Distributed Operating Systems

Introduction
Processes and Threads

Topics

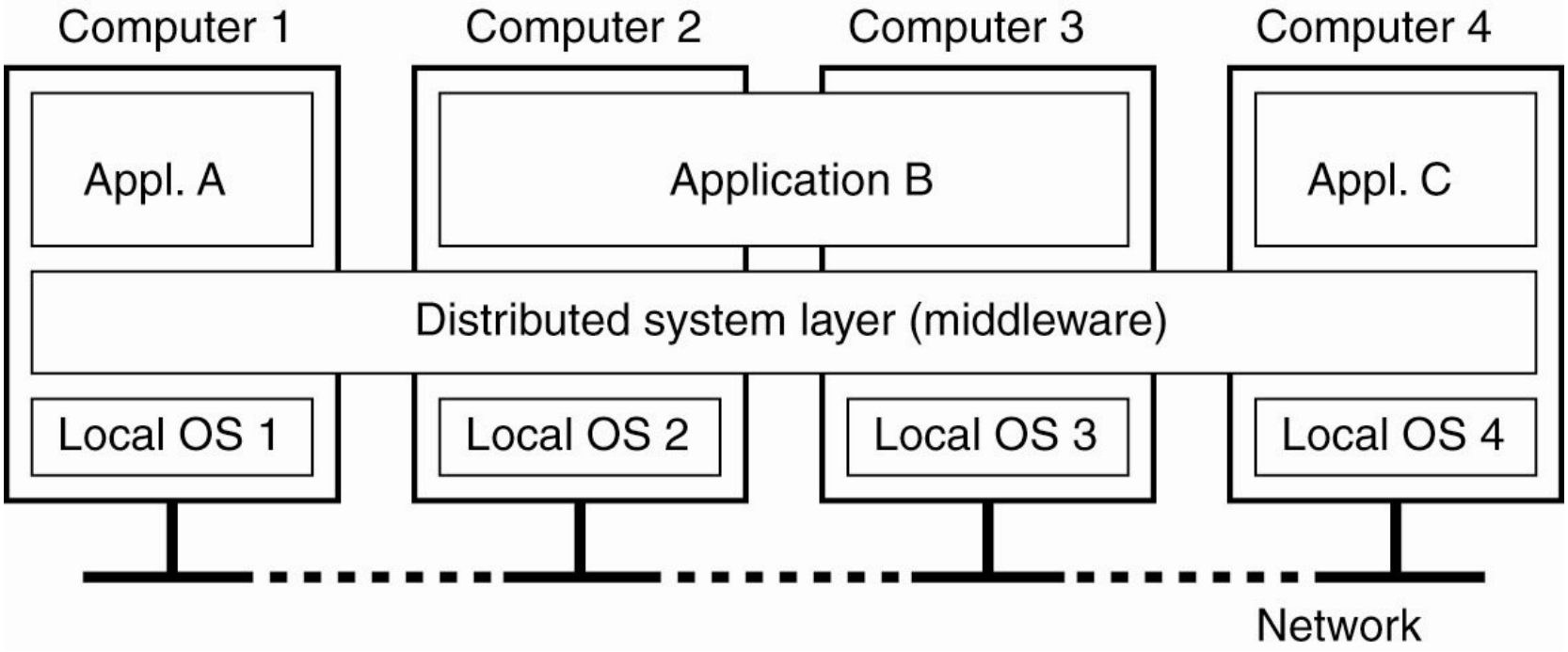
- Definition of a Distributed System
- Transparency in a Distributed System
- Scalability
- Types of Distributed Systems
- Transaction processing Systems
- Distributed Pervasive Systems
- Virtualization in Distributed Systems

Definition of a Distributed System (1)

- A distributed system is:

  A collection of independent computers that appears to its users as a single coherent system.

Definition of a Distributed System (2)



A distributed system organized as middleware. The middleware layer extends over multiple machines, and offers each application the same interface.

Transparency in a Distributed System

Transparency	Description
Access	Hide differences in data representation and how a resource is accessed
Location	Hide where a resource is located
Migration	Hide that a resource may move to another location
Relocation	Hide that a resource may be moved to another location while in use
Replication	Hide that a resource is replicated
Concurrency	Hide that a resource may be shared by several competitive users
Failure	Hide the failure and recovery of a resource

Different forms of transparency in a distributed system (ISO, 1995).

Scalability Problems

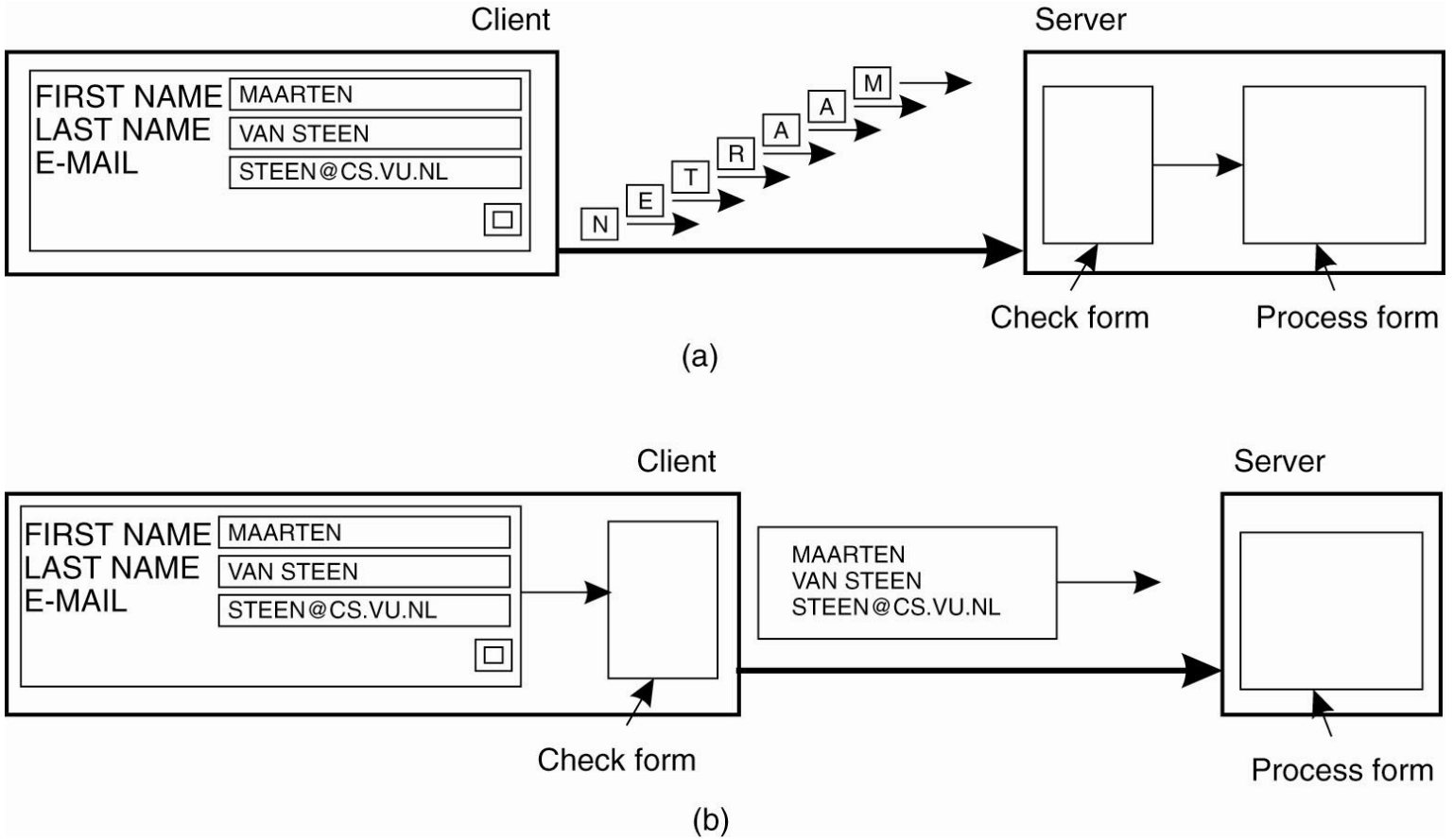
Concept	Example
Centralized services	A single server for all users
Centralized data	A single on-line telephone book
Centralized algorithms	Doing routing based on complete information

