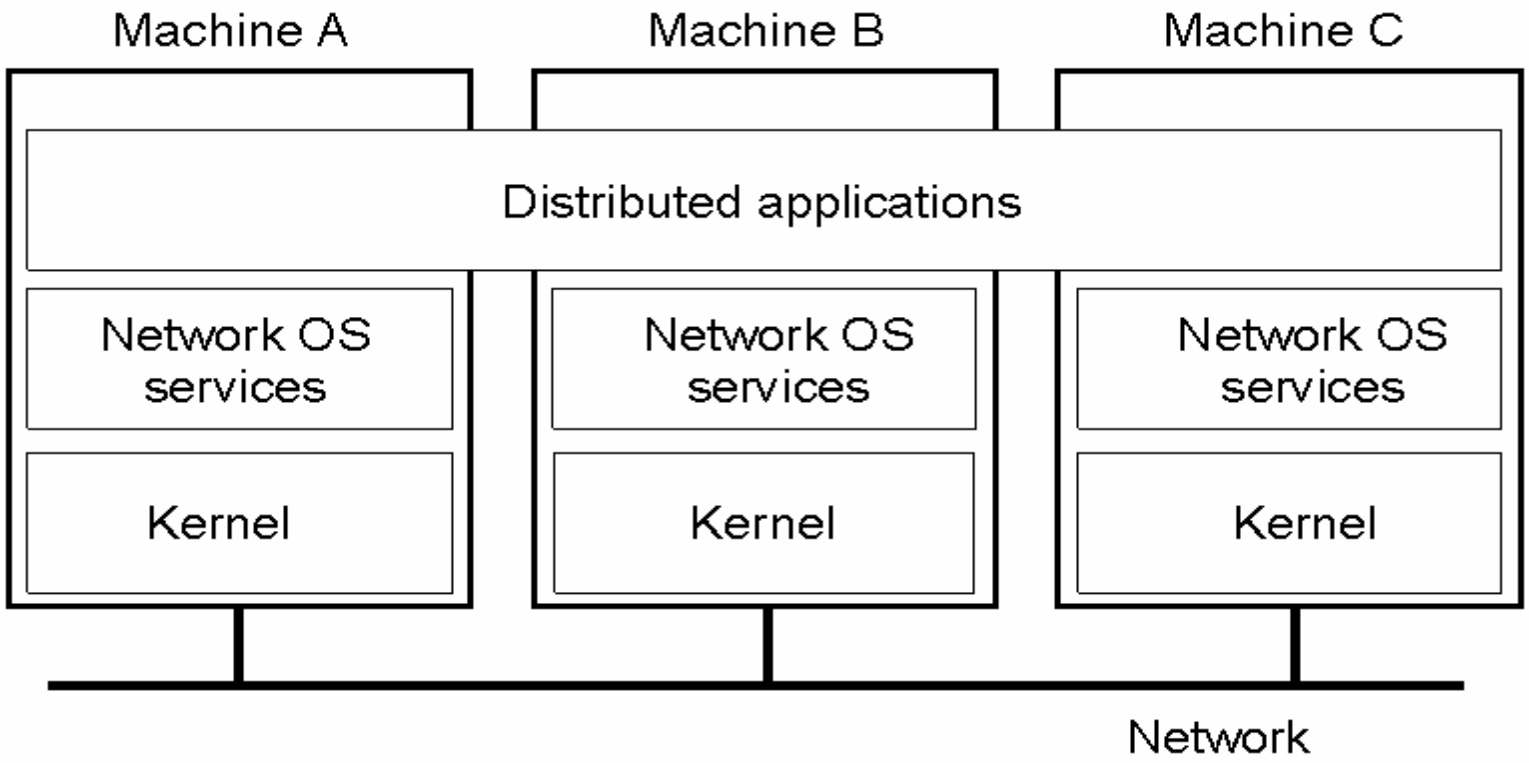


Software Concepts

System	Description	Main Goal
True Distributed Systems	Tightly coupled software on loosely coupled hardware	Single system image (does NOT exist in pure form)
DOS (Distributed Operating Systems)	Tightly-coupled operating system for tightly coupled or at least homogeneous hardware (multi-processors and homogeneous multi-computers)	Hide and manage hardware resources
NOS (Network Operating Systems)	Loosely-coupled operating system for loosely-coupled, heterogeneous multicomputers (LAN and WAN)	Offer local services to remote clients
Middleware	Additional layer atop of NOS implementing general-purpose services	Provide distribution transparency

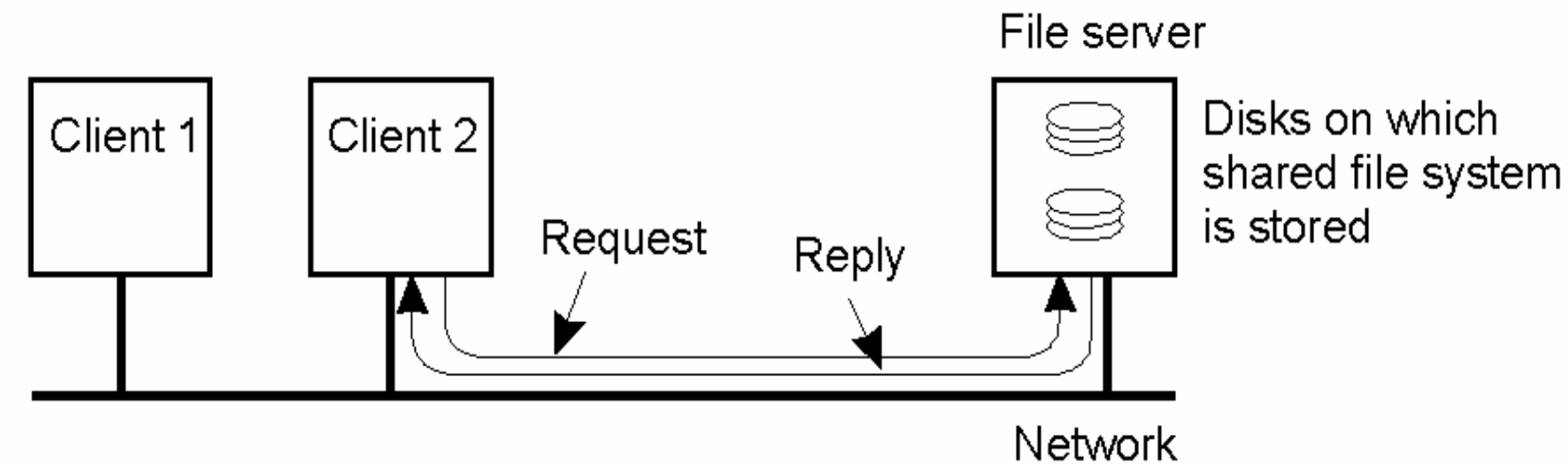
Network Operating System (1)

