Introduction to distributed computing

Definition of a distributed system

A distributed system is:

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

Important aspects of the definition are

1. A distributed system consists of components (computers) that are autonomous.
2. Users think they are dealing with a single system.

1.1 Definition of a Distributed System (2)

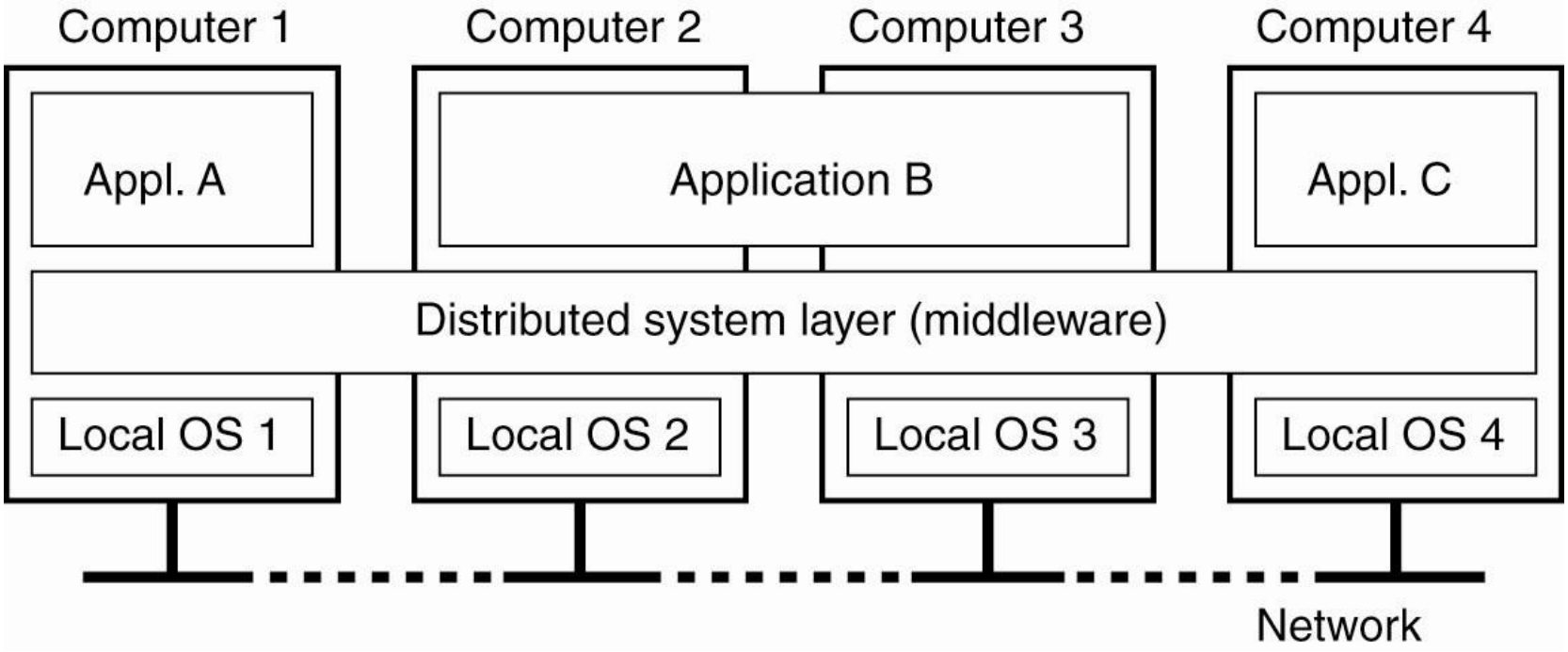


Figure 1-1. A distributed system organized as middleware. The middleware layer extends over multiple machines, and offers each application the same interface.

Characteristics of distributed system

1. One of the important characteristic is that differences between the various computers and the ways in which they communicate are mostly hidden from users.

   The same holds for the internal organization of the distributed system.

2. Users and applications can interact with a distributed system in a consistent and uniform way, regardless of where and when interaction takes place.

Goals

- A distributed system should
  - Easily accessible
  - Hide the fact that resources are distributed across a network- **distribution transparency**
  - Open
  - Scalable

1. Making resources accessible

- The main goal of distributed system is to make it easy for the users to access remote resources, and to share them in a controlled and efficient way.
- Connecting users and resources also makes it easier to collaborate and exchange information.
- The connectivity of the Internet is now leading to numerous virtual organizations in which geographically widely- dispersed groups of people work together by means of groupware, i.e., s/w for collaborative editing, teleconferencing, and so on.

- As connectivity and sharing increases security becoming important.
- System should provide protection against intrusion on communication(passwords, credit card number protection)
- Can lead to unwanted communication (spam)

2. Distribution Transparency

- The main goal of distributed system is to hide the fact that its processes and resources are physically distributed across multiple computers.
- A distributed system that is able to present itself to users and applications as if it were only a single computer system is said to be transparent.

Types of Transparency

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

Figure 1-2. Different forms of transparency in a distributed system (ISO, 1995).

- **Access transparency**
  - It deals with hiding differences in data representation and the way that resources can be accessed by users.
  - At a basic level, hide differences in machine architectures.
  - For eg, a distributed system may have computer systems that run different OS, each having their own naming conventions, as well as how files can be manipulated, should all be hidden from users and applications.

- **Location transparency**
  - It deals with the fact that users cannot tell where a resource is physically located in the system.
  - Naming plays an important role in achieving location transparency.
  - location transparency can be achieved by assigning only logical names to resources , ie, names in which the location of a resource is not secretly encoded.
  - An eg for such a name is the URL http://www.prenhall.com/index.html, which give s no clue about the location of prentice hall's main web server.

- The URL also gives no clue as to whether index.html has always been at its current location or was recently moved there.

- **Migration transparency.**
  - Distributed systems in which resources can be moved without affecting how those resources can be accessed are said to provide migration transparency
  - Even stronger is the situation in which resources can be relocated while they are being accessed without the user or application noticing anything. In such cases, the system is said to support **relocation transparency.**
  - An eg for relocation transparency is when mobile users can continue to use their wireless laptops while moving from place to place without ever being (temporarily) disconnected.

- **Replication transparency.**
  - It deals with hiding the fact that several copies of a resource exist.
  - To hide replication from users, it is necessary that all replicas have the same name.
  - Consequently, a system that supports replication transparency should generally support location transparency as well, because it would otherwise be impossible to refer to replicas at different locations.

- Two independent users may each have stored their files on the same file server or may be accessing the same tables in a shared database.
- In such cases, it is important that each user does not notice that other is making use of the same resource. This phenomenon is called **concurrency transparency.**
- It is imp. Issue is that concurrent access to a shared resource leaves that resource in a consistent state.
- Consistency can be achieved through locking mechanisms, by which users are, in turn given exclusive access to the desired resource.