Examples of scalability limitations.

Scalability Problems

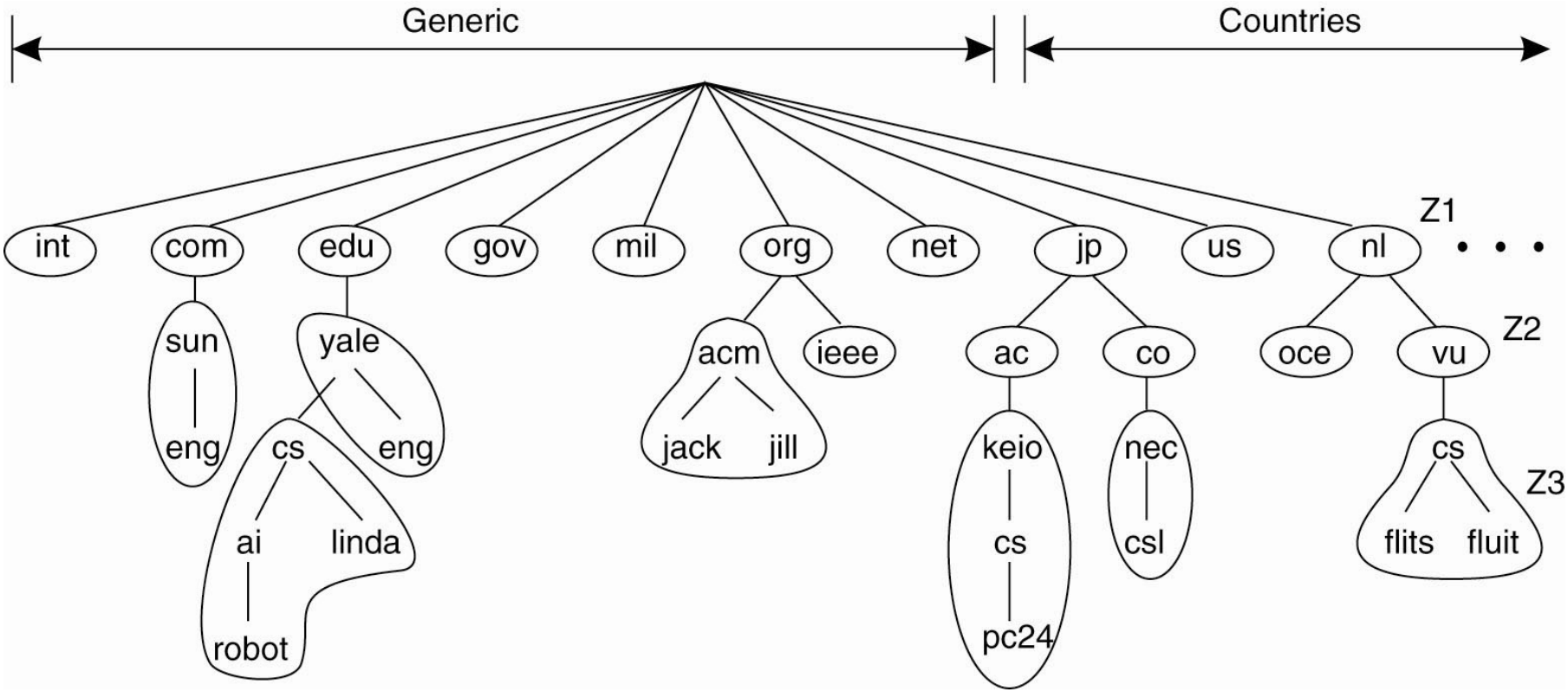
- Characteristics of decentralized algorithms:
  - No machine has complete information about the system state.
  - Machines make decisions based only on local information.
  - Failure of one machine does not ruin the algorithm.
  - There is no implicit assumption that a global clock exists.

Scaling Techniques (1)



The difference between letting (a) a server or (b) a client check forms as they are being filled.

Scaling Techniques (2)



An example of dividing the DNS name space into zones.