- Coupling is typically restricted to the file system (also called Network File System)
- Different underlying Operating Systems (Unix, Windows, etc.)



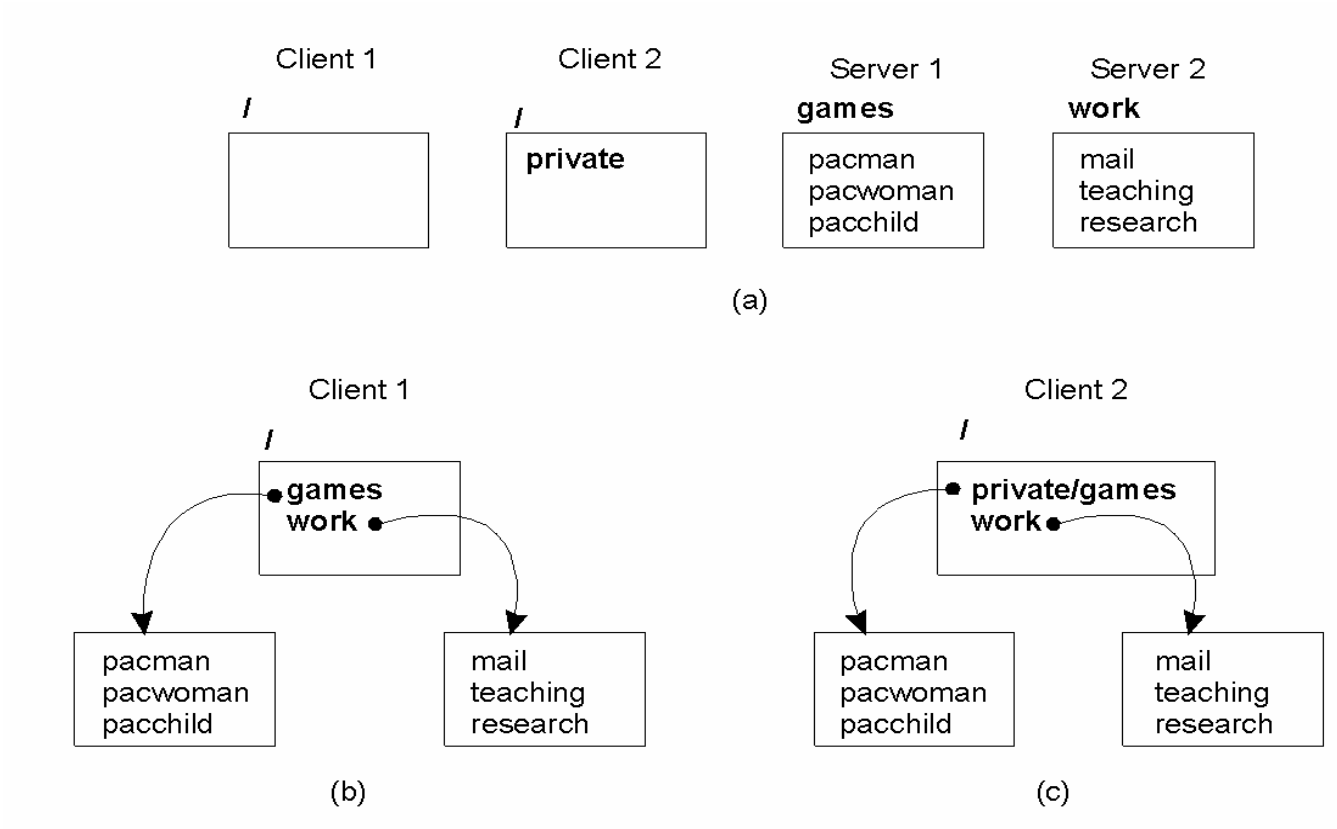
Network Operating System (2)

- File access is uniform (almost)
- NFS (Network File System, SUN), SMB (Server Message Block, Microsoft), Andrew file system …
- Client / Server Model
- Example: Two clients and a server in a NOS



