Distributed Systems

Introduction

Chapter 1

Course/Slides Credits

Note: all course presentations are based on those developed by Andrew S. Tanenbaum and Maarten van Steen. They accompany their "Distributed Systems: Principles and Paradigms" textbook (1st & 2nd editions). http://www.prenhall.com/divisions/esm/app/author_tanenbaum/custom/dist_sys_1e/index.html

And additions made by Paul Barry in course CW046-4: Distributed Systems

http://glasnost.itcarlow.ie/~barryp/net4.html

Outline

- ❖What is a distributed system?
- ❖Design Goals
- ❖Pitfalls when Developing Distributed Systems
- ❖Types of Distributed Systems

Definition of a Distributed System (1)

A distributed system is:

A distributed system is a collection of autonomous computing elements that appears to its users as a single coherent system.

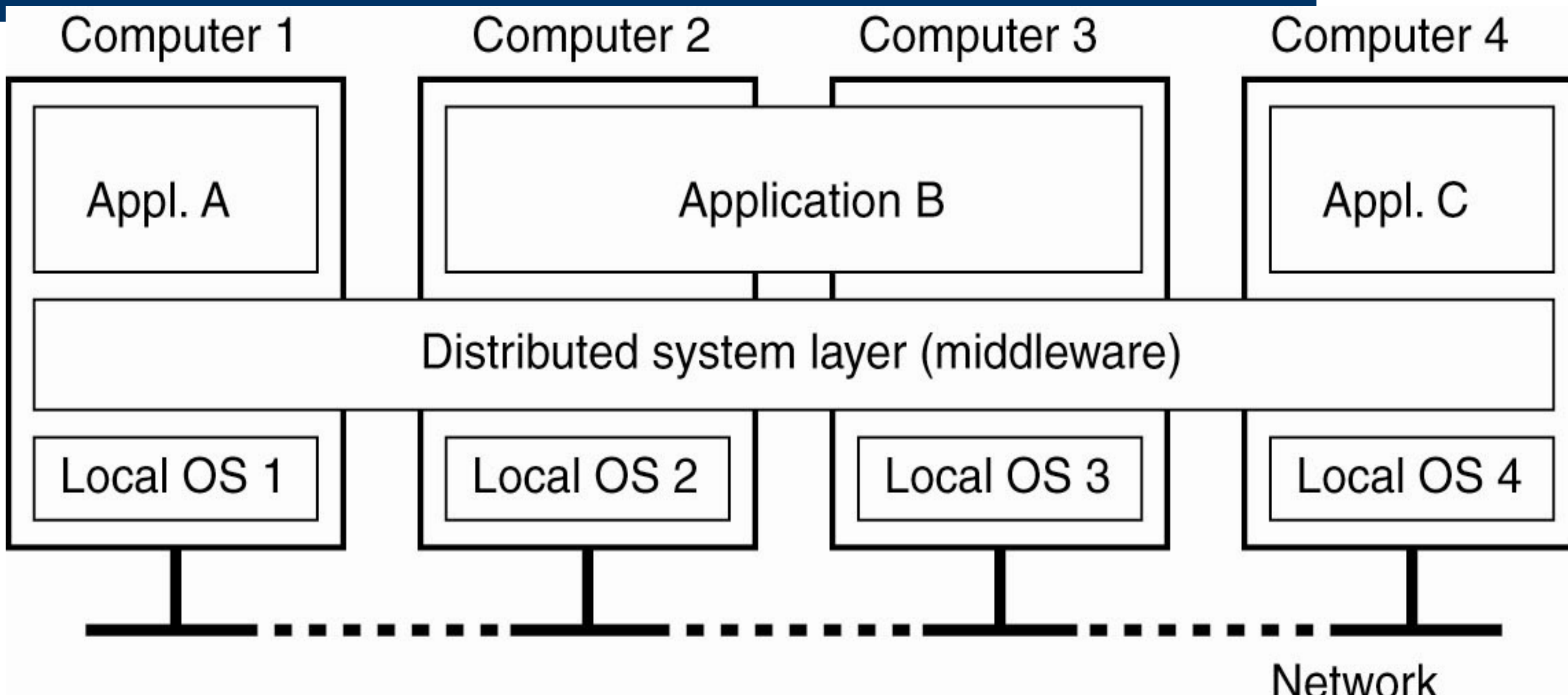
Characteristic 1: Collection of autonomous computing elements

- **Autonomuous computing nodes**
- **Independent behavior: No global time- Need of synchronization**
- **Group membership issues.**

Characteristic 2: Single coherent system

- **User or application perceive a single system -->Nodes need to collaborate**
- **The collection of nodes as a whole operates the same, no matter where, when, and how interaction between a user and the system takes place.**
- **End user would not be able to tell exactly on which computer a process is Currently executing / part of a task has been spawned off to another process / where data is stored or replicated should be of no concern**

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. It hides the differences in hardware and OSs.
- Contains commonly used components or functions that need not to be implemented by each applications.

Definition of a Distributed System (2)

- Middleware = OS of distributed System
  - Manager of resources offering its applications to efficiently share and deploy those resources across a network.
  - Facilities for inter-application communication.
  - Security services.
  - Accounting services.
  - Masking of and recovery from failures.
- Difference with OS is that their services are offered in a networked environment.

Goals of Distributed Systems

- Easily Connect Users/Resources and support resource sharing.
- Exhibit Distribution Transparency
- Support Openness
- Be Scalable:
  - in size
  - geographically
  - administratively

Looking at these goals helps use answer the question: “Is building a distributed system worth the effort?”

Connect Users/Resources and support resource sharing.

- Resources : storage, services, data…
  - Example: file sharing peer-to-peer; Bittorrent
- Cheaper to have a single reliable storage facility
- Connecting makes collaboration easy.

Transparency in a Distributed System

Hide the fact that processes and resources are physically distributed across different computers.

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)

Openess

Able to interact with services from other open systems, irrespective of the underlying environment:

- Well defined interface.
- Easily interoperate – coexist with others
- Support portability
- Easily extensible – Add functionality/component

Scalability in Distributed Systems

- Size scalability: Number of users/processes
- Geographical scalability: Maximum distance between nodes
- Administrative scalability: Number of administrative domains

Scalability Limitations

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

Scaling Techniques (1)

Scaling is limited by the servers and network capacity.