- **Failure transparency.**
  - Making a distributed system failure transparent means that a user does not notice that a resource(he has possibly never heard of) fails to work properly, and that the system subsequently recovers from that failure.
  - Masking failures is the hardest issue
  - Inability to distinguish between a dead resource and slow resource

Degree of Transparency

- A trade-off between a high degree of transparency and the performance of a system
- Full distribution transparency is simply impossible

3. Openness

- Degree to which new resource sharing services can be added and made available to the users.
- An open distributed system is a system that offers services according to standard rules that describe the syntax and semantics of those services.
- A system offer services according to standard rules and such rules are formalized in protocols.
- In distributed systems, services are generally specified through interfaces**,** which are often described in **an interface definition language(IDL).(***eg-facebook trift , protocol buffers(googl's IDL***)**

Openness

- Interoperability
  - Different manufacturers can co-exist and work together
- Portability
  - System A can be executed, without modification, on a different distributed system B
- Extensible
  - in an extensible system, it should be relatively easy to add parts that run on a different OS, or even to replace an entire file system.

- Separating policy from mechanism
  - To achieve flexibility in open distributed systems, it is crucial that the system is organized as a collection of relatively small and easily replaceable or adaptable components.

Scalability

- One of the most important design goals for developers of distributed systems

Can be measured along three different dimensions

1) With respect its size
   - Means we can easily add more users and resources to the system.

2) Geographically scalable system
   - In which users and resources may lie far apart.

3) administratively scalable
   - Means it can still be easy to manage even if it spans many independent administrative organizations.

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

Figure 1-3. 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.

- Geographical scalability has its own problems.
  - one of the main reasons why it is hard to scale existing distributed systems that were designed for LAN is that they are based on **synchronous communication.**
  - In synchronous communication , a party requesting service , generally referred to as a client, blocks until a reply is sent back.

- Another problem is that communication in WAN is unreliable, and virtually always point-to-point.

Scaling Techniques

- Three techniques for scaling
  - Hiding communication latencies
  - Distribution
  - Replication

Hiding communication latencies

- It is imp. to achieve geographical scalability.
- Here the idea is try to avoid waiting for responses to remote service requests as much as possible.
- For e.g., when a server has been requested at a remote machine, an alternative to waiting for a reply from a server is to do another useful work at the requester's side.
- I.e. It uses only **asynchronous communication.**

- When a reply comes in , the application is interrupted and a special handler is called to complete the previously- issued request.
- Asynchronous communication is often used in batch- processing systems and parallel applications.

Asynchronous communication

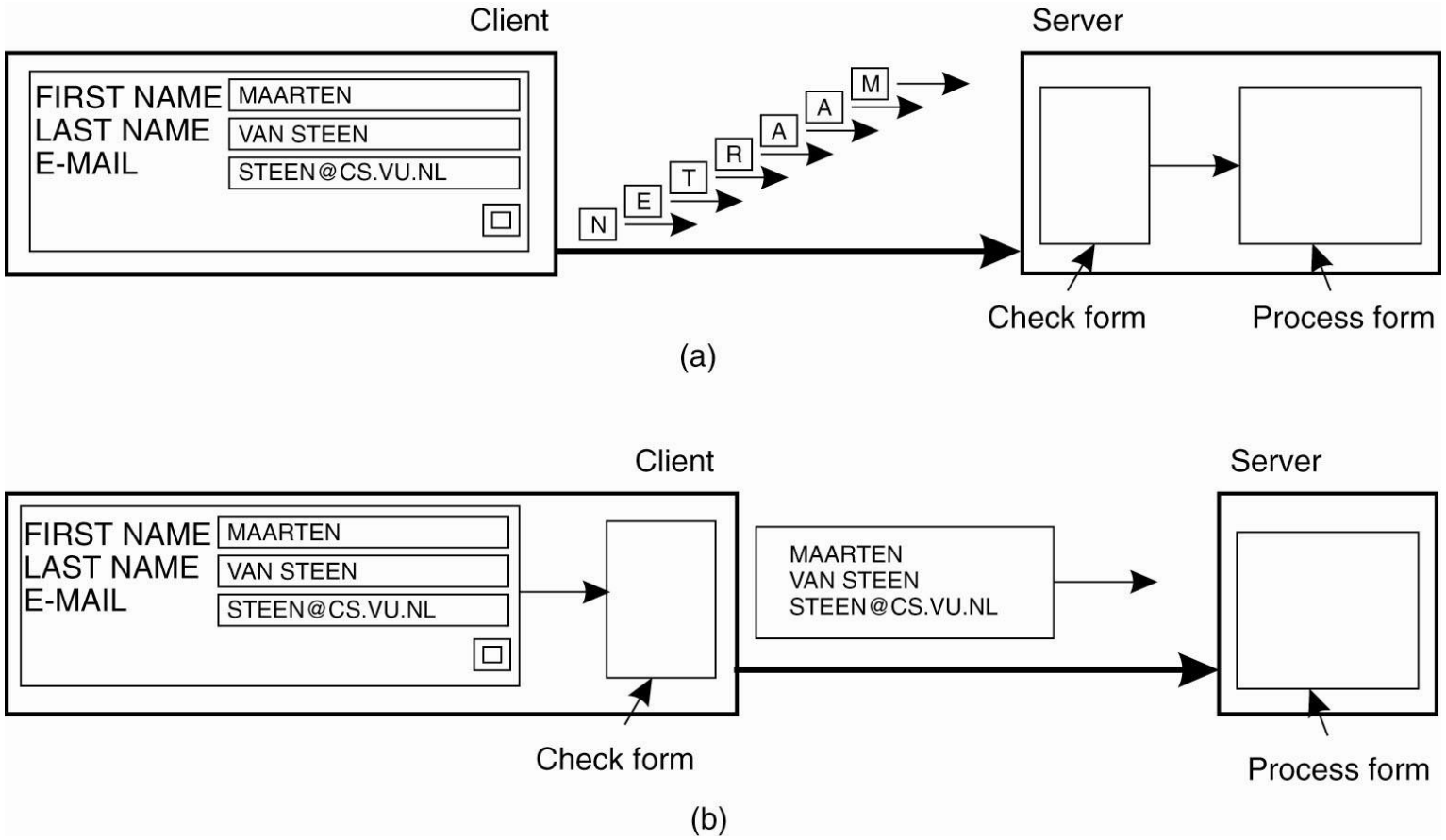


Figure 1-4. The difference between letting (a) a server or (b) a client check forms as they are being filled.

Distribution

- It involves taking a component, splitting it into smaller parts, and subsequently spreading those parts across the system.
- E.g. is DNS.
- The DNS name space is hierarchically organized into a tree of domains , which are divided into non overlapping zones.
- The names in each zone are handled by a single name server