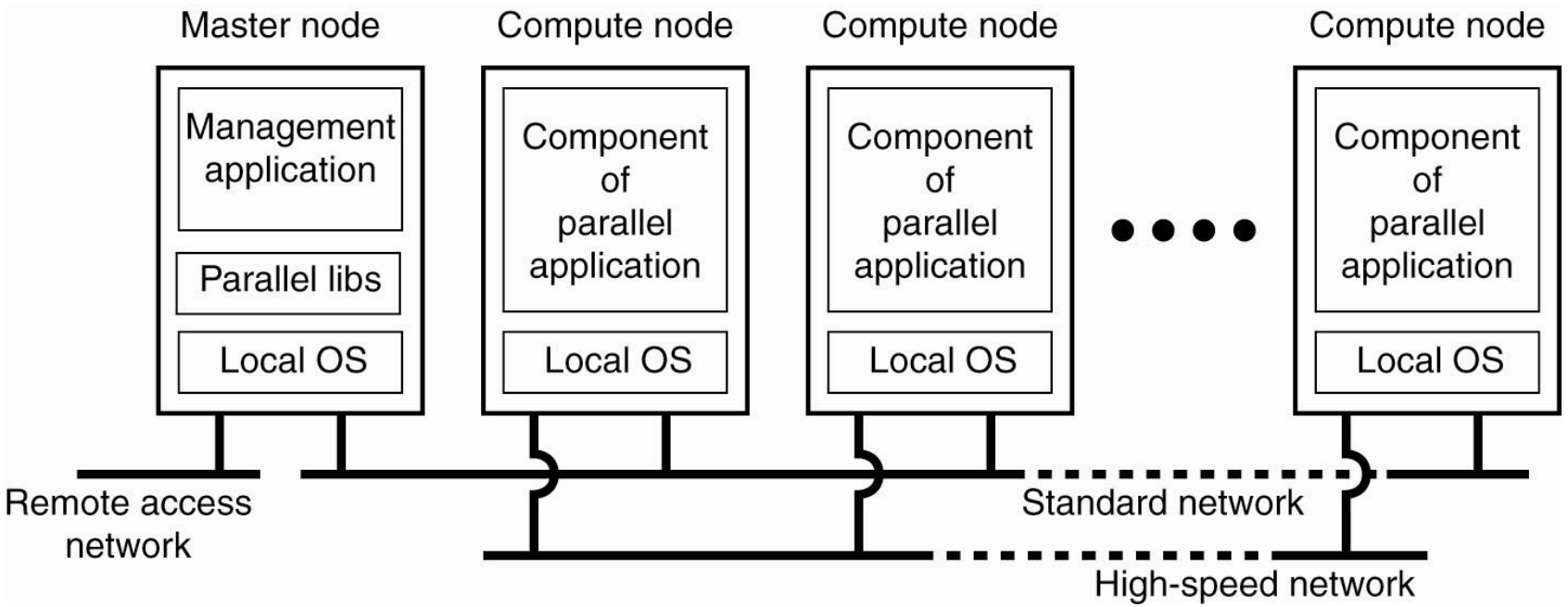
Pitfalls when Developing Distributed Systems

- False assumptions made by first time developer:
  - The network is reliable.
  - The network is secure.
  - The network is homogeneous.
  - The topology does not change.
  - Latency is zero.
  - Bandwidth is infinite.
  - Transport cost is zero.
  - There is one administrator.

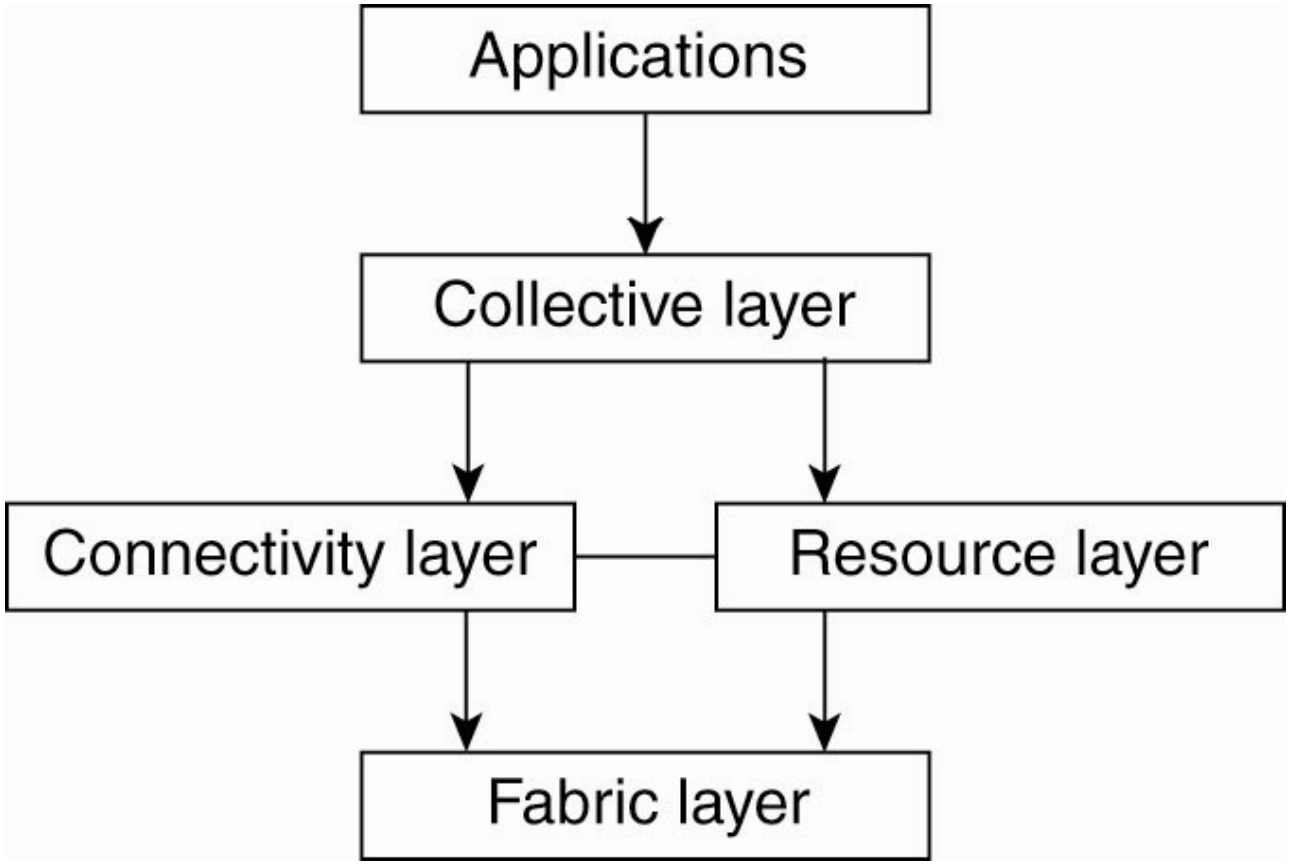
Types of Distributed Systems

Cluster Computing Systems

An example of a cluster computing system.



Grid Computing Systems



A layered architecture for grid computing systems.

Distributed Information Systems

- In many cases, a networked application simply consisted of a server running that application (often including a database) and making it available to remote clients.
- Integration at the lowest level would allow clients to wrap a number of requests, possibly for different servers, into a single larger request and have it executed as a distributed transaction.
- The key idea was that all, or none of the requests would be executed.