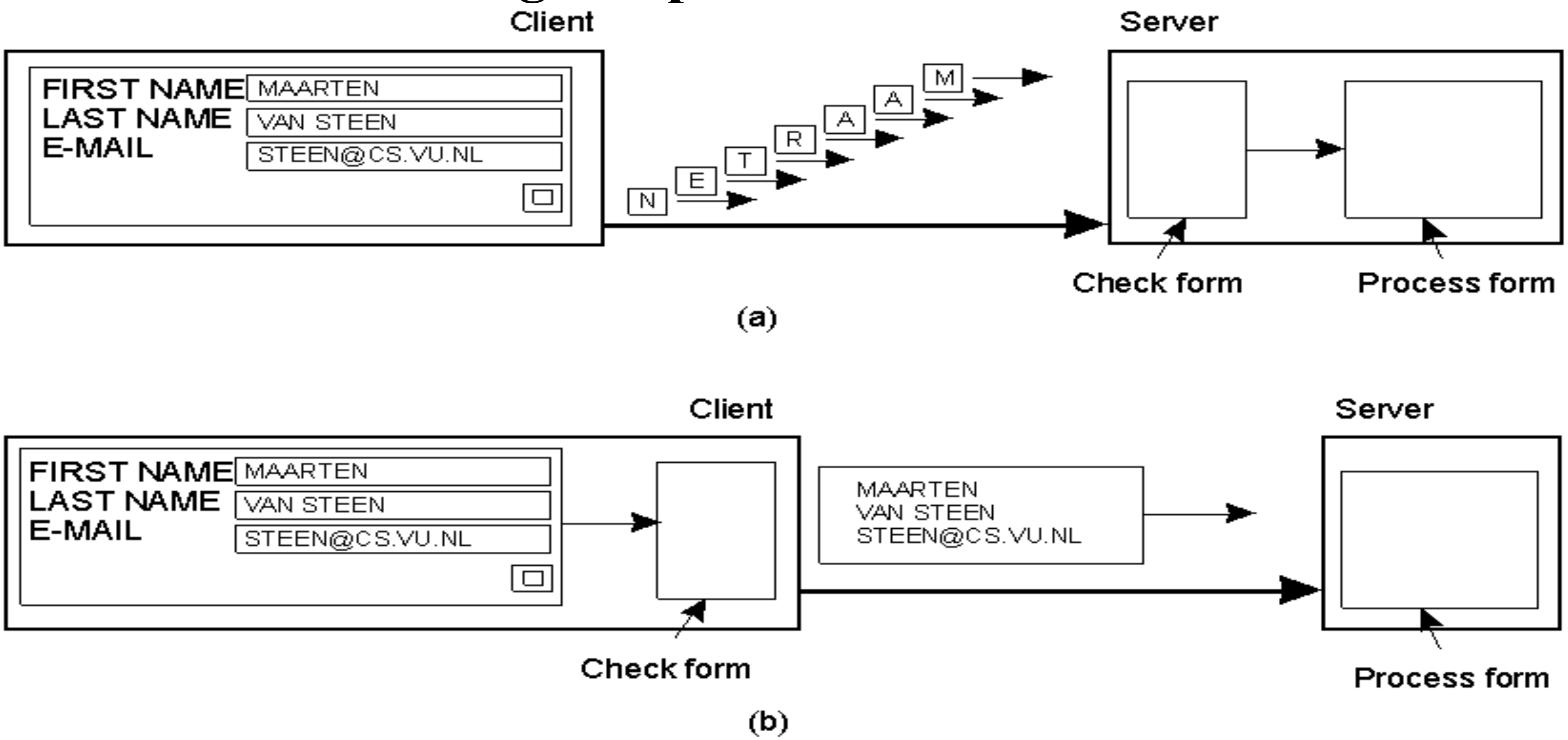
Solutions:

Scaling up: Increase the capacities (CPU, memory, network modules..)

Scaling out: expanding the DS by deploying more machines.

Scaling Techniques (1)

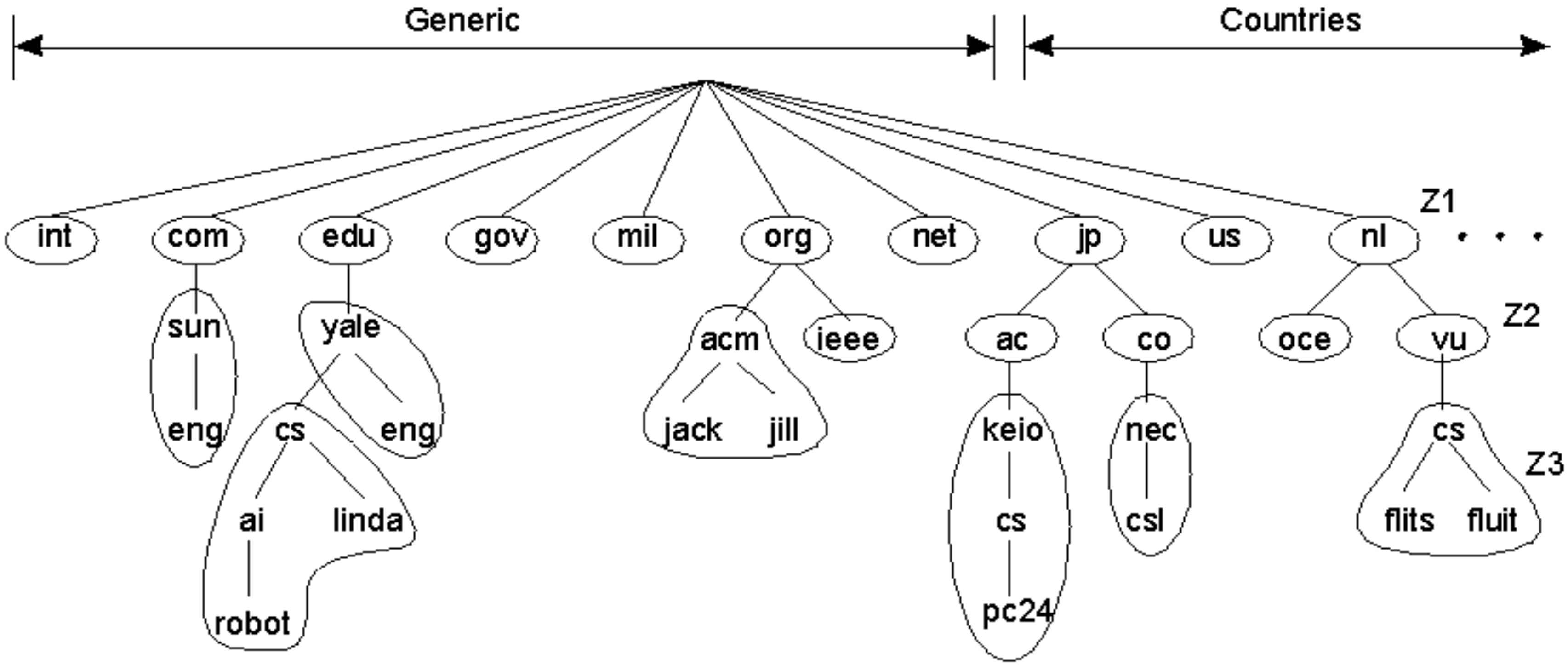
Moving computation to client.



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

Scaling Techniques (2)

Partition data and computation across multiple machine
Replication and caching (problems of inconsistency)



An example of dividing the DNS name space into zones

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.