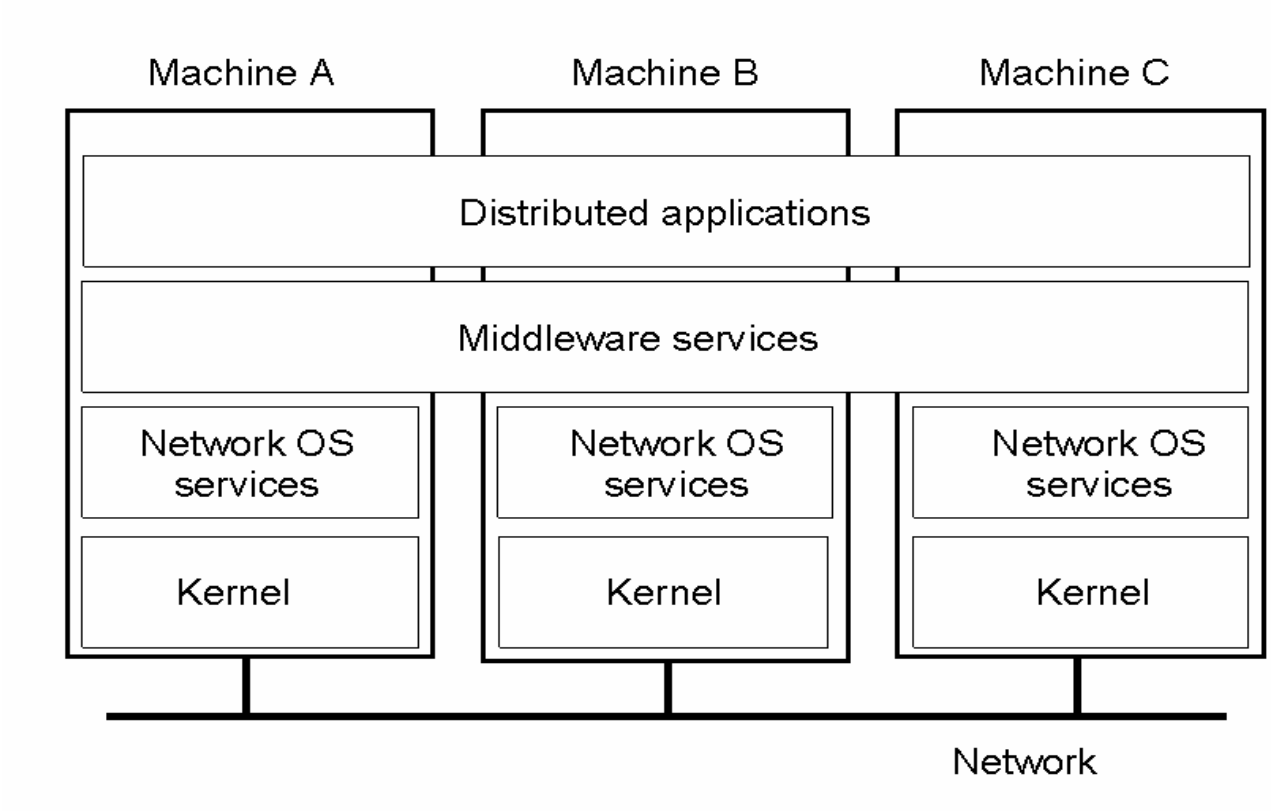
Network Operating System (3)

- Servers export and clients import (mount) files
- Different clients may mount the servers in different places



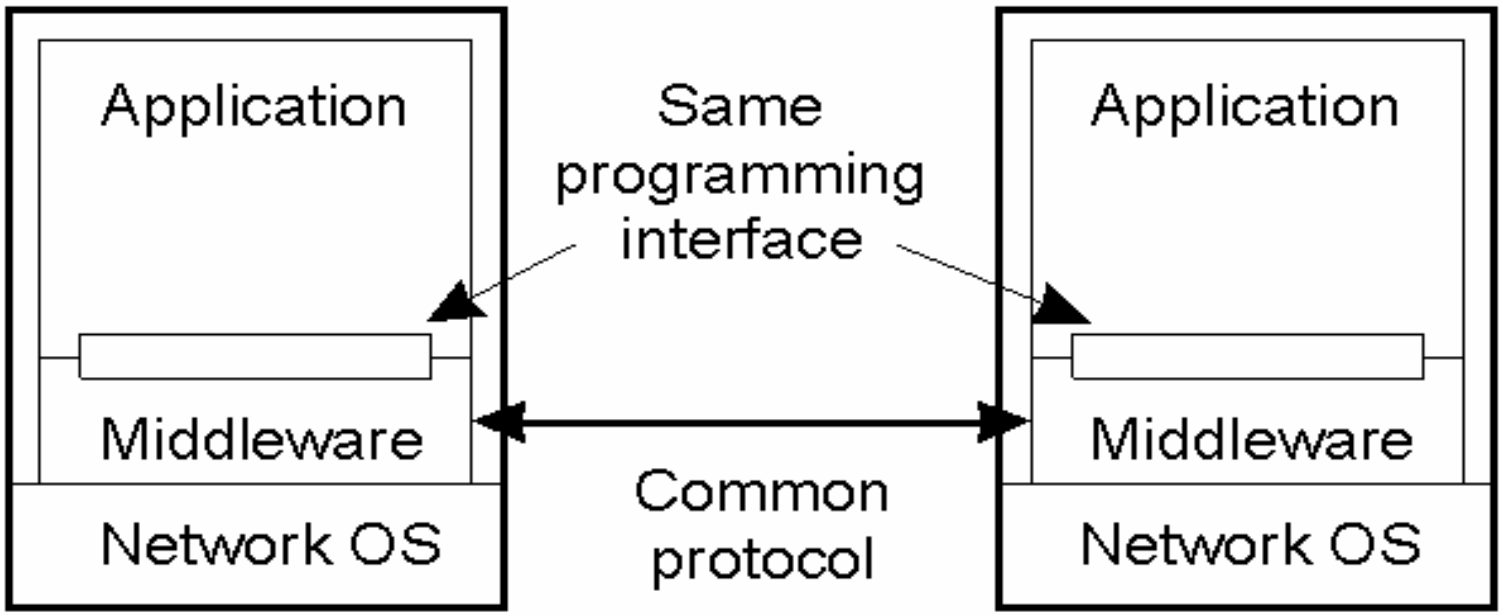
Positioning Middleware

- Middleware is the practical compromise among “true” distributed and network file system
- Additional layer provides *interoperability*



Middleware and Openness

- In an open middleware-based system following should be the same
  - The protocols used by each middleware layer
  - The services and interfaces they offer to applications