Transaction Processing Systems (1)

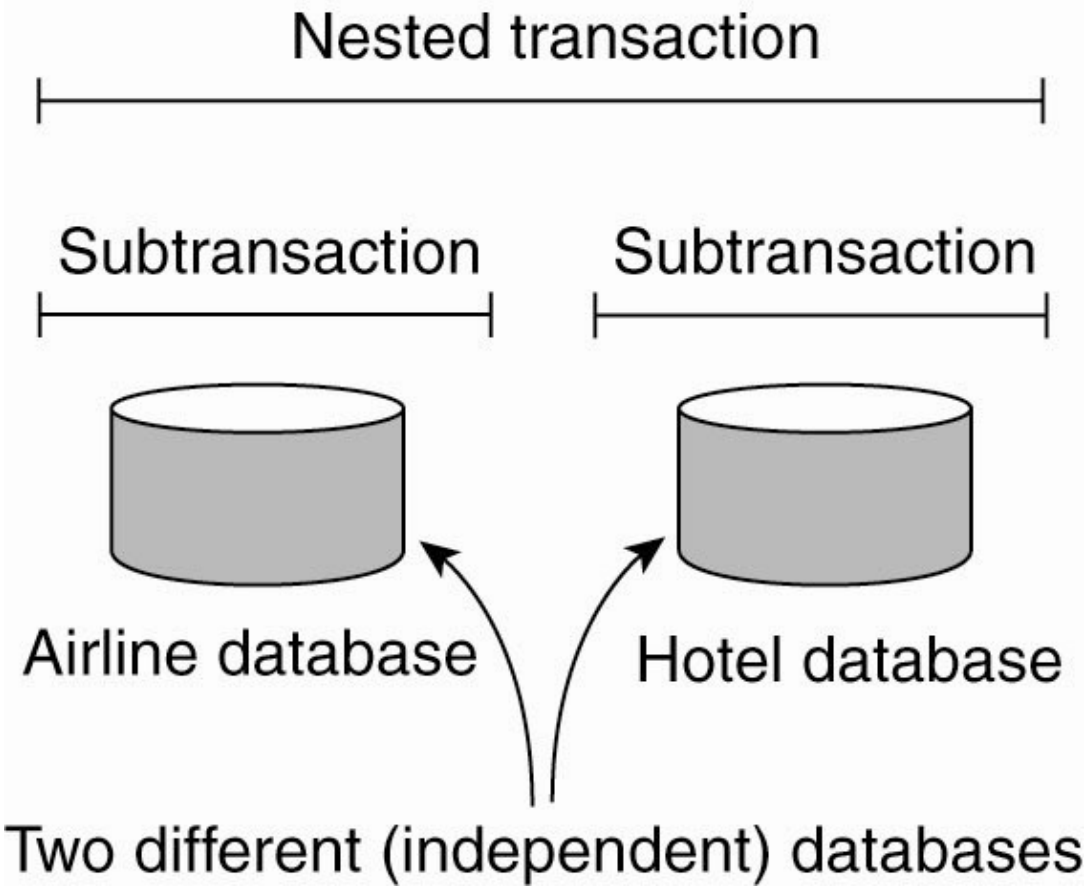
Primitive	Description
BEGIN_TRANSACTION	Mark the start of a transaction
END_TRANSACTION	Terminate the transaction and try to commit
ABORT_TRANSACTION	Kill the transaction and restore the old values
READ	Read data from a file, a table, or otherwise
WRITE	Write data to a file, a table, or otherwise

Example primitives for transactions.

Transaction Processing Systems (2)

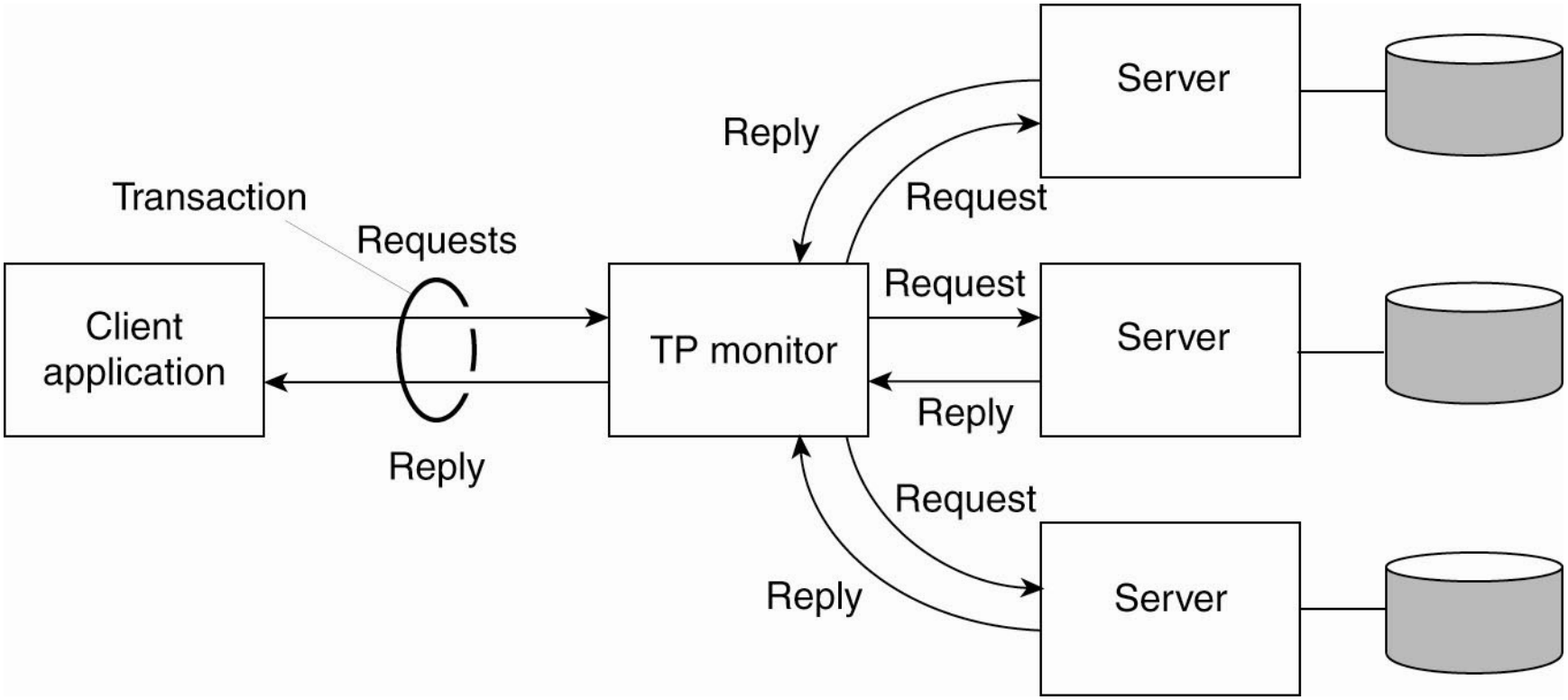
- Characteristic properties of transactions:
  - Atomic: To the outside world, the transaction happens indivisibly.
  - Consistent: The transaction does not violate system invariants.
  - Isolated: Concurrent transactions do not interfere with each other.
  - Durable: Once a transaction commits, the changes are permanent.