Distribution

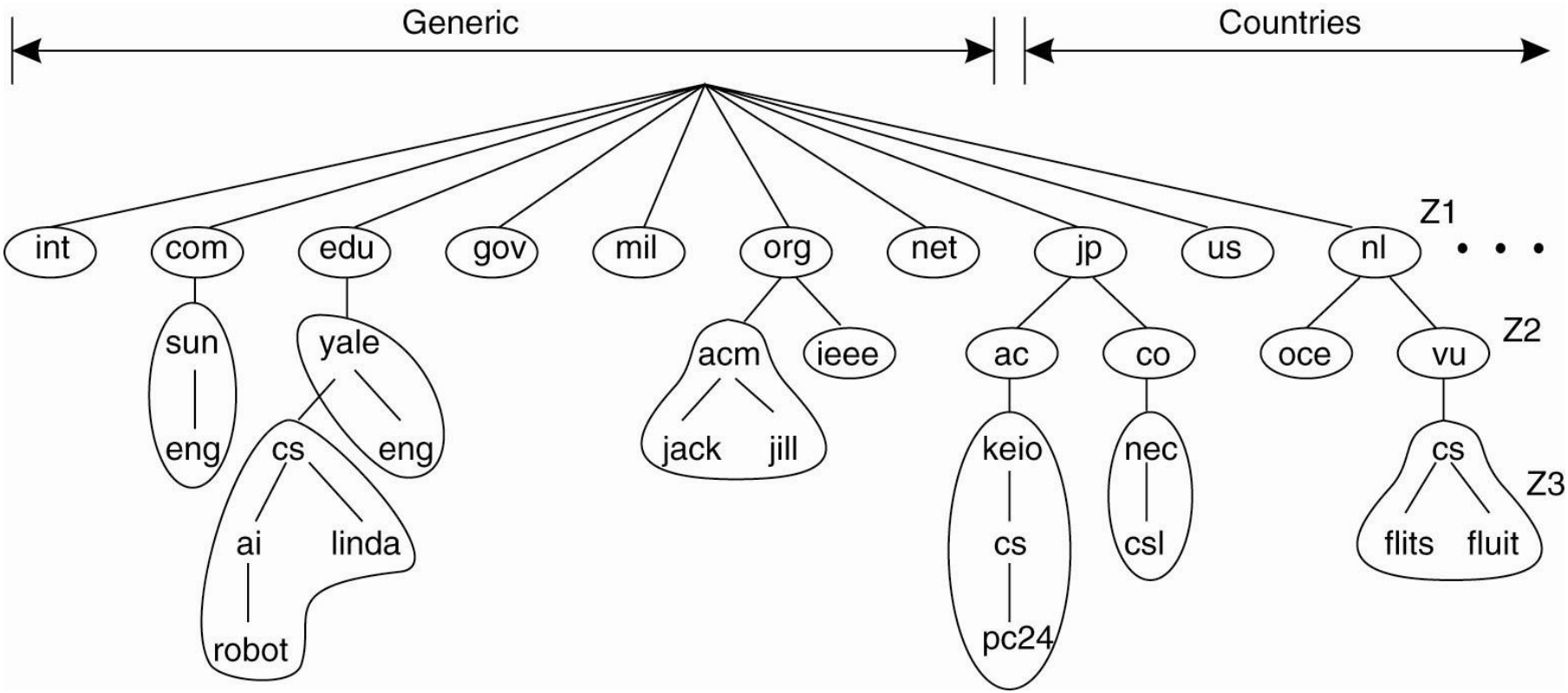


Figure 1-5. An example of dividing the DNS name space into zones.

- Scalability problems often appear in the form of performance degradation, it is generally a good idea to actually replicate components across a distributed system.
- Replication not only increases availability, but also helps to balance the load between components leading to better performance.

- Caching is a special form of replication
- Drawback of caching and replication is it leads to consistency problems.(modifying one copy makes that copy different from the others.)

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.

Distributed System Challenges

Lack of global state information

- Different nodes have different view of system
  - What are the contents of file A?
  - How many jobs are running on node X?
  - Which nodes are currently part of the system?
- See delays, different ordering of messages, lost messages, network partitions
- Tension with goal of "single coherent system"

Handling slow, failed and misbehaving nodes

- How do you avoid slow nodes?
- How do you get back data or work from failed node?
- When nodes disagree, how do you know who is wrong?
- Tension with goal of "available and reliable"

When is it okay to have some centralized components?

- Simplifies state management, but single point-of-failure and performance bottleneck

Benefits of Distributed Systems

Great price/performance

- Leverage commodity components (nodes and networks)
- Use many, many of them

Incremental scalability

- Can add x% new nodes (or disks or memory) to improve performance x%

Improved availability

- Continue operating when some nodes stop working

Improved reliability

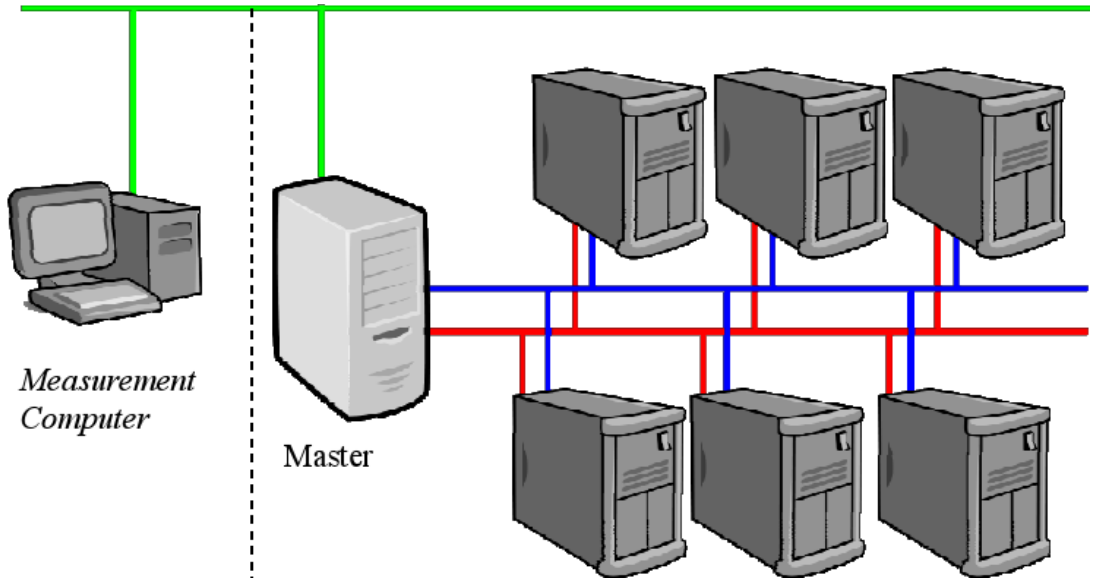
- Deliver correct results when some nodes misbehave, corrupt data

Allow geographically-distributed individuals to share data or cooperate

1.3 Types of Distributed Systems

1) Distributed computing systems
   - Cluster computing
   - Grid computing
2) Distributed information systems
   - Transaction processing systems(TPS)
   - Enterprise application integration(EAI)
3) Distributed pervasive systems
   - Home systems
   - Electronic health systems
   - Sensor networks