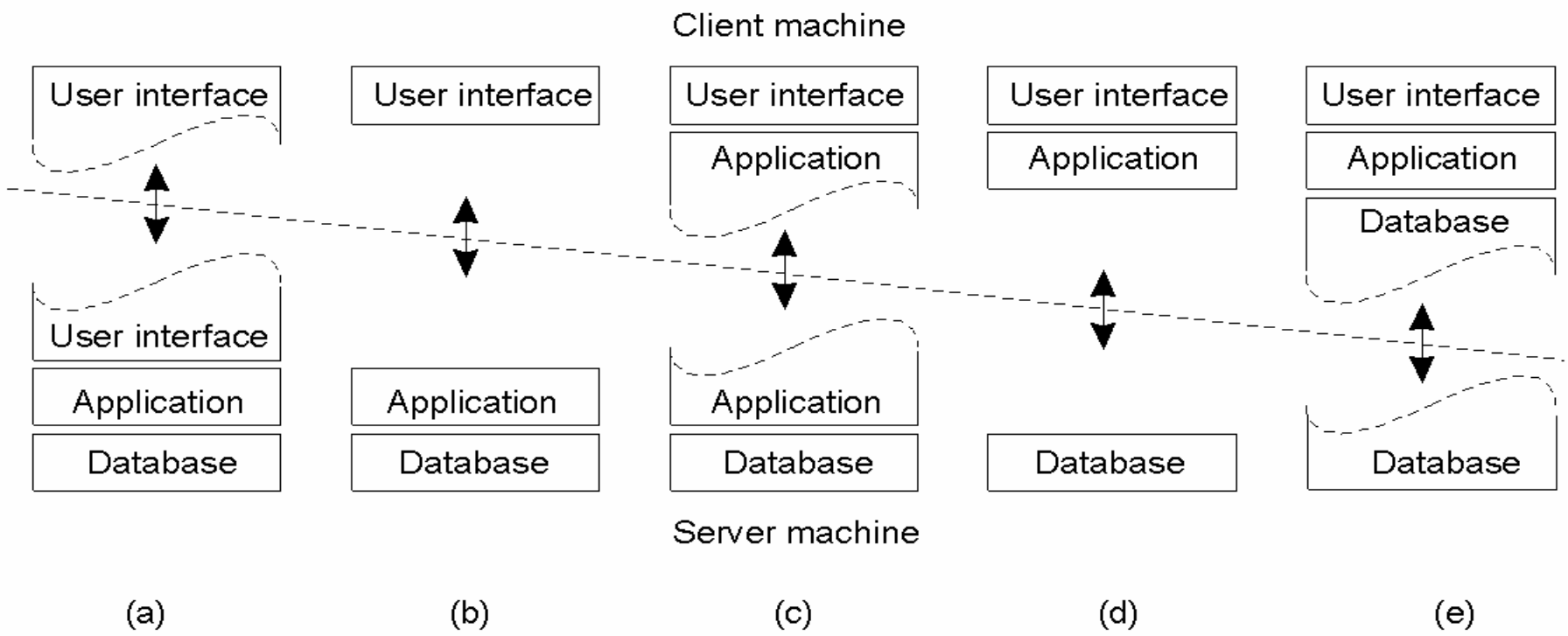


Comparison between Systems

Item	Distributed OS		Network OS	Middleware-based OS
	Multiproc.	Multicomp.		
Degree of transparency	Very High	High	Low	High
Same OS on all nodes	Yes	Yes	No	No
Number of copies of OS	1	N	N	N
Basis for communication	Shared memory	Messages	Files	Model specific
Resource management	Global, central	Global, distributed	Per node	Per node
Scalability	No	Moderately	Yes	Varies
Openness	Closed	Closed	Open	Open

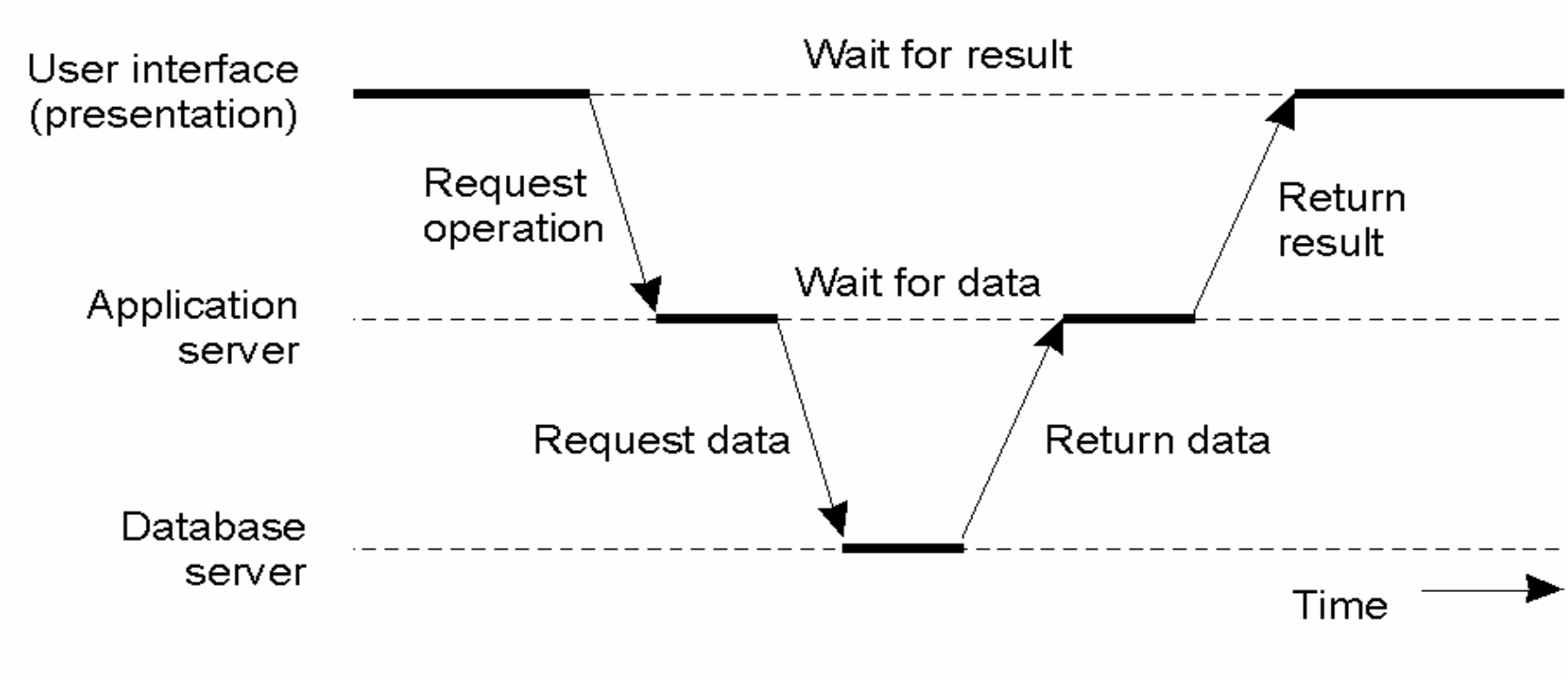
Multitiered Architectures (1)