Transaction Processing Systems (3)

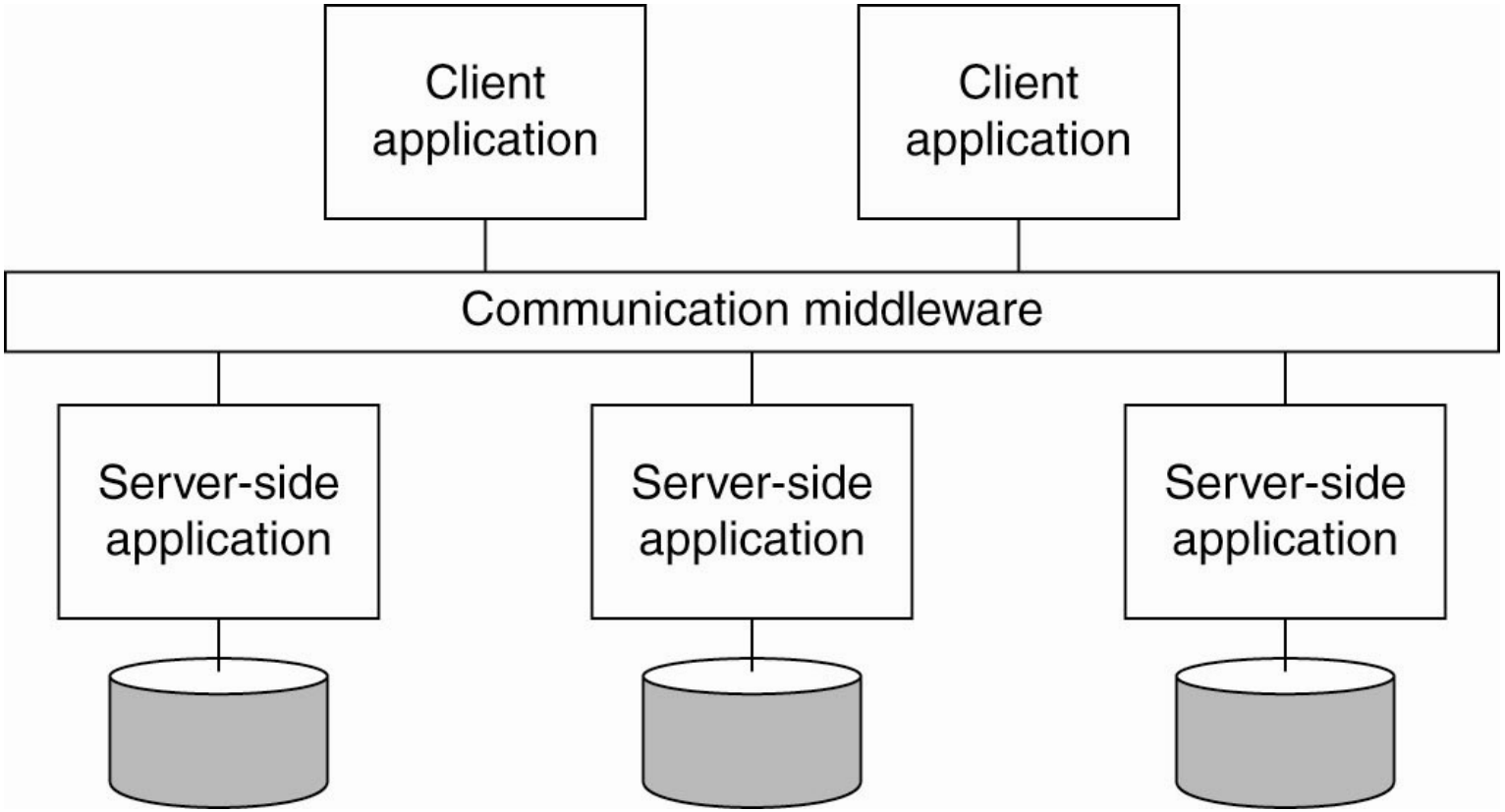


A nested transaction.

Transaction Processing Systems (4)



Enterprise Application Integration

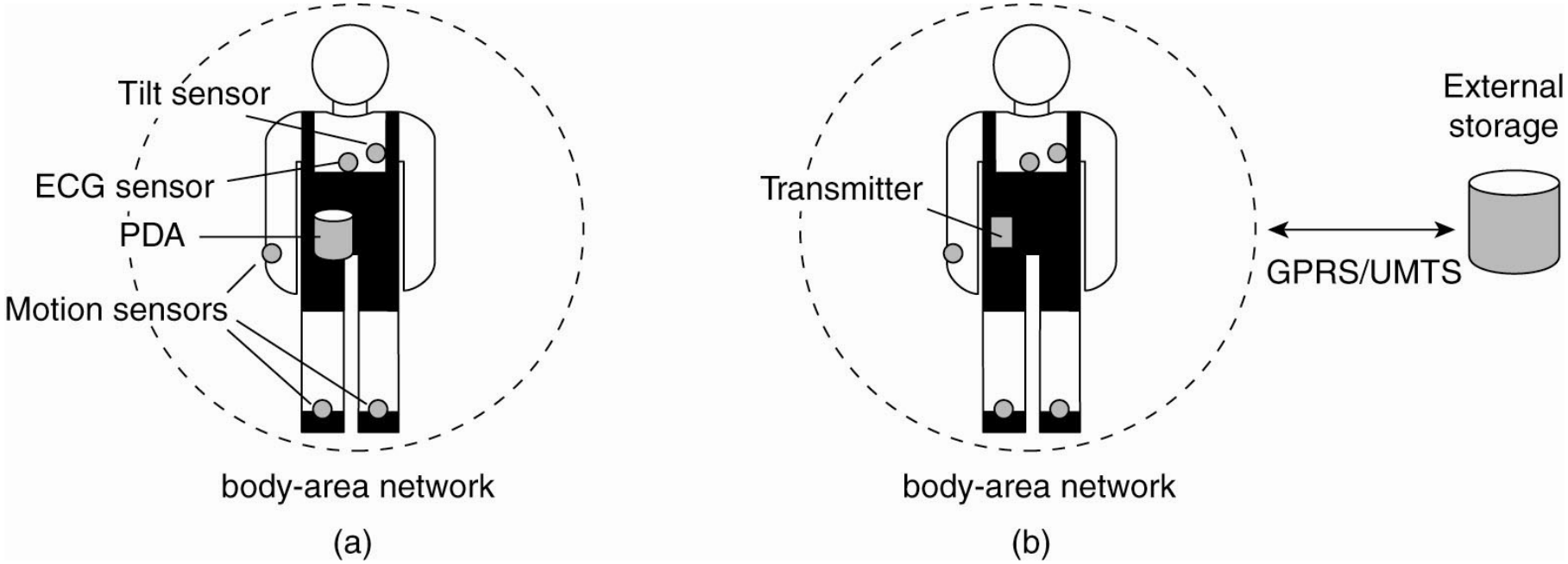


Middleware as a communication facilitator in enterprise application integration.