Pitfalls when Developing Distributed Systems

- 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

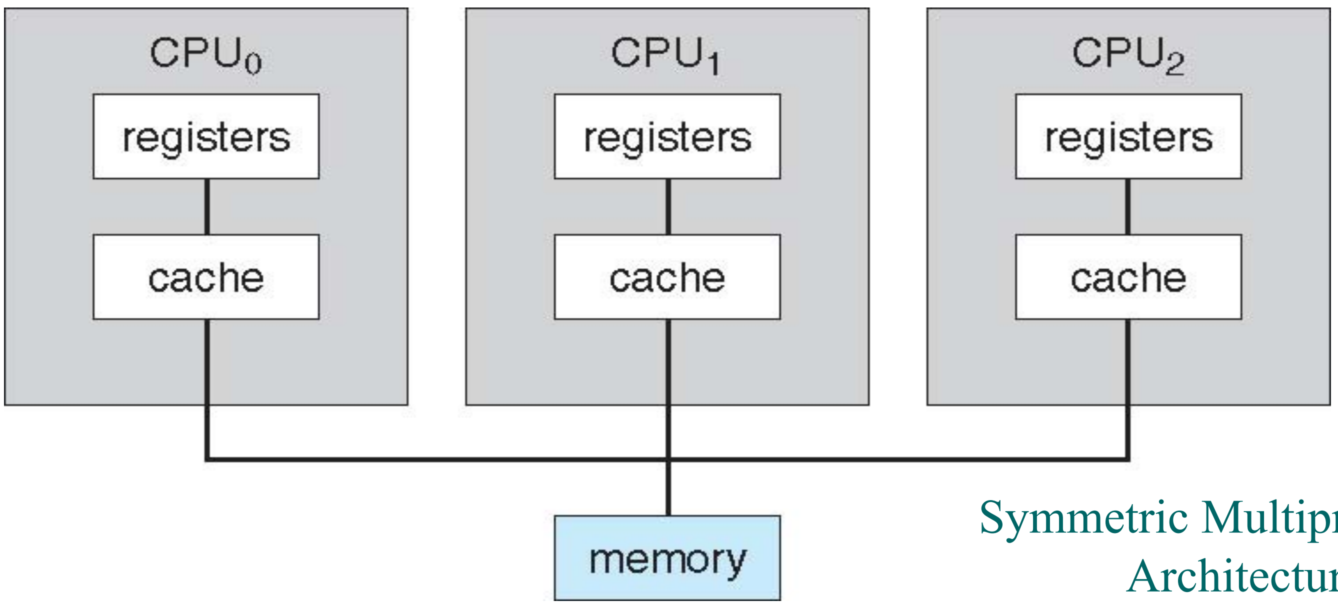
Types of Distributed Systems

- Distributed Computing Systems
  - High Performance Computing (HPC)
- Distributed Information Systems
  - Transaction Processing Systems (TPS)
  - Enterprise Application Integration (EAI)
- Distributed Pervasive Systems
  - Ubiquitous Systems

- Recall OS

Types of Multiprocessor Systems

- Symmetric multiprocessing
  - Each processor performs all tasks within OS
  - No master-slave relationship exists between processors



Symmetric Multiprocessing Architecture

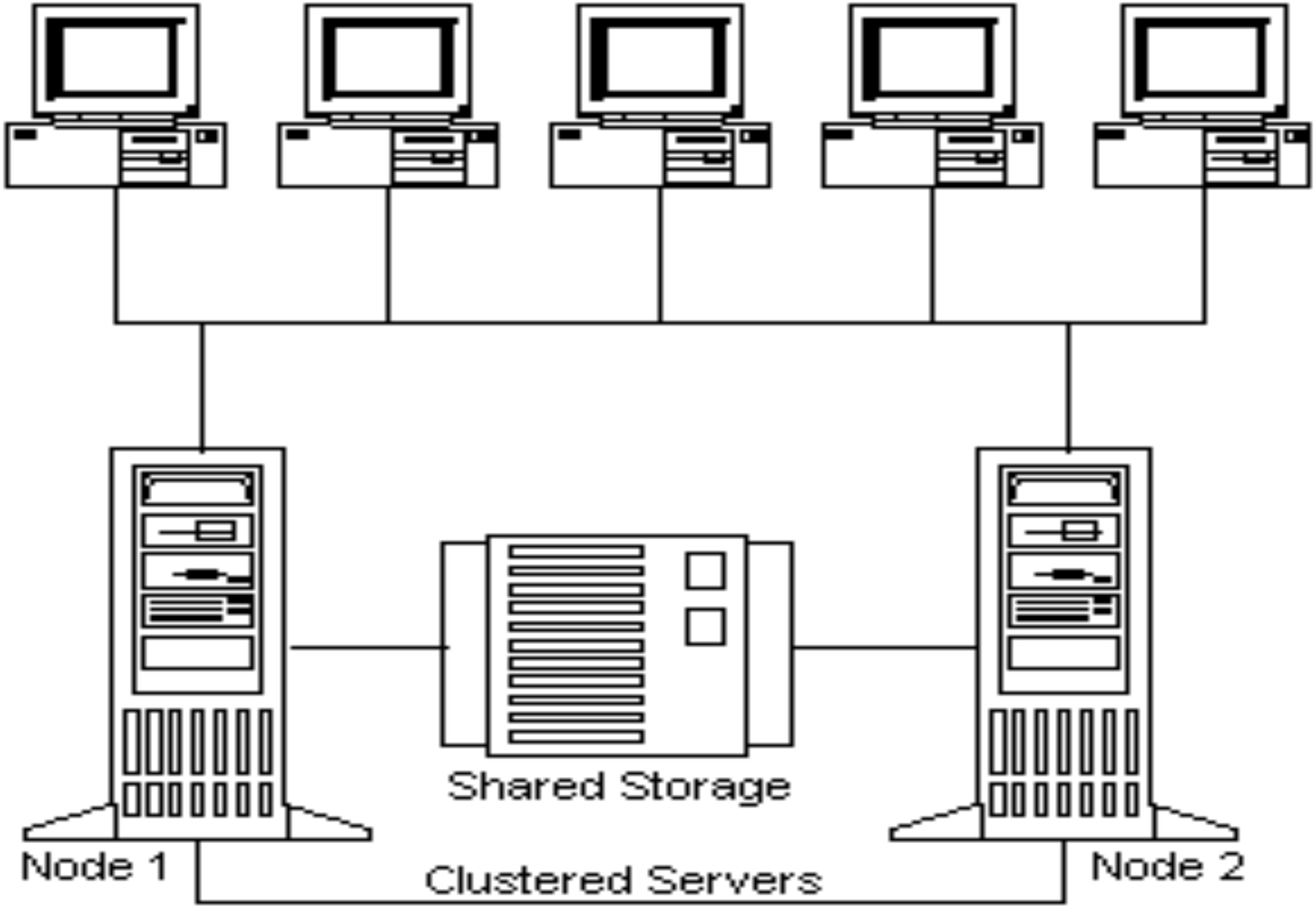
- Asymmetric multiprocessing
  - Master processor controls and allocates work to the slave processors
  - More common in extremely large systems