Cluster Computing

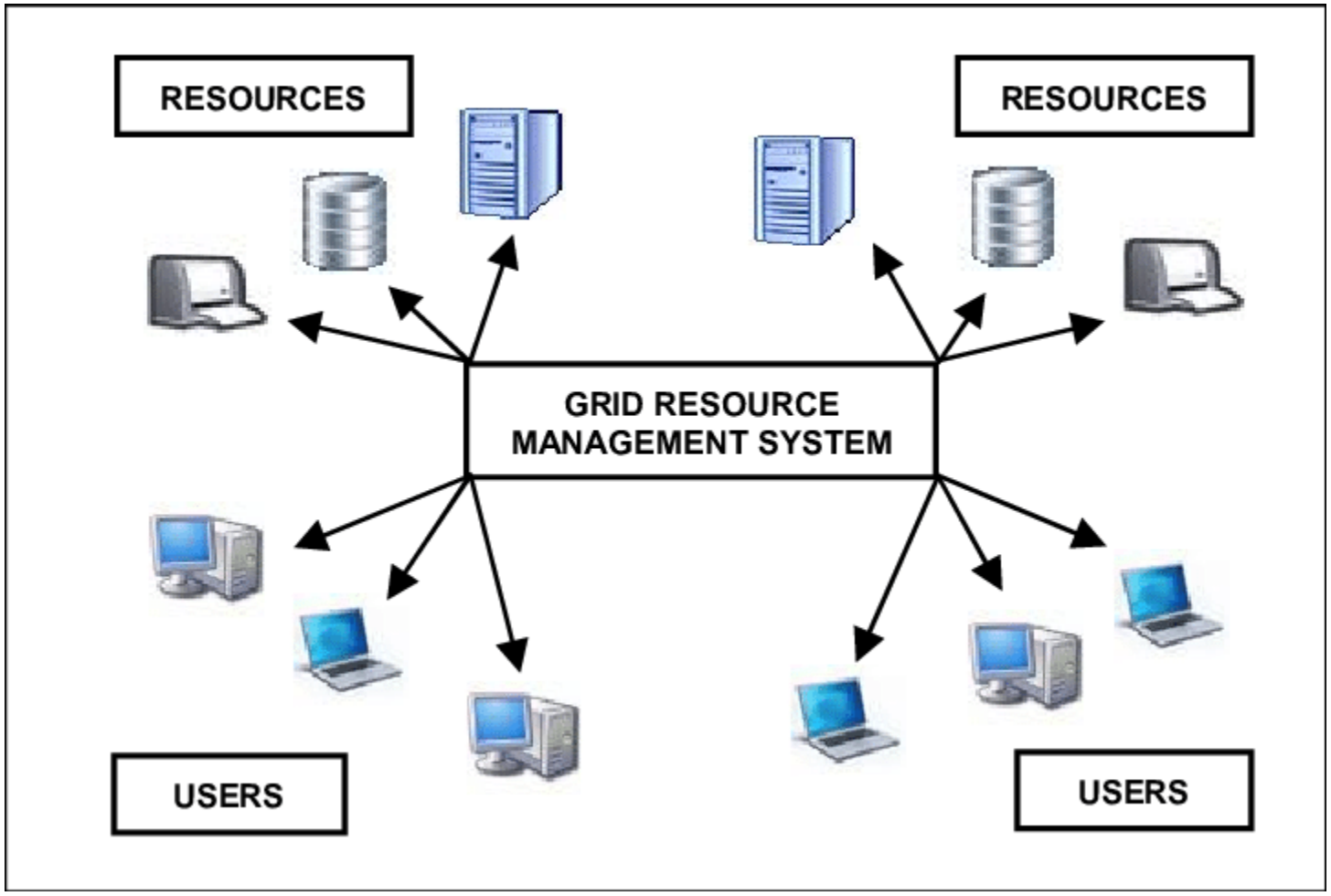


- Technique of linking two or more computers into a network (usually through LAN) in order to take advantage of the parallel processing power of those computers.

Distributed Computing systems

- An important class of distributed systems is the one used for high- performance computing tasks.
- In **cluster computing** the underlying h/w consists of a collection of similar workstations or PCs, closely connected by means of a high speed LAN.
- Here each node runs the same OS.

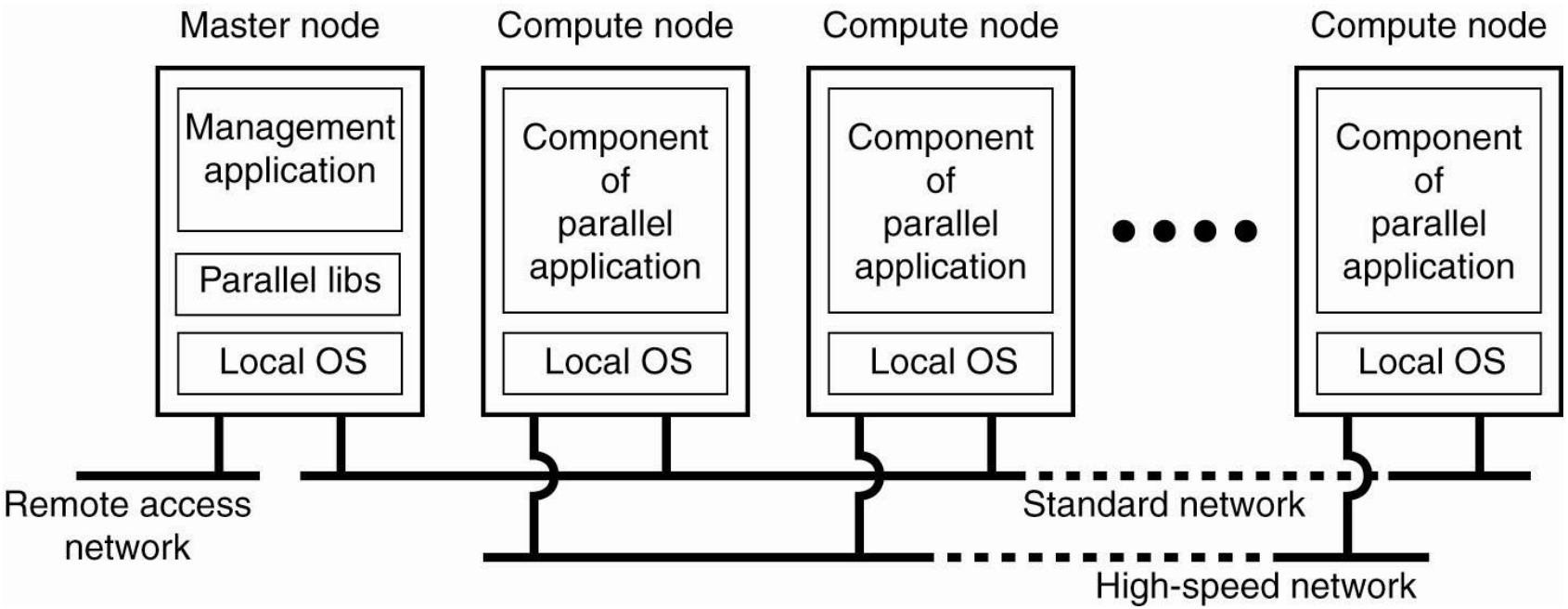
- In **grid computing**, the subgroup consists of distributed systems that are often constructed as a federation of computer systems , where each system may fall under a different administrative domain, and may be very different when it comes to h/w, s/w and n/w technology.



Cluster computing systems

- Cluster computing is used for parallel programming in which a single program is run in parallel on multiple machines.
- In the following figure, each cluster consists of a collection of compute nodes that are controlled and accessed by means of a single master node.