Distributed Pervasive Systems

- The distributed systems are largely characterized by their stability:
  - nodes are fixed
  - have a more or less permanent and high-quality connection to a network.
- Pervasive distributed systems are characterized by
  - being small,
  - battery-powered,
  - mobile,
  - and having only a wireless connection,
- ***although not all these characteristics apply to all devices***.

Electronic Health Care Systems (1)



Monitoring a person in a pervasive electronic health care system, using

(a) a local hub or

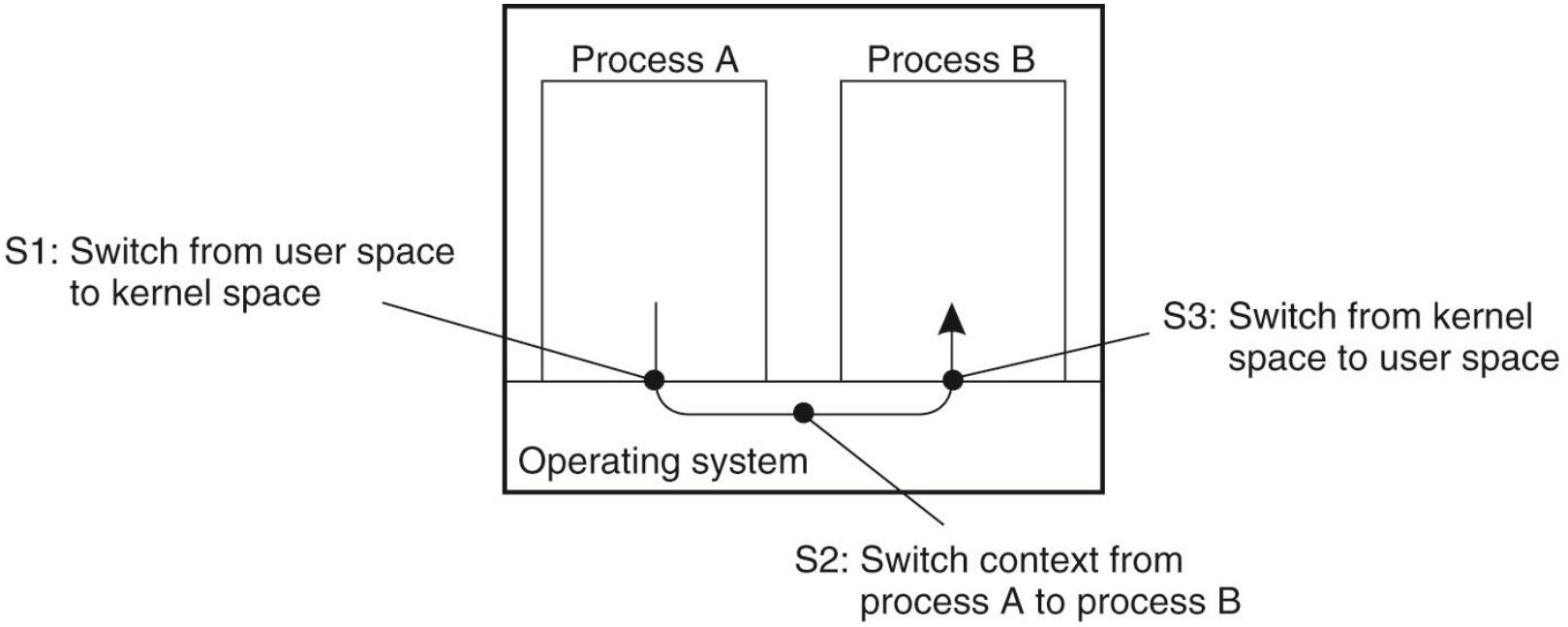
(b) a continuous wireless connection.

Electronic Health Care Systems (2)

- Questions to be addressed for health care systems:
  - Where and how should monitored data be stored?
  - How can we prevent loss of crucial data?
  - What infrastructure is needed to generate and propagate alerts?
  - How can physicians provide online feedback?
  - How can extreme robustness of the monitoring system be realized?
  - What are the security issues and how can the proper policies be enforced?