Clustered Systems

- Like multiprocessor systems, but multiple systems working together
  - Usually sharing storage via a **storage-area network (SAN)** & closely linked via a **LAN**
  - Provides a **high-availability** service which survives failures
    - **Asymmetric clustering** has one machine in hot-standby mode monitoring the active server while the other is running the applications
    - **Symmetric clustering** has multiple nodes running applications, monitoring each other
  - Some clusters are for **high-performance computing (HPC)**
    - Applications must be written to use **parallelization** (dividing a program into separate components that run in parallel on individual PCs)

- END Recall OS

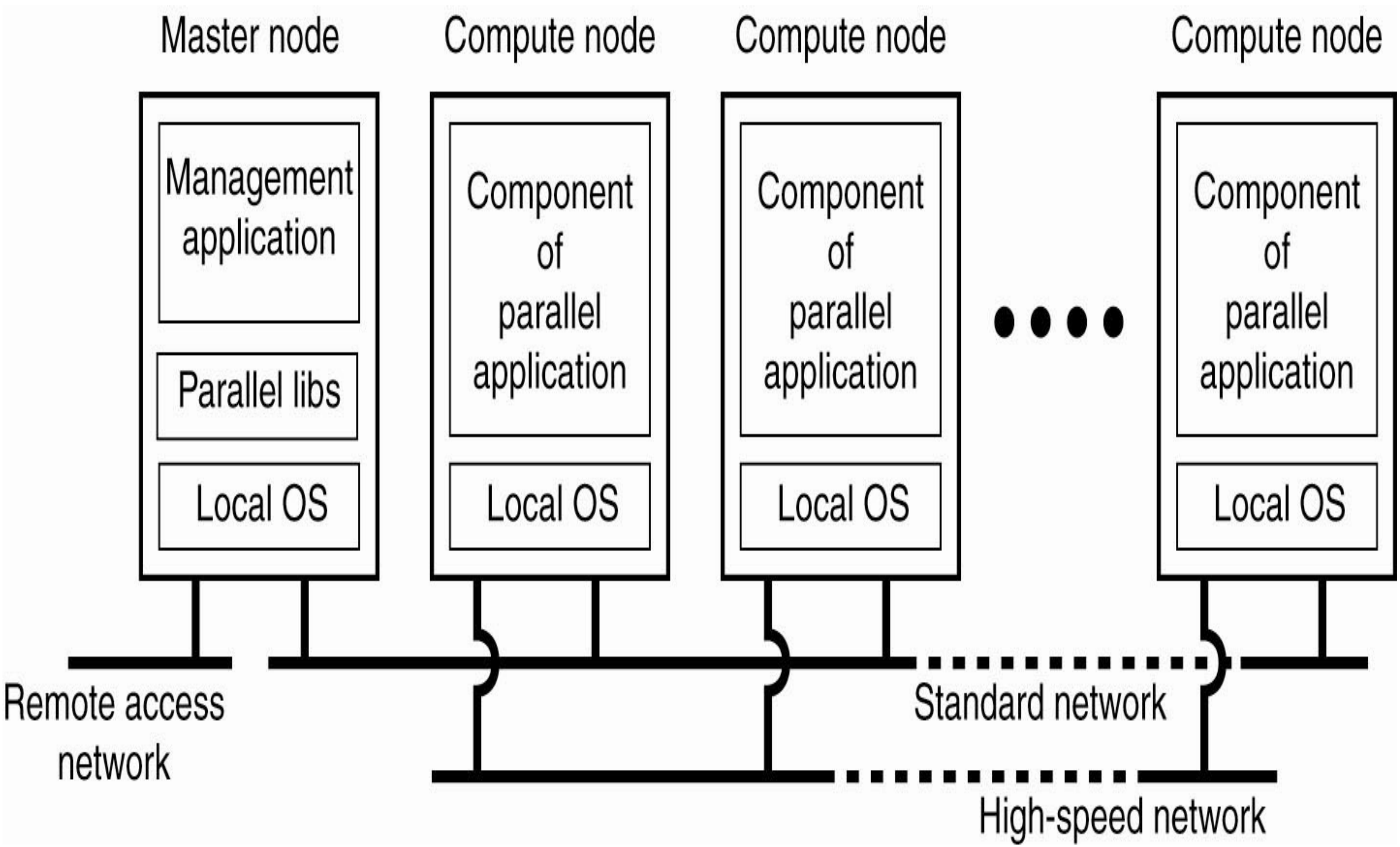
Clustered Systems Architecture



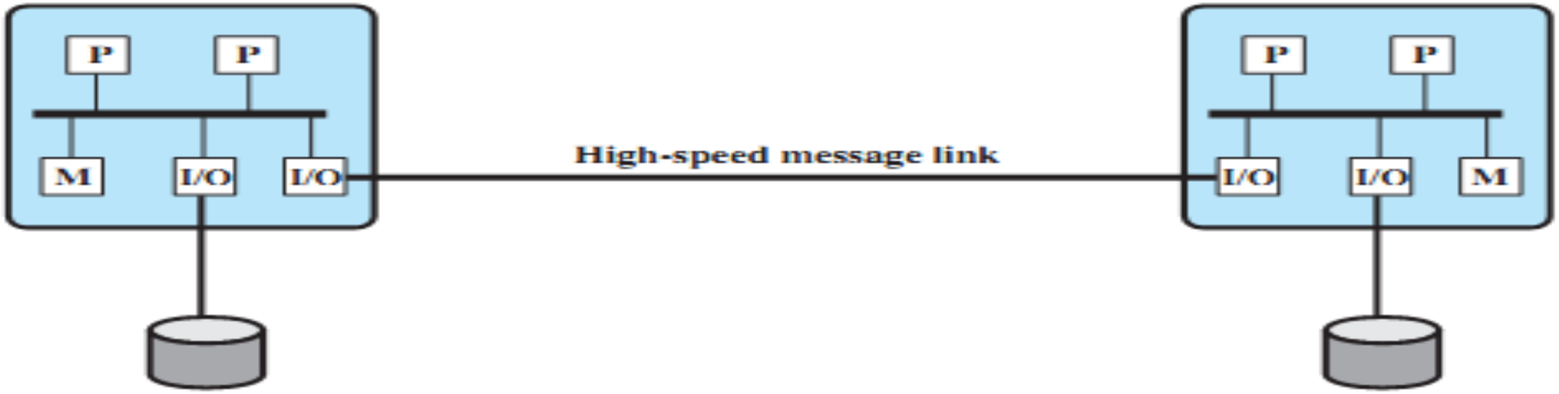
Cluster Computing Systems

- Collection of similar workstations/PCs, closely connected by means of a high-speed LAN:
  - Each node runs the same OS.
  - Homogeneous environment
  - Can serve as a supercomputer
  - Excellent for parallel programming
- Examples: Linux-based Beowulf clusters, MOSIX (from Hebrew University).