Cluster Computing Systems



Grid computing systems

- A characteristic feature of cluster computing is its homogeneity.
- A key issue in grid computing system is that resources from different organizations are brought together to allow the collaboration of a group of people or institutions. Such a collaboration is realized in the form of a **virtual organization.**

Grid Computing Systems

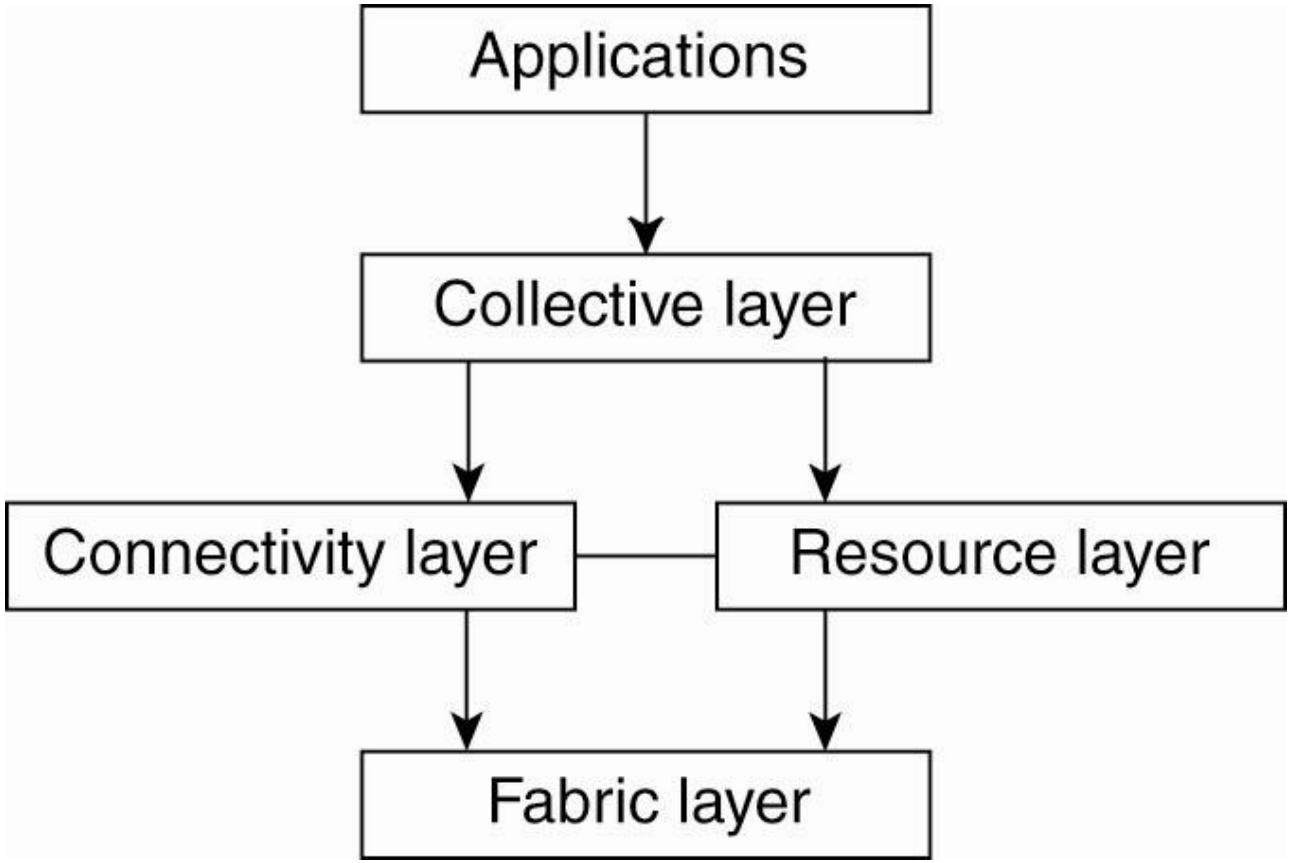


Figure 1-7. A layered architecture for grid computing systems.

Grid Computing System Architecture

- Fabric layer
  - Lowest layer
  - Provide interface to local resources at a specific site
- Connectivity layer
  - Communication protocols for supporting grid transactions that span the usage of multiple resources.

Grid Computing Systems

- Resource layer
  - Manage a single resource.
  - It uses the functions provided by the connectivity layer and calls directly the interfaces made available by the fabric layer.
  - Responsible for access control, and hence rely on the authentication performed as part of the connectivity layer.

- Collective layer
  - Handle access to multiple resources
  - Consist of services for resource discovery, allocation, and scheduling of tasks onto multiple resources, data replication and so on.
  - Unlike the connectivity and resource layer, which consist of a relatively small, standard collection of protocols, the collective layer may consist of many different protocols for many different purposes.

- Application layer
  - Consist of applications and operate within a virtual organization and which make use of the grid computing environment.
  - The collective, connectivity and resource layer form the heart of the grid middleware layer..
  - These layers jointly provide access to and management of resources that are potentially dispersed across multiple sites.

Distributed Computing Systems – Advantages

- Inherently distributed applications
  - Distributed DB, worldwide airline reservation, banking system
- Information sharing among distributed users
  - groupware
- Resource sharing
  - Sharing DB/expensive hardware and controlling remote lab. devices
- Better cost-performance ratio / Performance
  - Effective for coarse-grained or embarrassingly parallel applications
- Reliability
  - Non-stopping (availability).
- Scalability
  - Loosely coupled connection and hot plug-in
- Flexibility
  - Reconfigure the system to meet users' requirements

Challenges (Problems / Disadvantages)

- Application distribution
- Heterogeneity
- Security
- Scalability
- Failure handling/reliability
- Concurrency
- Transparency
- Management

2) Distributed information systems

- Integration at the lowest level would allow clients to wrap a number of requests, possible for different servers, into a single larger request and have it executed as a distributed transaction.

Transaction Processing Systems

RPC, procedure calls to remote servers, are often encapsulated in a transaction, leading to what is known as a **transactional RPC.**

Figure 1-8. Example primitives for transactions.

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

- BEGIN – TRANACTION AND END-TRANSACTION are used to delimit the scope of a transaction.
- The operations between them form the body of the transaction.
- The characteristic feature of a transaction is either all of these operations are executed or none are executed.

Transaction Processing Systems – ACID

Characteristic properties of transactions:

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

Transaction Processing Systems

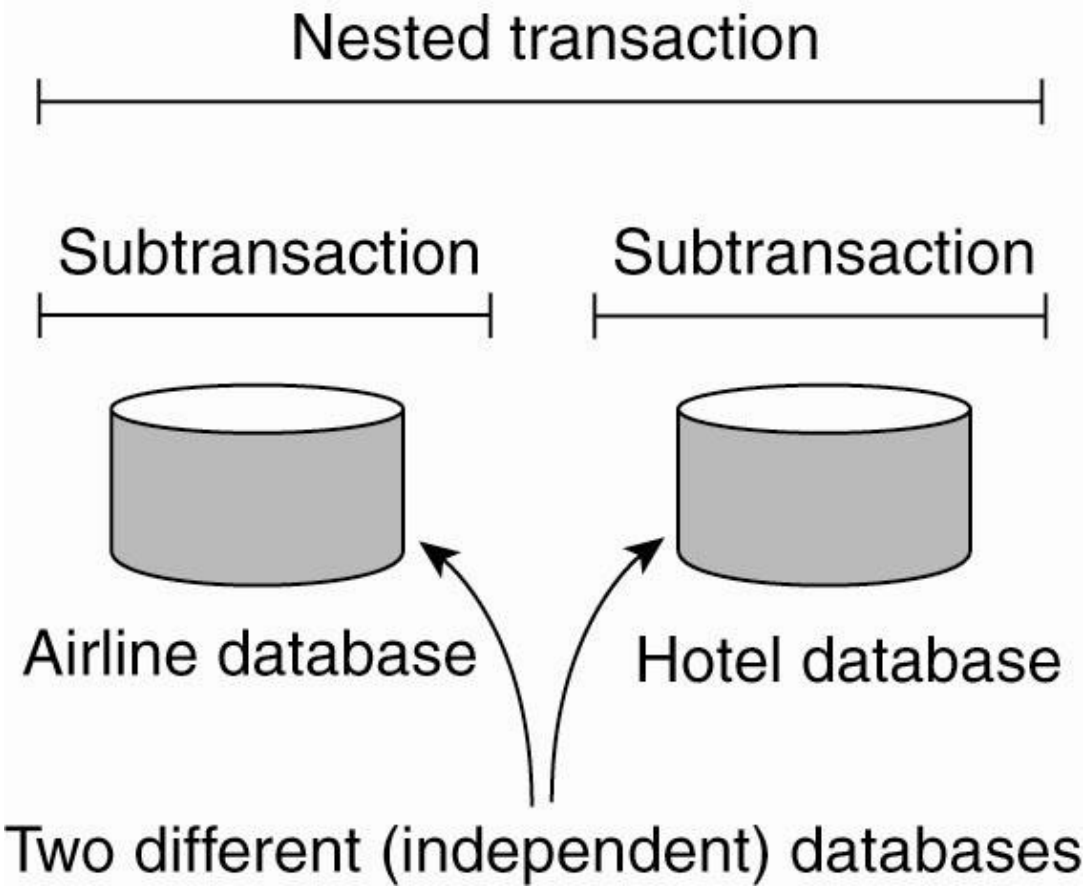


Figure 1-9. A nested transaction.

Transaction processing monitor or TP monitor

- Its main task was to allow an application to access multiple server/databases by offering it a transactional programming model as shown in the following figure.

Transaction Processing Systems

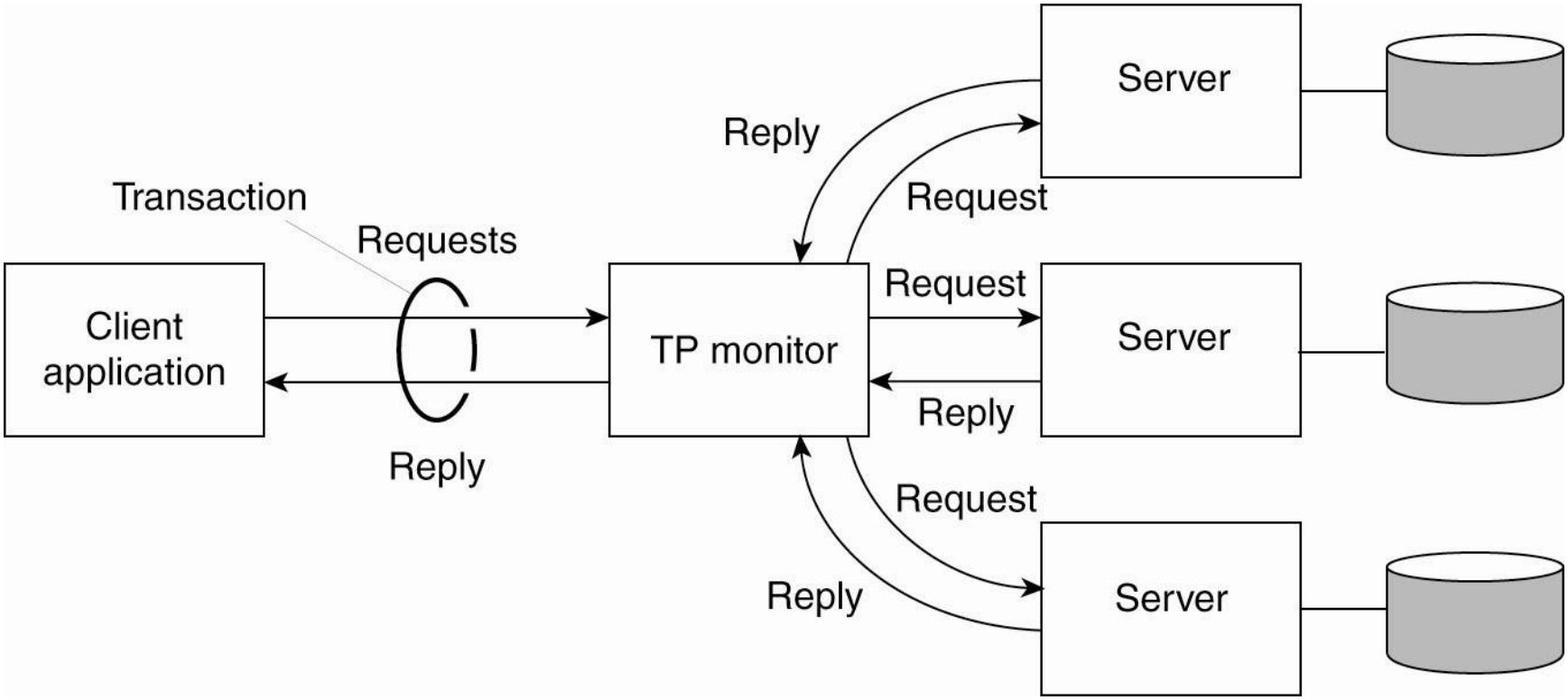


Fig: the role of a TP monitor in distributed systems

Enterprise Application Integration(EAI)

- Remote procedure calls (RPC)
  - Send a request to another application component by doing a local procedure call
- Remote method invocations (RMI)
  - It operates on objects instead of applications
- Disadvantage
  - Caller and callee need to be up and running at the time of communication
  - Tight coupling

- Message oriented middleware(MOM)
  - Here applications simply send messages to logical contact points, often described by means of a subject.
  - Publish/subscribe systems form an important and expanding class of distributed systems.