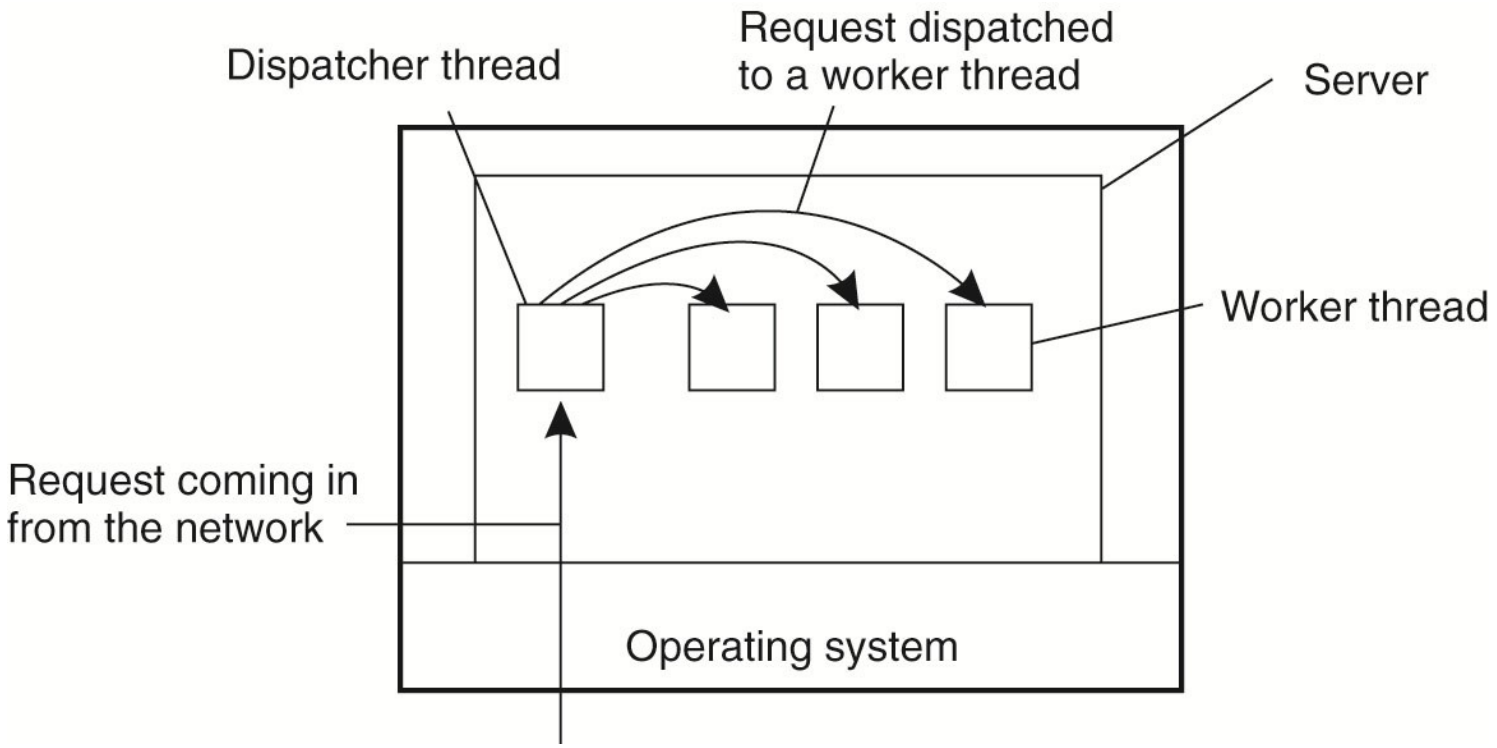
Threads and Processes

Thread Usage in Nondistributed Systems



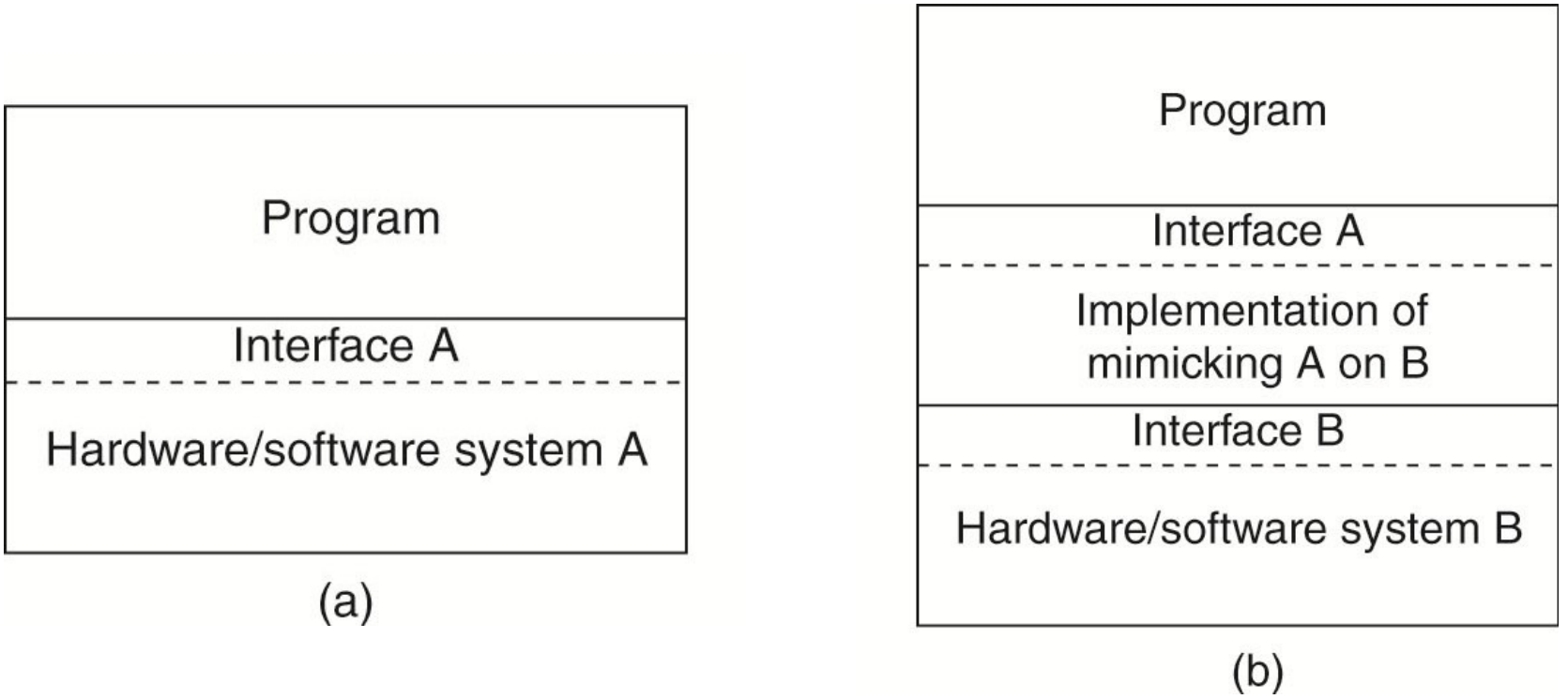
Context switching as the result of IPC.

Multithreaded Servers



A multithreaded server organized in a dispatcher/worker model.

The Role of Virtualization in Distributed Systems



- (a) General organization between a program, interface, and system. (b) General organization of virtualizing system A on top of system B.

Architectures of Virtual Machines (1)

- Interfaces at different levels

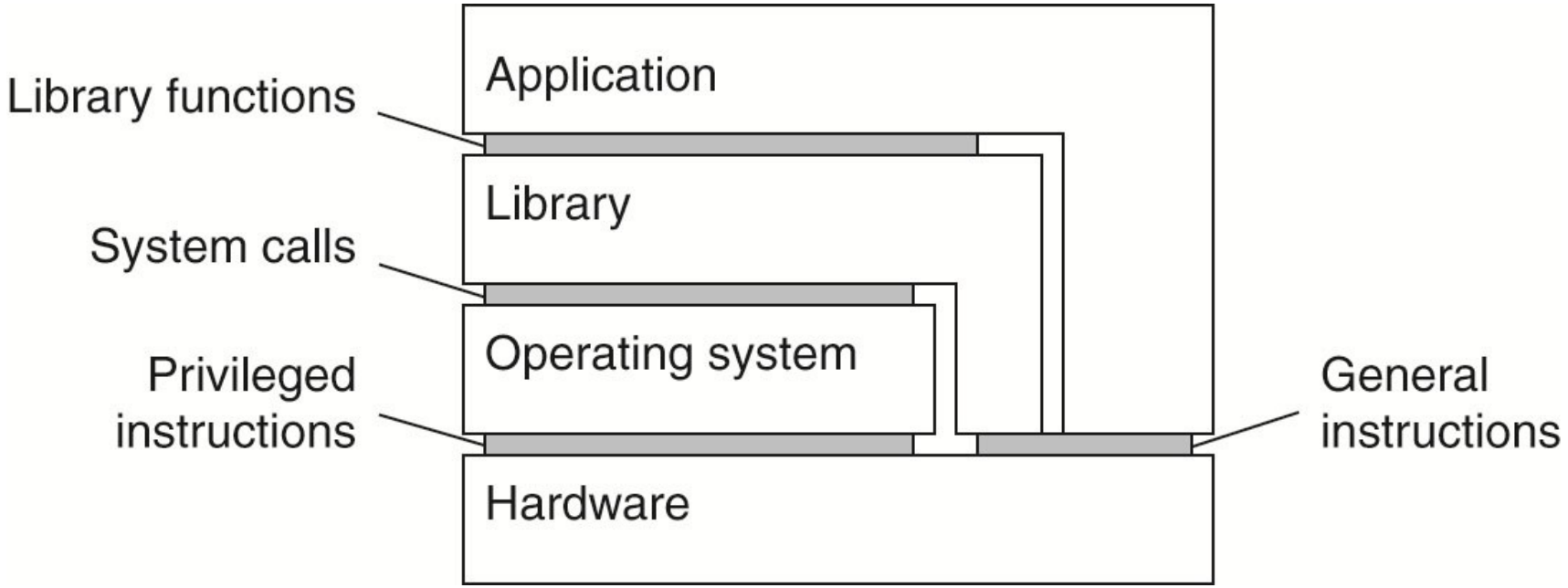
- An interface between the hardware and software consisting of machine instructions that can be invoked by any program.
- An interface between the hardware and software, consisting of machine instructions that can be invoked only by privileged programs, such as an operating system.