- Alternative client-server organizations
  - In (a) only half of the user interface is running at the client, in (e) even half of the database management



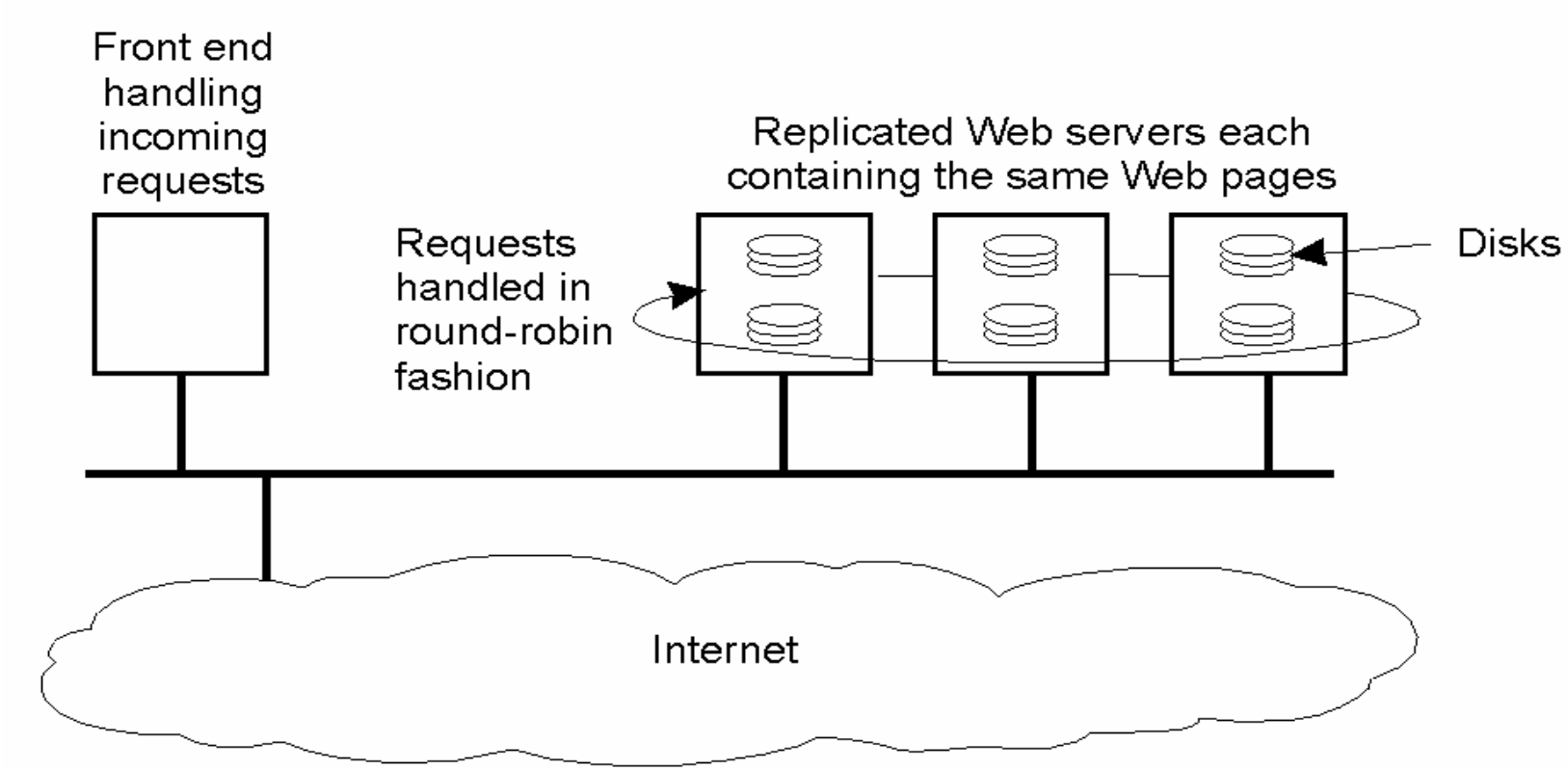
Multitiered Architectures (2)

- An example of a server acting as a client along the usual *vertical* distribution
  - User interface, business logic, data