Enterprise Application Integration

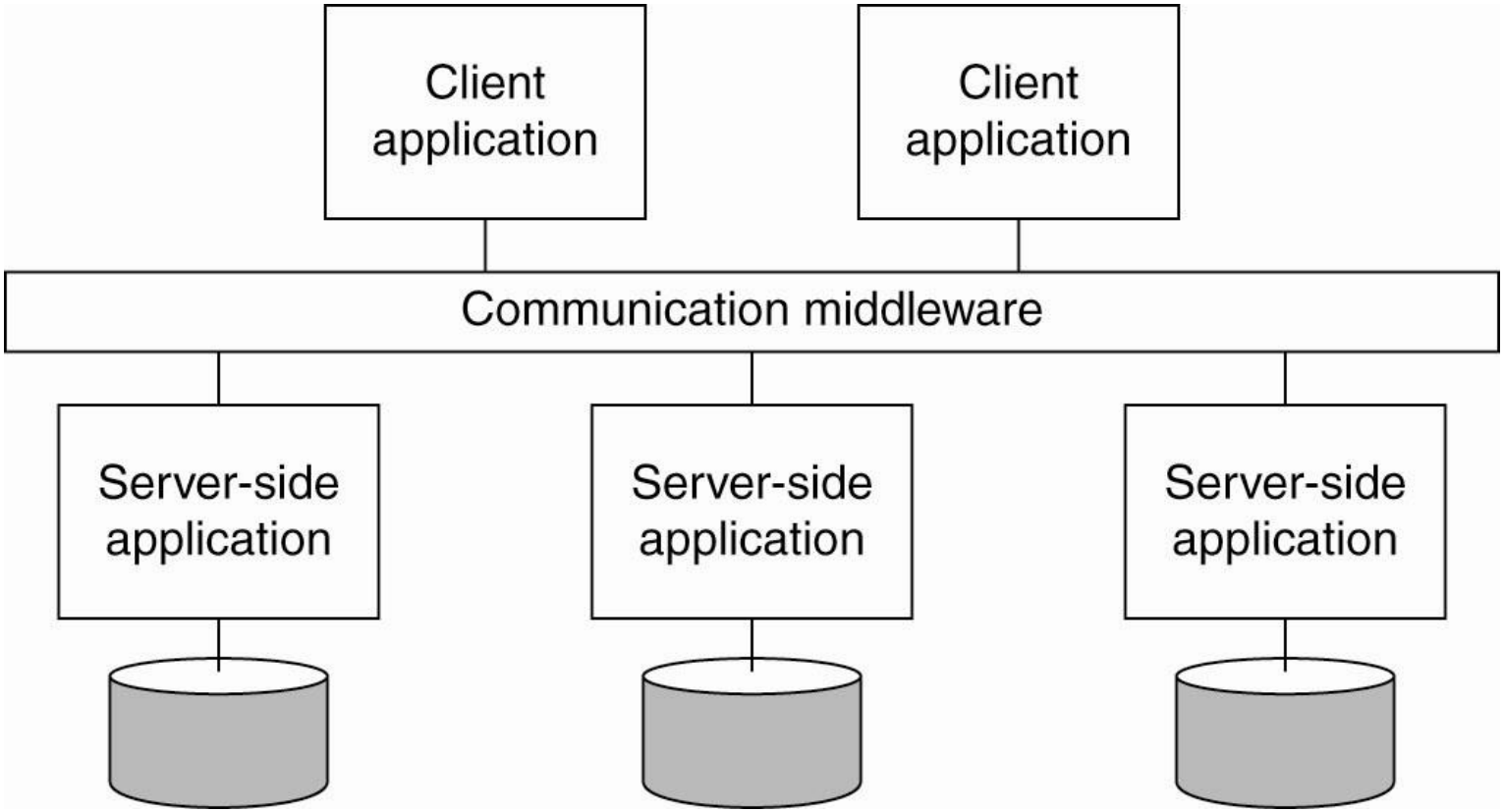


Figure 1-11. Middleware as a communication facilitator in enterprise application integration.

Distributed Pervasive Systems

Requirements for pervasive systems

- Embrace contextual changes.
  - A device must be aware of the fact that its environment may change all the time
- Encourage ad hoc composition.
  - Different ways by different users
- Recognize sharing as the default.
  - Access (and possibly provide) information

Home Systems

- Typical consumer electronics such as
  - TVs, audio and video equipment, gaming devices, (smart) phones, PDAs, and other personal wearable's into a single system
- Personal space
  - Agenda, family photo’s, a diary, music and videos, etc.

Electronic Health Care Systems

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

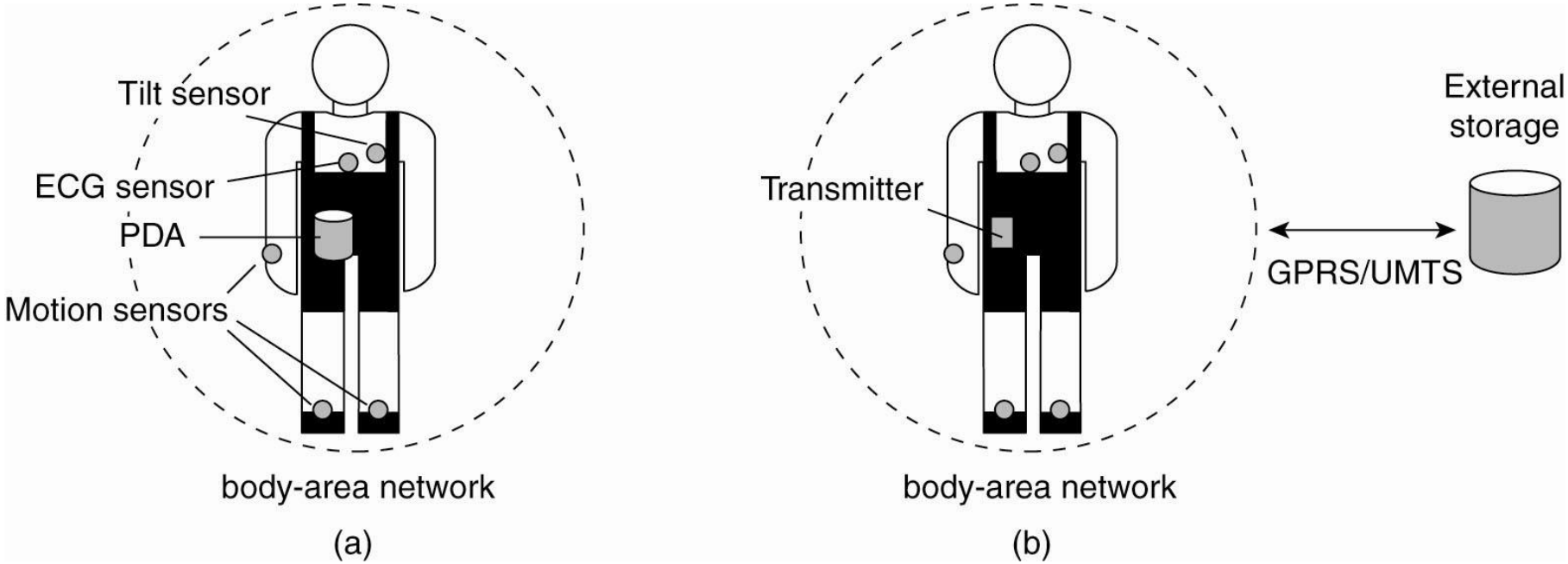


Figure 1-12. Monitoring a person in a pervasive electronic health care system, using (a) a local hub or

(b) a continuous wireless connection.