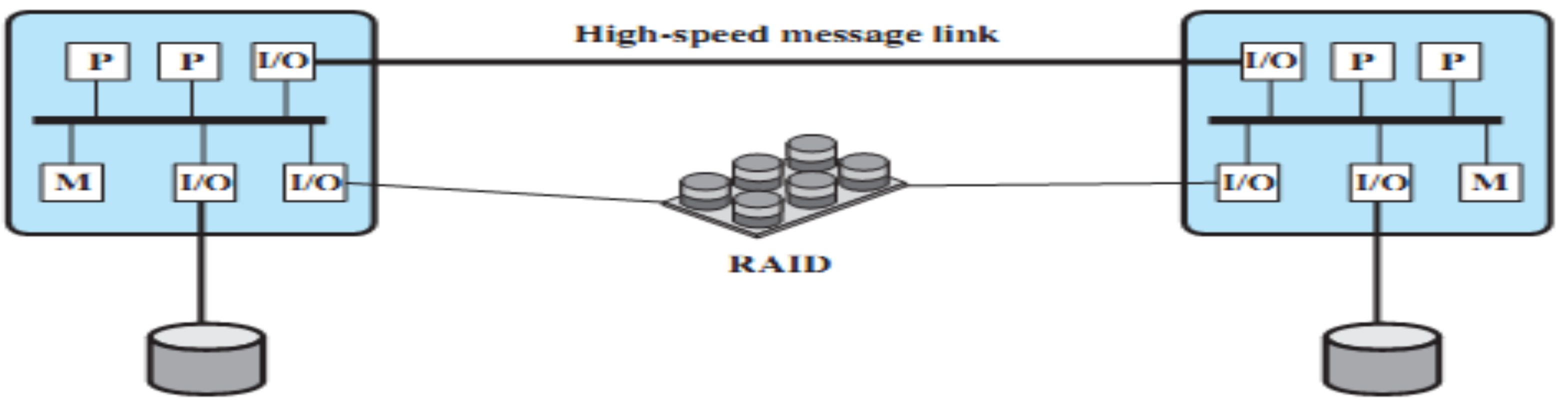
Architecture for Cluster Computing System



Cluster Configurations



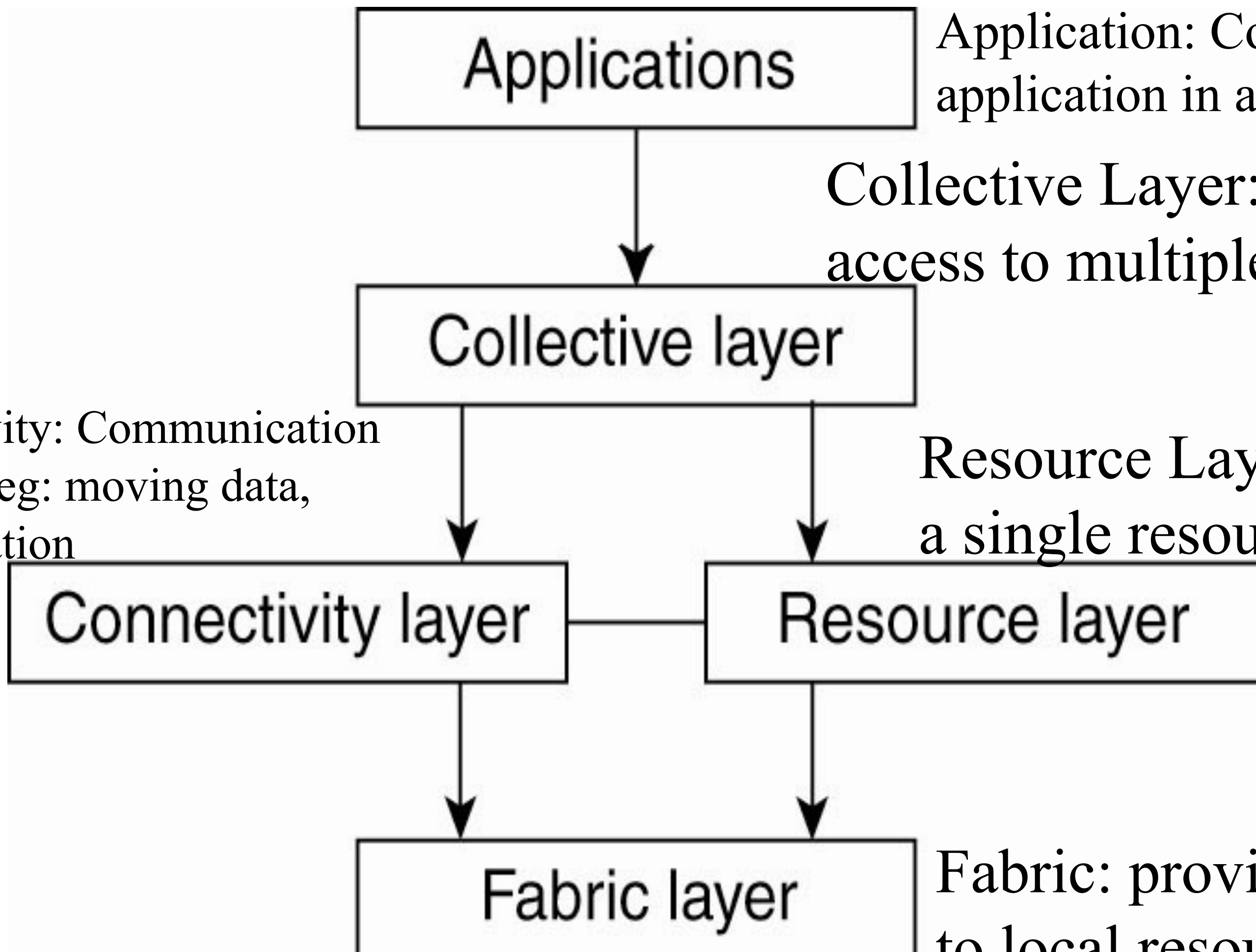
(a) Standby server with no shared disk



Grid Computing Systems

- Collection of computer resources, usually owned by multiple parties and in multiple locations, connected together such that users can share access to their combined power:
  - Can easily span a wide-area network
  - Heterogeneous environment
  - Crosses administrative/geographic boundaries
  - Supports Virtual Organizations (VOs): grouping of users that will allow for authorization on resource allocation.
- Examples: EGEE - Enabling Grids for E-SciencE (Europe), Open Science Grid (USA).

Architecture for Grid Computing Systems



Application: Contains actual grid application in a single organization

Collective Layer: Handles access to multiple resources

Connectivity: Communication protocols eg: moving data, authentication

Resource Layer: manage a single resource

Fabric: provide interfaces to local resources

- Recall OS

Computing Environments – Cloud Computing

- Delivers computing, storage, even apps as a service across a network
- Logical extension of virtualization as based on virtualization
- Many types
  - **Public cloud**
  - **Private cloud**
  - **Hybrid cloud** – includes both public and private cloud components
- Models
  - Software as a Service (**SaaS**) – one or more applications available via the Internet (i.e. word processor)
  - Platform as a Service (**PaaS**) – software stack ready for application use via the Internet (i.e a database server)
  - Infrastructure as a Service (**IaaS**) – servers or storage available over Internet (i.e. storage available for backup use)

- END Recall OS

Cloud Computing Systems (1)

- Collection of computer resources, usually owned by a single entity, connected together such that users can lease access to a share of their combined power:
  - Location independence: the user can access the desired service from anywhere in the world, using any device with any (supported) system.
  - Cost-effectiveness: the whole infrastructure is owned by the provider and requires no capital outlay by the user.
  - Reliability: enhanced by way of multiple redundant sites, though outages can occur, leaving users unable to remedy the situation.