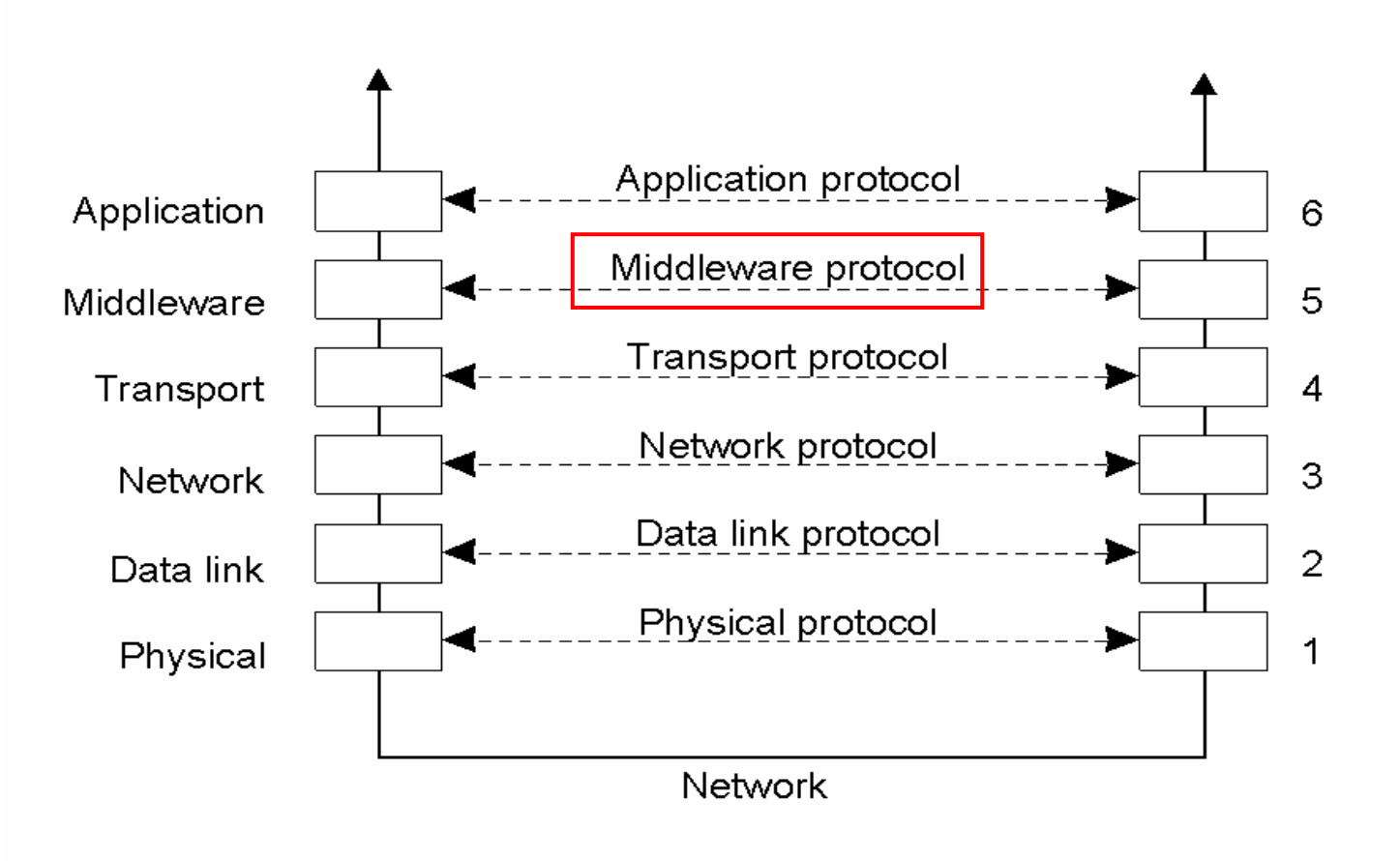
Modern Architectures

- An example of *horizontal* distribution of a Web service
  - The replication serves for scalability and enhanced performance