Sensor Networks

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)

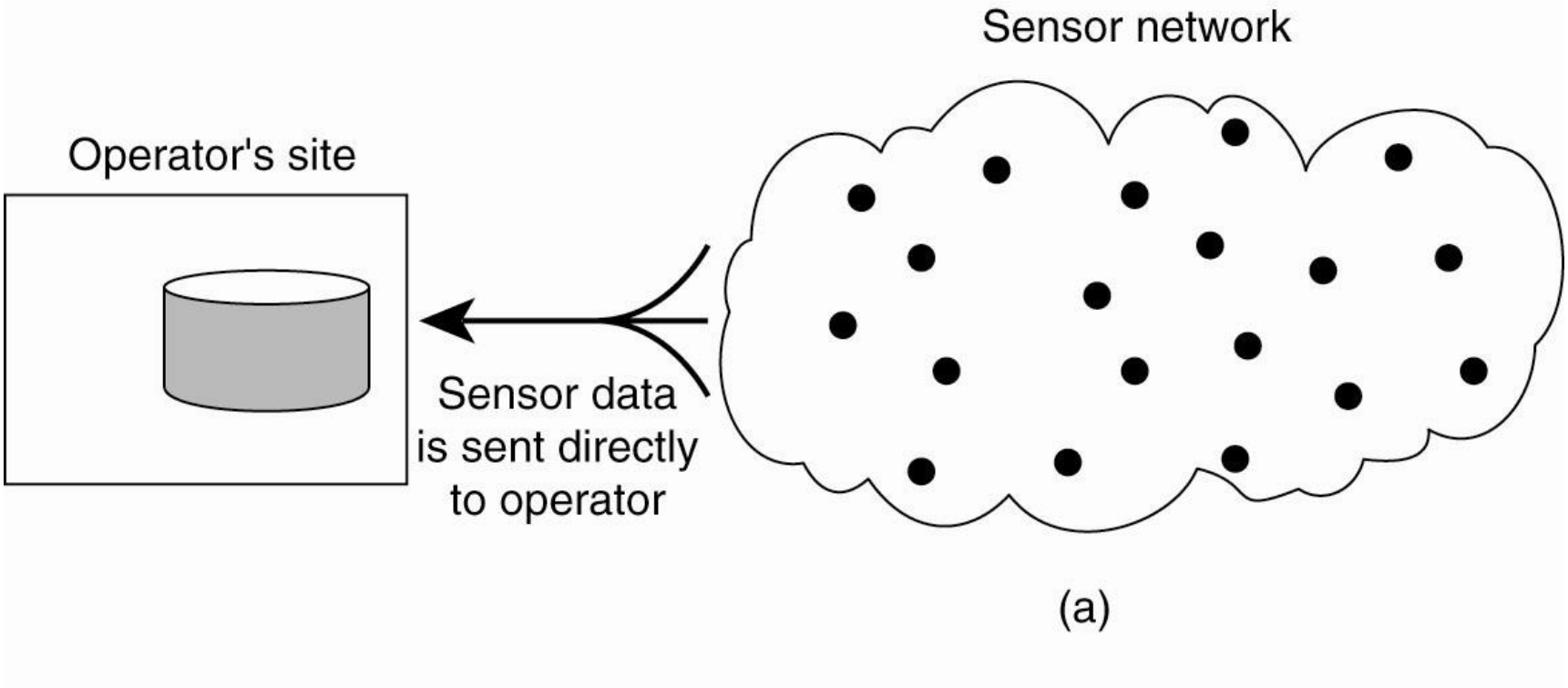


Figure 1-13. Organizing a sensor network database, while storing and processing data (a) only at the operator's site or …

Sensor Networks (3)

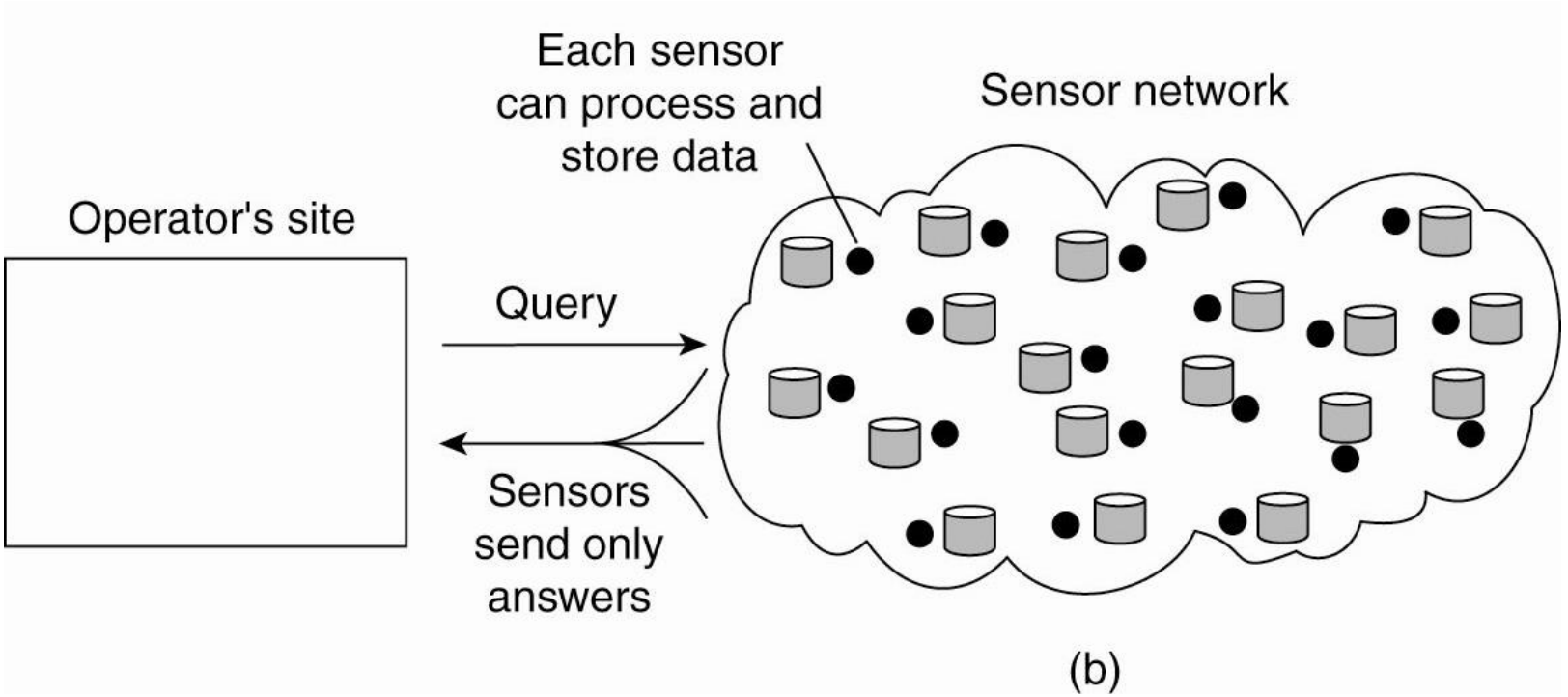


Figure 1-13. Organizing a sensor network database, while storing and processing data … or (b) only at the sensors.

Advantages of Distributed Systems over Independent PCs

- **Data sharing:** allow many users to access to a common data base
- **Resource Sharing:** expensive peripherals like color printers
- **Communication:** enhance human-to-human communication, e.g., email, chat
- **Flexibility:** spread the workload over the available machines

Advantages of Distributed Systems over Centralized Systems

- **Economics**: a collection of microprocessors offer a better price/performance than mainframes. Low price/performance ratio: cost effective way to increase computing power.
- **Speed**: a distributed system may have more total computing power than a mainframe. Ex. 10,000 CPU chips, each running at 50 MIPS. Not possible to build 500,000 MIPS single processor since it would require 0.002 nsec instruction cycle. Enhanced performance through load distributing.
- **Inherent distribution**: Some applications are inherently distributed. Ex. a supermarket chain.
- **Reliability**: If one machine crashes, the system as a whole can still survive. Higher availability and improved reliability.
- **Incremental growth**: Computing power can be added in small increments. Modular expandability
- **Another deriving force**: the existence of large number of personal computers, the need for people to collaborate and share information