Cloud Computing Systems (2)

- Scalability: user needs can be tailored to available resources as demand dictates – cost benefit is obvious.
- Security: low risk of data loss thanks to centralization, though problems with control over sensitive data need to be solved.
- Readily consumable: the user usually does not need to do much deployment or customization, as the provided services are easy to adopt and ready-to-use.

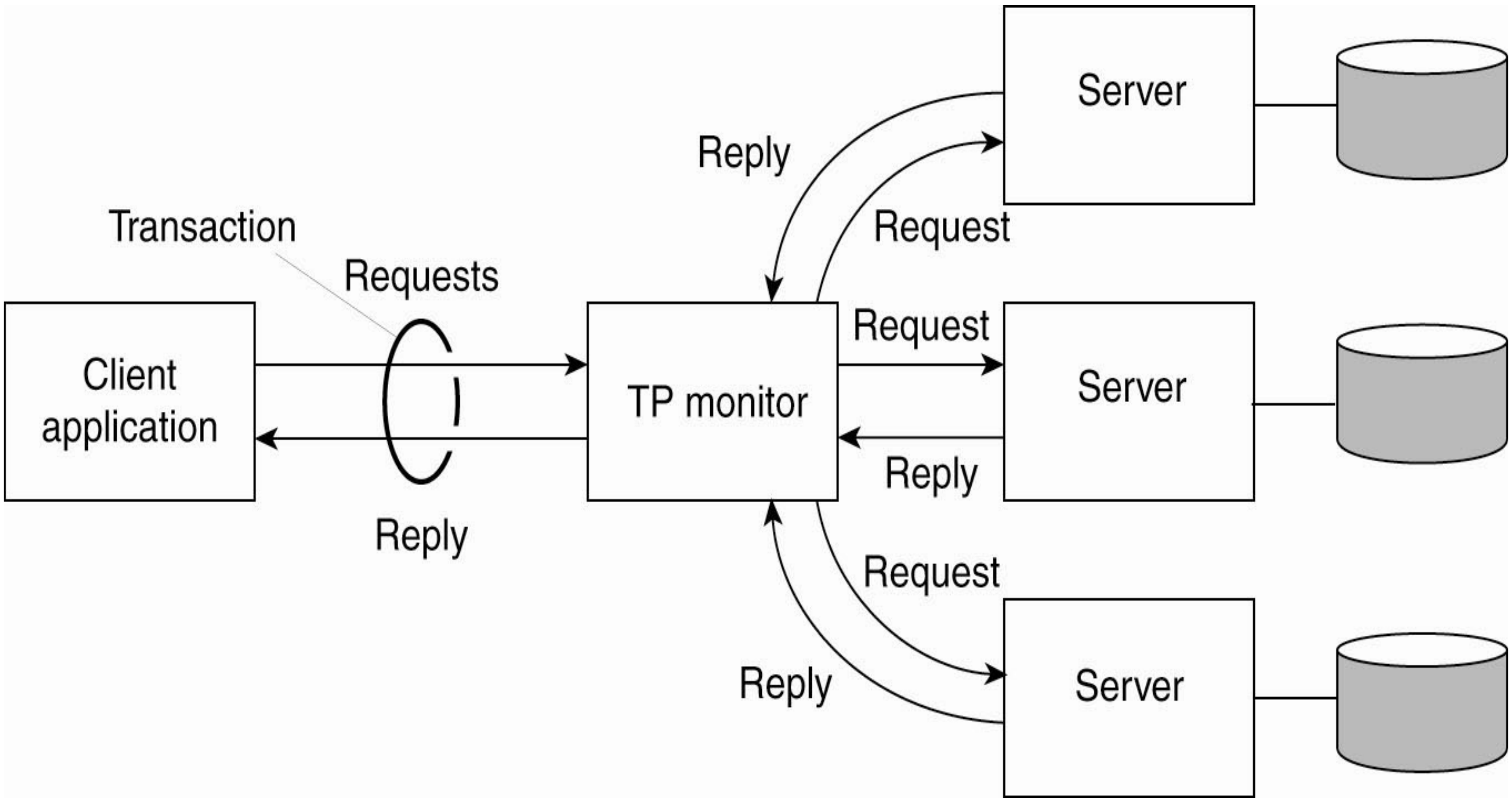
• Examples: Amazon EC2 (Elastic Compute Cloud), Google App Engine, IBM Enterprise Data Center, MS Windows Azure, SUN Cloud Computing.

Integrating Applications

- Uniting the databases and workflows associated with business applications to ensure the business uses them consistently.

- Example: data from CRM can be integrated with e-mail marketing platform.

Transaction Processing Systems (TPS)