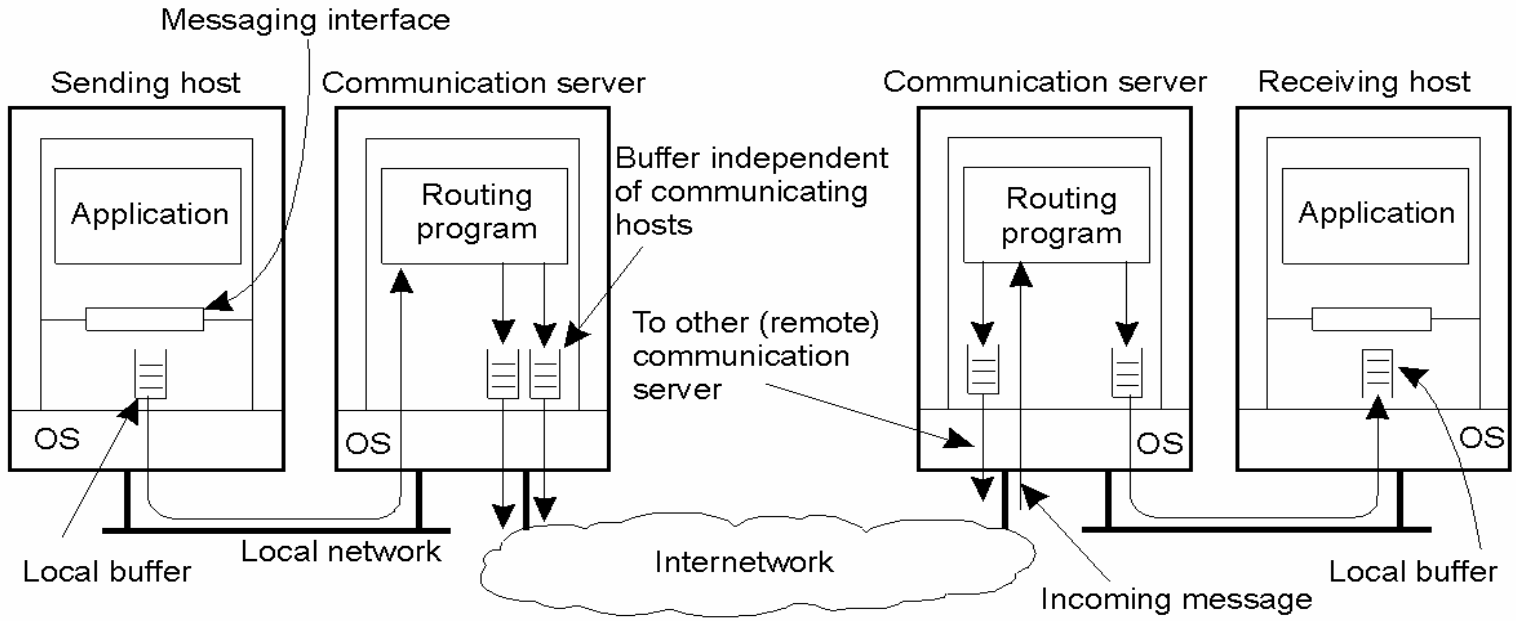
Middleware Protocols

- An adapted reference model for networked communication



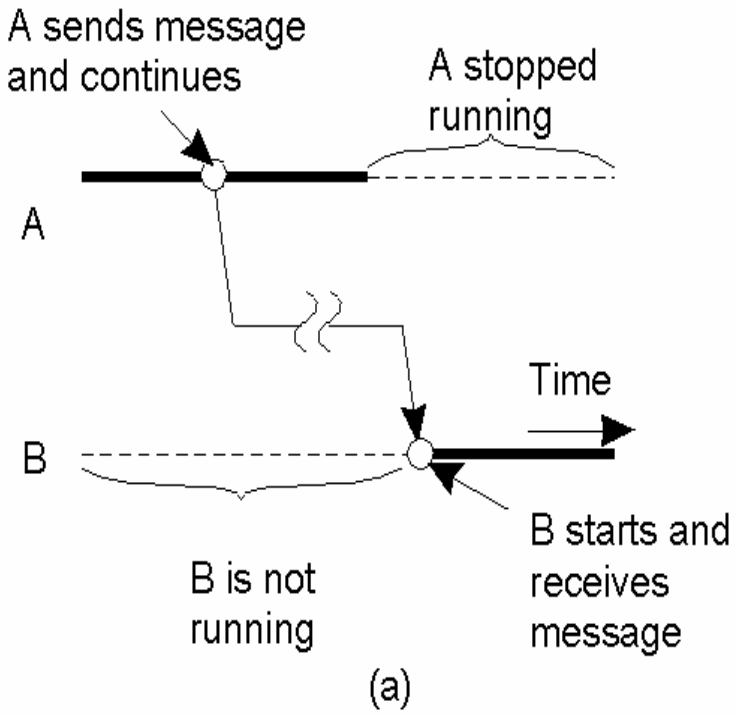
Persistence and Synchronicity in Communication (1)

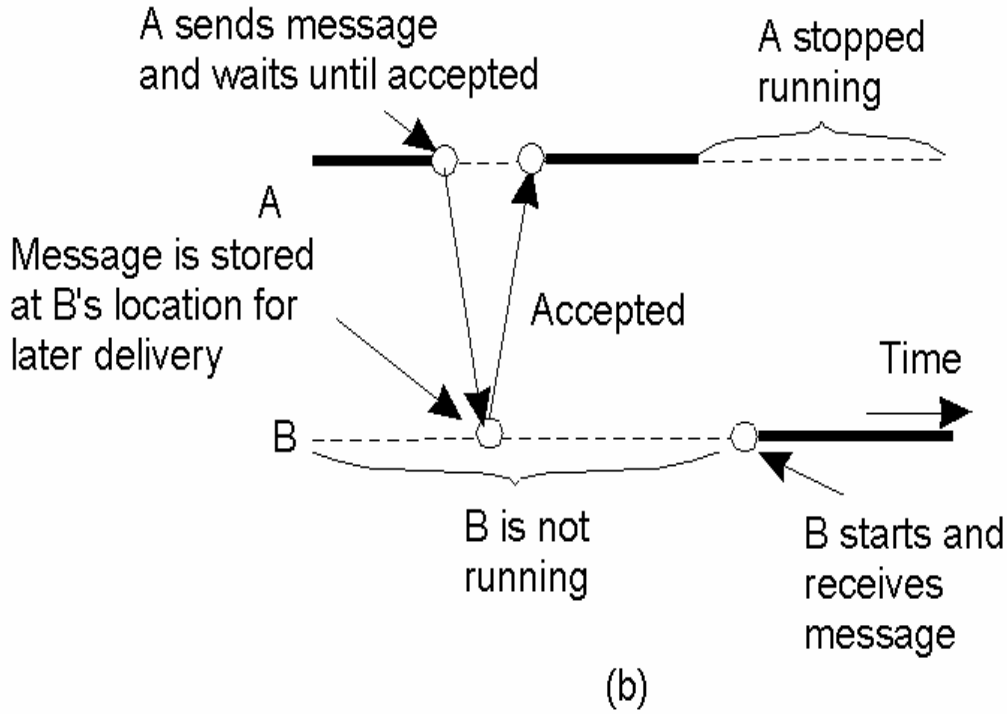
- The communication is *persistent* if the message is stored in the communication system, *transient* otherwise
- The communication is *synchronous* if the sender is blocked until the message reached "some" target (see later), *asynchronous* otherwise
- General architecture:



Persistence and Synchronicity in Communication (2)

a) Persistent asynchronous communication
b) Persistent synchronous communication

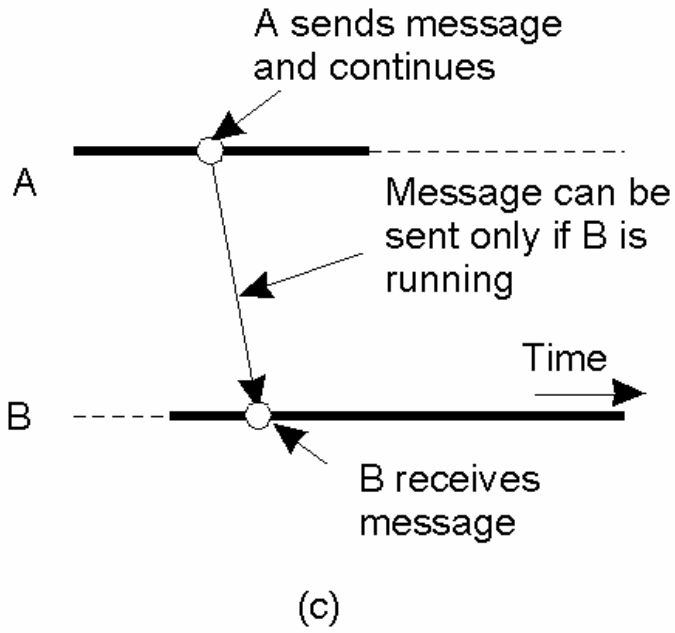


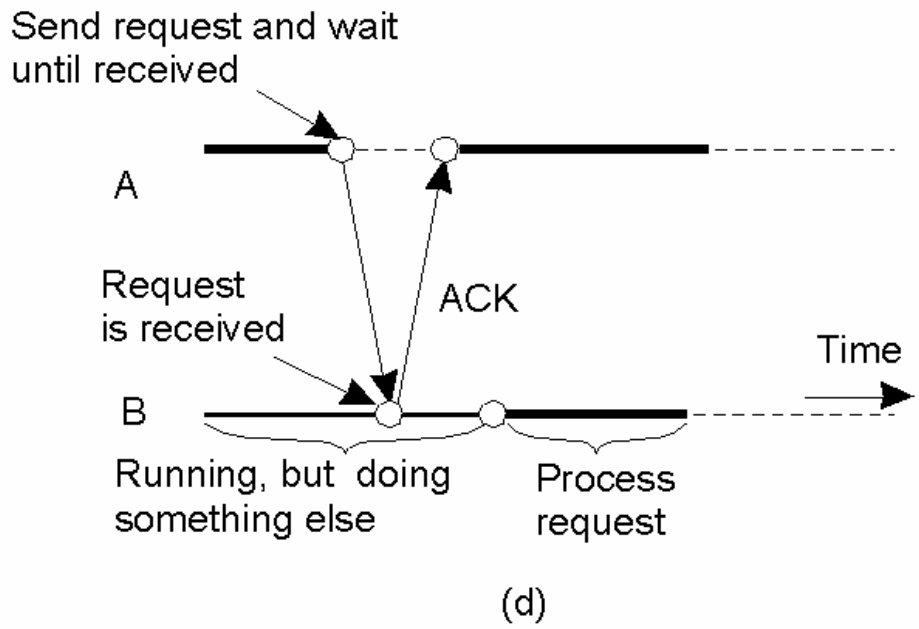


Persistence and Synchronicity in Communication (3)

c) Transient asynchronous communication

d) Receipt-based transient synchronous communication

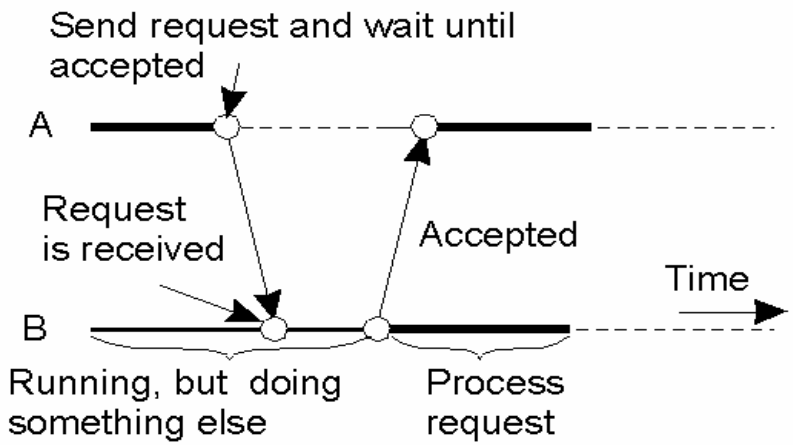




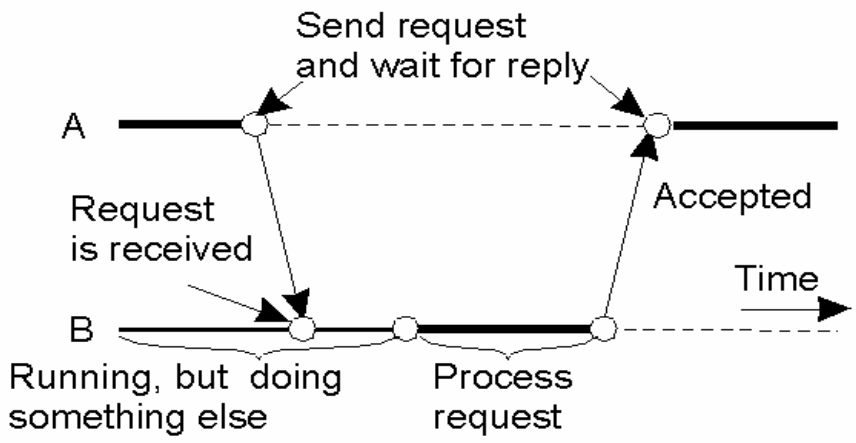
Persistence and Synchronicity in Communication (4)

e) Delivery-based transient synchronous communication

f) Response-based transient synchronous communication



(e)



(f)