Architectures of Virtual Machines (2)

- Interfaces at different levels
  - An interface consisting of system calls as offered by an operating system.
  - An interface consisting of library calls generally forming what is known as an application programming interface (API).
  - (In many cases, the aforementioned system calls are hidden by an API.)

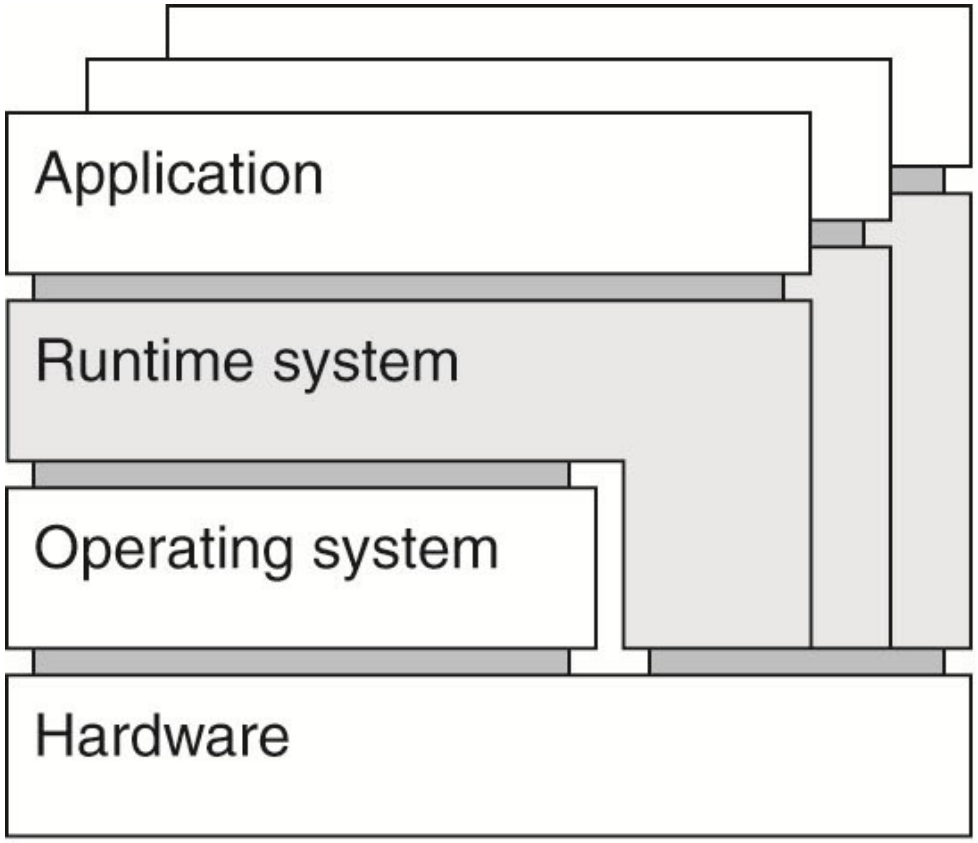
Architectures of Virtual Machines (3)

- Various interfaces offered by computer systems.



Architectures of Virtual Machines (4)

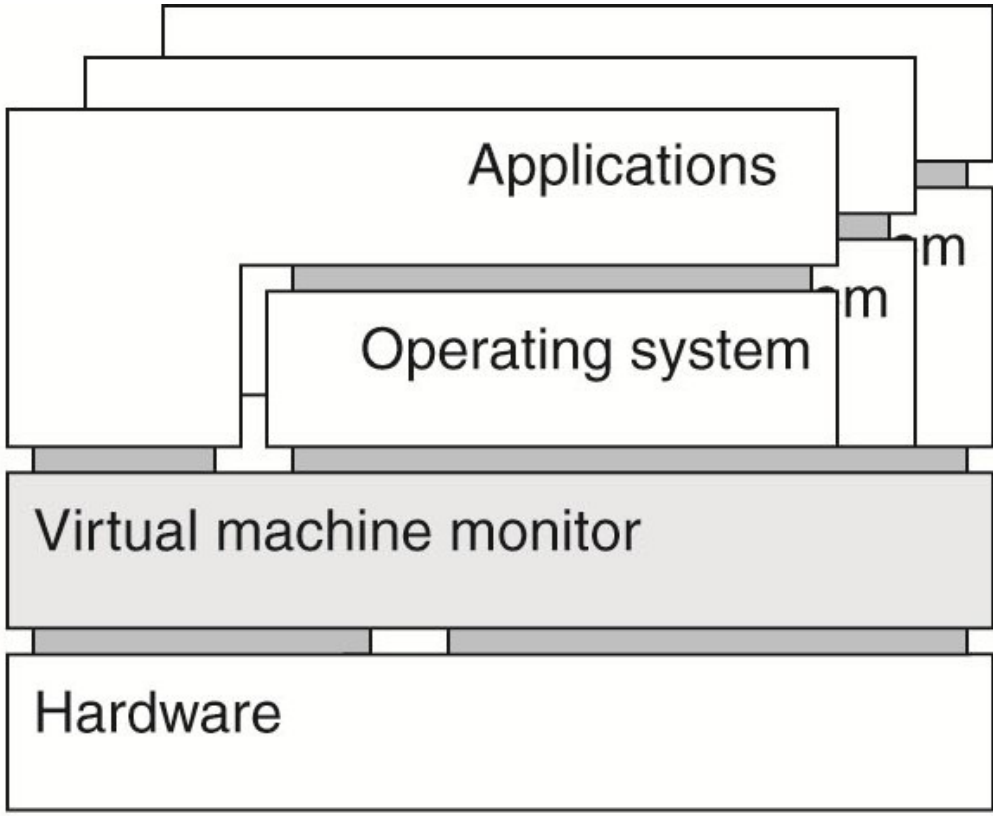
- A process virtual machine, with multiple instances of (application, runtime) combinations.



(a)

Architectures of Virtual Machines (5)

- A virtual machine monitor, with multiple instances of (applications, operating system) combinations.



(b)

Questions?