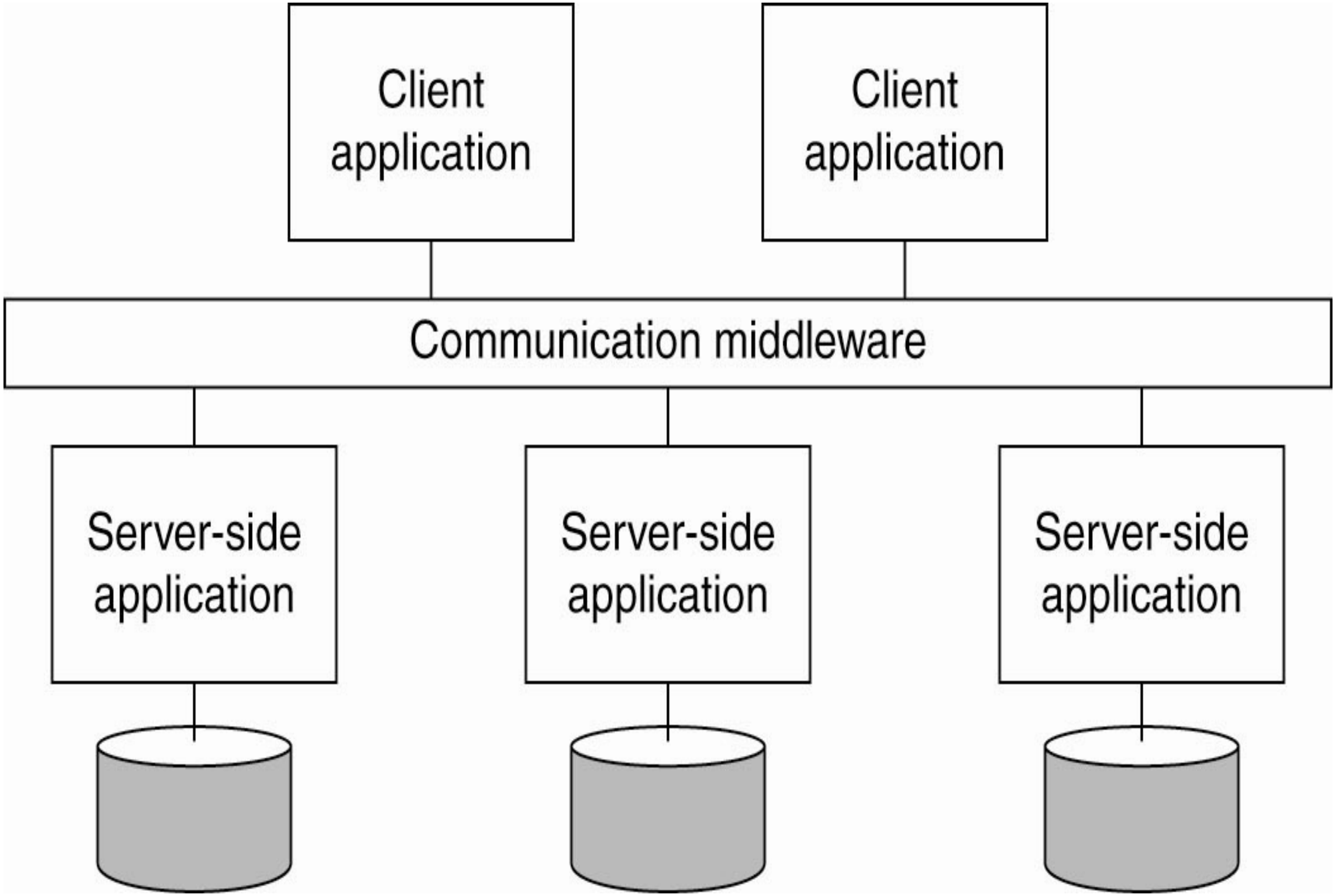


The role of a TP monitor in distributed systems

- TP Monitor: Transaction Processing Monitor
  - Allows an application to access different server
  - Coordinates the commit of the transactions

→ So, it provides services that are useful for many applications avoiding that such service be implemented by the applications themselves.

Enterprise Application Integration



Communication Middleware Models/Paradigm

- Distributed File Systems
- Remote Procedure Call (RPC)
- Distributed Objects (RMI)
- Distributed Documents

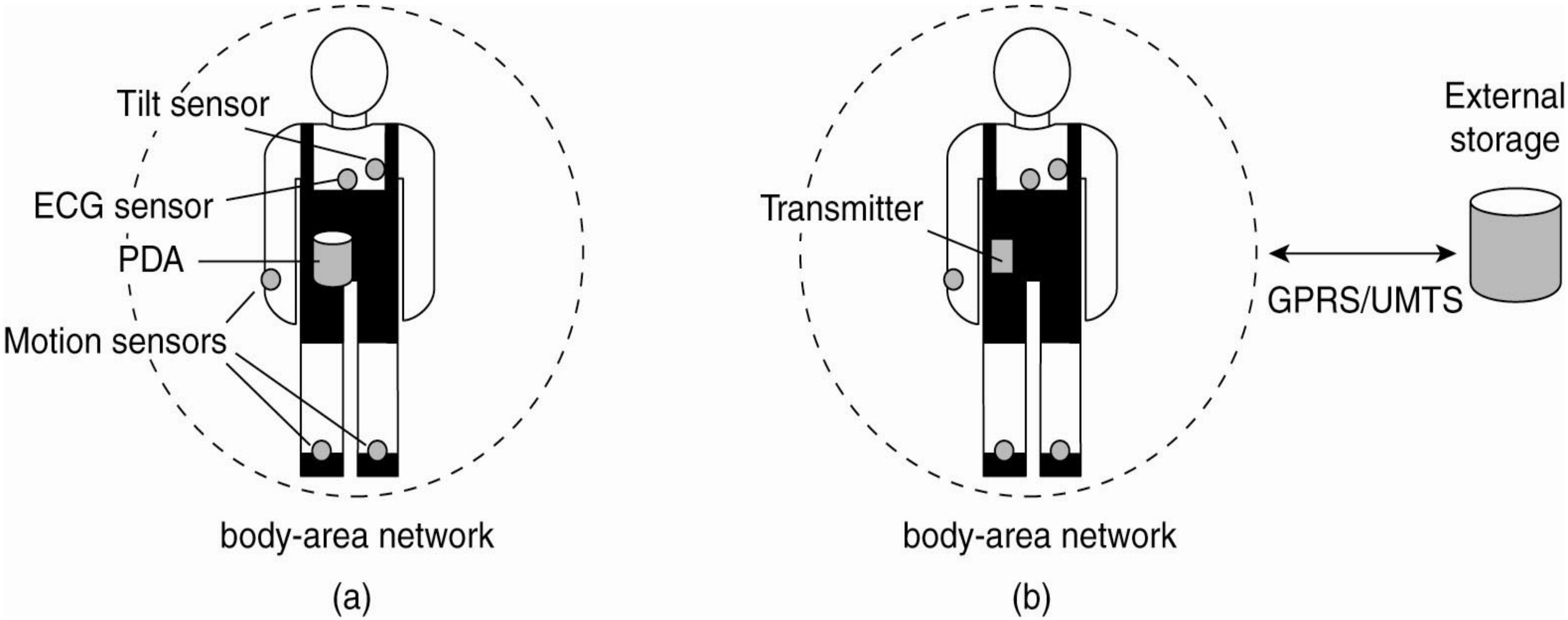
Distributed Pervasive Systems

- Requirements for pervasive systems:
  - Embrace contextual changes
  - Encourage ad hoc composition
  - Recognize sharing as the default
  - Support distribution transparency

Electronic Health Care Systems (1)

- 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?

Electronic Health Care Systems (2)

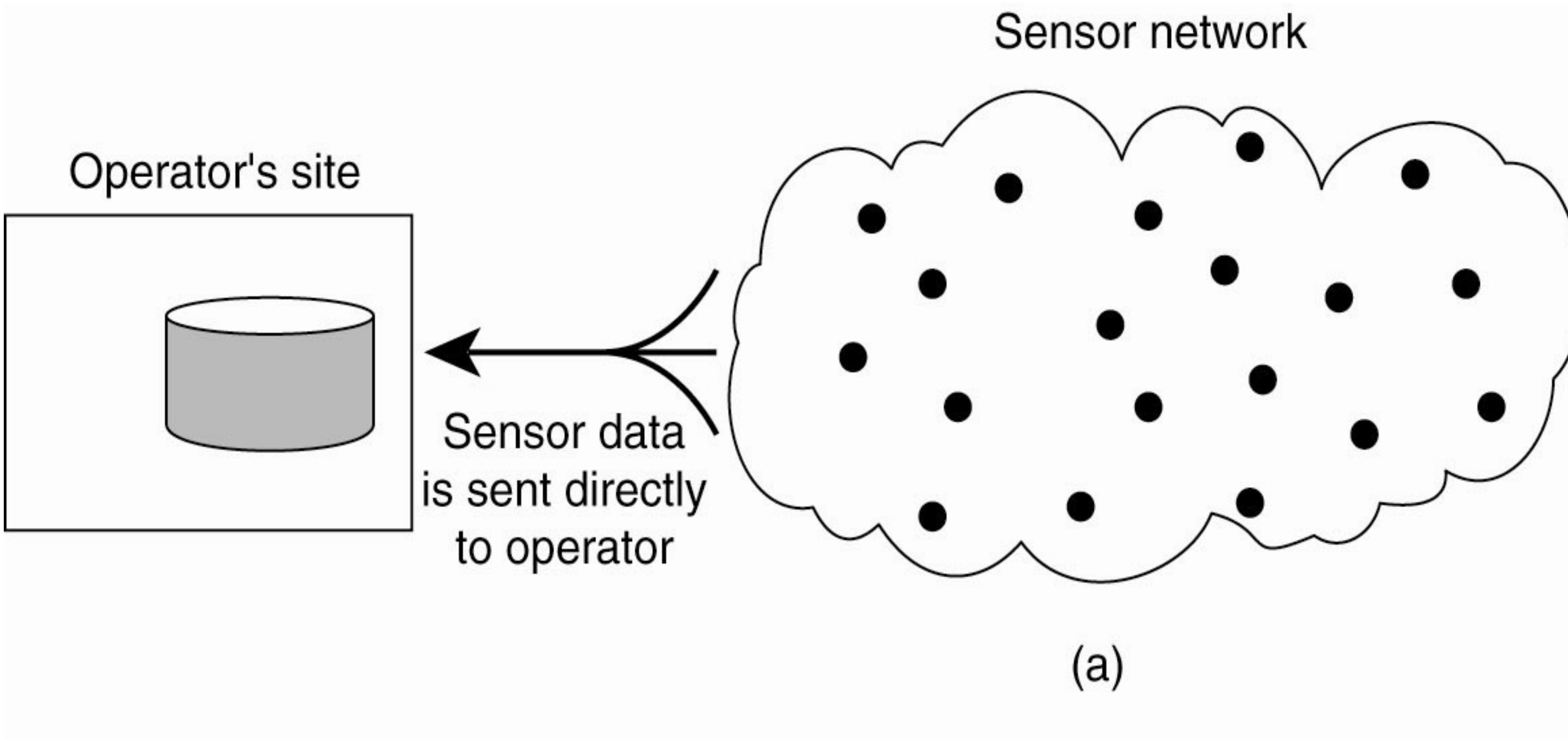


Monitoring a person in a pervasive electronic health care system, using (a) a local hub or (b) a continuous wireless connection

Sensor Networks (1)

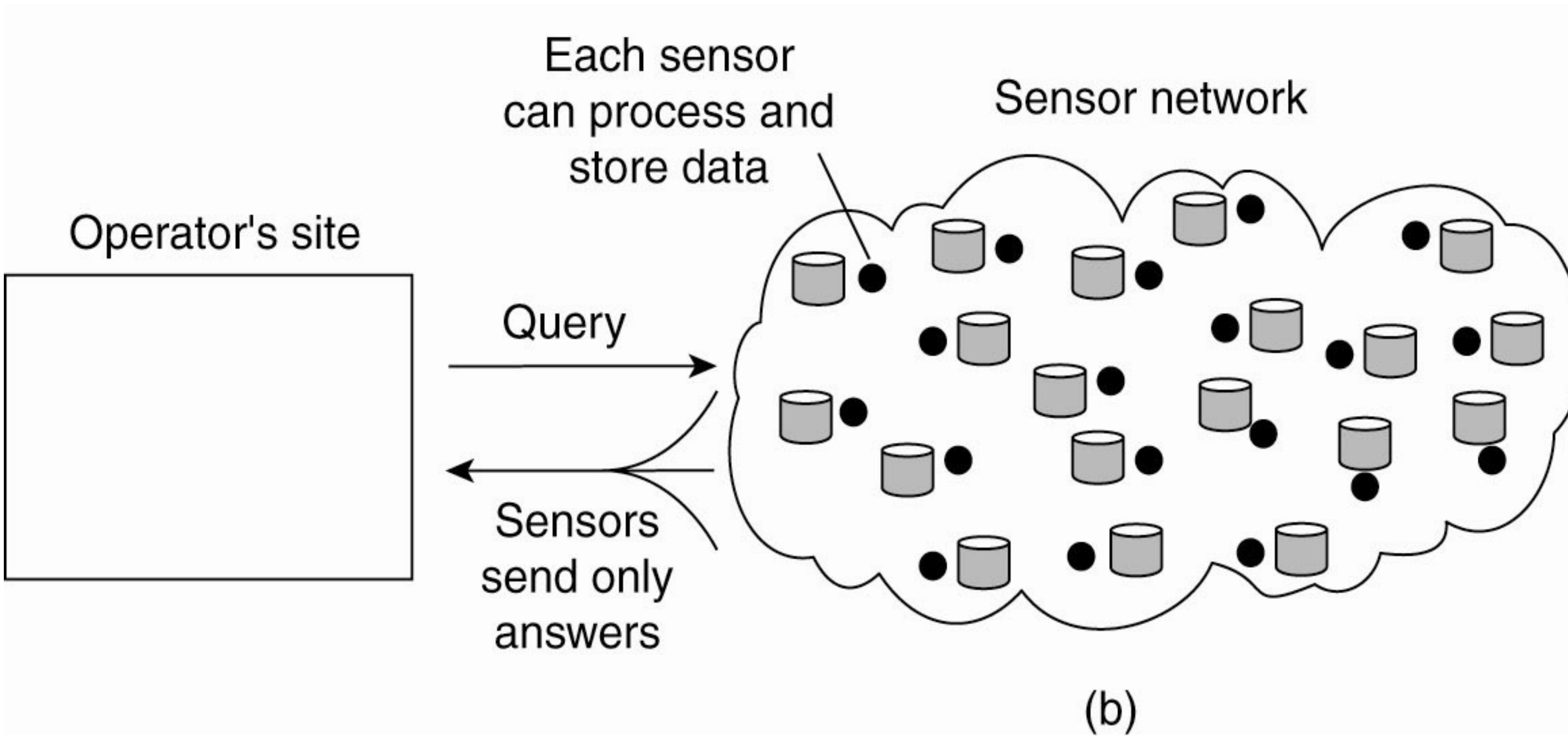
- The nodes to which sensors are attached are:
  - Many (10s-1000s).
  - Simple (i.e., hardly any memory, CPU power, or communication facilities).
  - Often battery-powered (or even battery-less).
- Questions concerning sensor networks:
  - How do we (dynamically) set up an efficient tree in a sensor network?
  - How does aggregation of results take place? Can it be controlled?
  - What happens when network links fail?

Sensor Networks (2)



Organizing a sensor network database, while storing and processing data (a) only at the operator's site or …

Sensor Networks (3)



Organizing a sensor network database, while storing and processing data … or (b) only at the sensors