Introduction Distributed Systems

Outline

- Definition of a Distributed System
- Goals of a Distributed System
- Types of Distributed Systems

By,
Er. Nabaraj Bahadur Negi

What Is A Distributed System?

- A collection of independent computers that appears to its users as a single coherent system.
- Features:
  - No shared memory – message-based communication
  - Each runs its own local OS
  - Heterogeneity
- Ideal: to present a single-system image:
  - The distributed system “looks like” a single computer rather than a collection of separate computers.

Distributed System Characteristics

- To present a single-system image:
  - Hide internal organization, communication details
  - Provide uniform interface
- Easily expandable
  - Adding new computers is hidden from users
- Continuous availability
  - Failures in one component can be covered by other components
- Supported by **middleware**

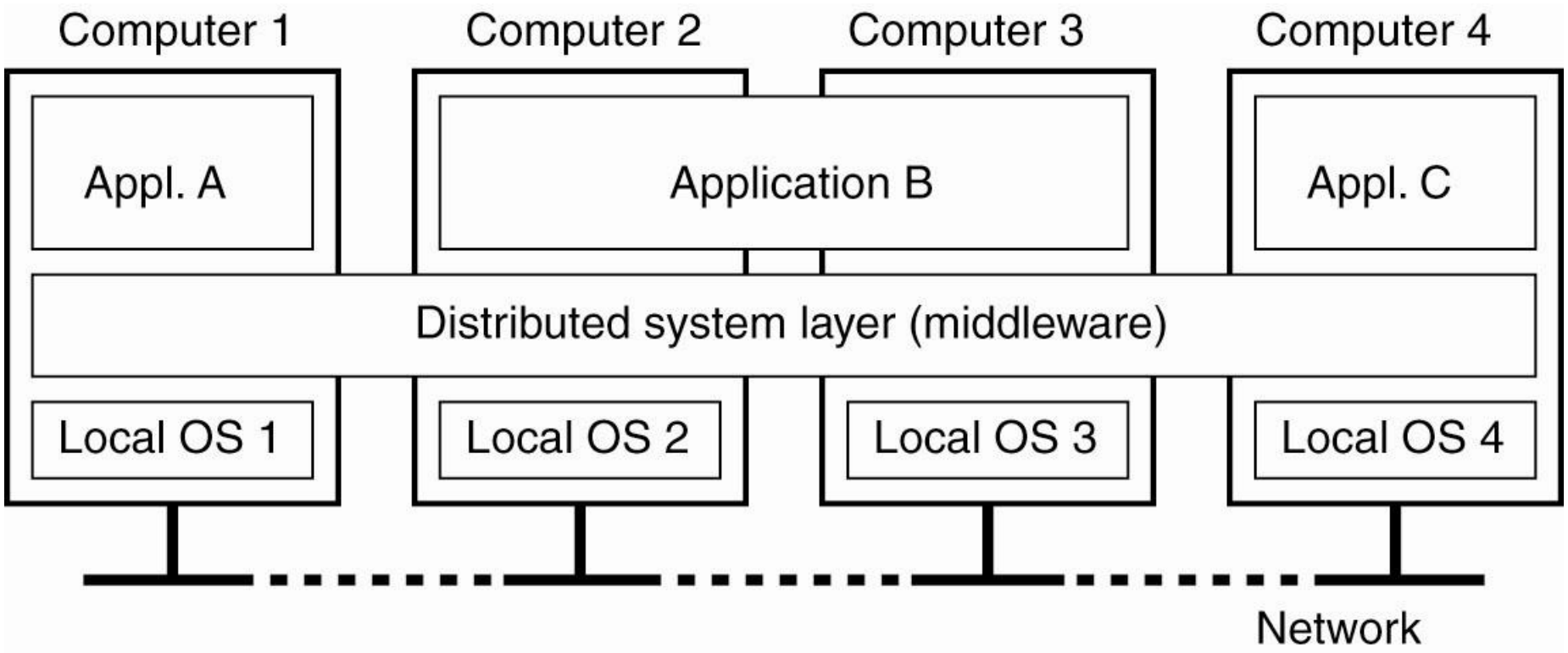

Figure 1-1. A distributed system organized as middleware. The middleware layer runs on all machines, and offers a uniform interface to the system.

- Middleware is typically used in distributed systems where it simplifies software development by doing the following:
  - Hides the intricacies of distributed applications
  - Hides the heterogeneity of hardware, operating systems, and protocols
  - Provides uniform and high-level interfaces used to make interoperable, reusable, and portable applications
  - Provides a set of common services that minimize duplication of efforts and enhances collaboration between applications

- Some of the abstractions provided by middleware include the following:
  - Remote method invocation
  - Group communication
  - Event notification
  - Object replication
  - Real-time data transmission
- In some early research systems: MW tried to provide the illusion that a collection of separate machines was a single computer.
  - E.g. NOW(Network of Workstations) project: GLUNIX (Global Layer Unix for a Network of Workstations)middleware, operating system middleware for a cluster of workstations.

- Today:
  - clustering software allows independent computers to work together closely
  - MW also supports seamless access to remote services, doesn’t try to look like a general-purpose OS
- Examples of middleware include the following:
  - Java RMI
  - CORBA
  - DCOM

Distributed System Goals

- Resource Accessibility
- Distribution Transparency
- Openness
- Scalability

Goal 1 – Resource Availability

- Support user access to remote resources (printers, data files, web pages, CPU cycles) and the fair sharing of the resources
- Economics of sharing expensive resources
- Performance enhancement – due to multiple processors; also due to ease of collaboration and info exchange – access to remote services
  - Groupware: tools to support collaboration
- Resource sharing introduces security problems.

Goal 2 – Distribution Transparency

- Software hides some of the details of the distribution of system resources.
    - Makes the system more user friendly.
- A distributed system that appears to its users & applications to be a single computer system is said to be *transparent*.
    - Users & apps should be able to access remote resources in the same way they access local resources.
- Transparency has several dimensions.

Types of Transparency

Transparency	Description
Access	Hide differences in data representation & resource access (enables interoperability)
Location	Hide location of resource (can use resource without knowing its location)
Migration	Hide possibility that a system may change location of resource (no effect on access)
Replication	Hide the possibility that multiple copies of the resource exist (for reliability and/or availability)
Concurrency	Hide the possibility that the resource may be shared concurrently
Failure	Hide failure and recovery of the resource. How does one differentiate betw. slow and failed?
Relocation	Hide that resource may be moved during use

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

Goal 2: Degrees of Transparency

- Trade-off: transparency versus other factors
  - Reduced performance: multiple attempts to contact a remote server can slow down the system – should you report failure and let user cancel request?
  - Convenience: direct the print request to my local printer, not one on the next floor
- Too much emphasis on transparency may prevent the user from understanding system behavior.

Goal 3 - Openness

- An **open distributed system** :System that offers services according to standard rules that describe the syntax and semantics of those services." In other words, the interfaces to the system are clearly specified and freely available.
  - Compare to network protocols
  - *Not* proprietary
- **Interface Definition/Description Languages (IDL):** used to describe the interfaces between software components, usually in a distributed system
  - Definitions are language & machine independent
  - Support communication between systems using different OS/programming languages; e.g. a C++ program running on Windows communicates with a Java program running on UNIX
  - Communication is usually RPC-based.

Open Systems Support …

- **Interoperability**: the ability of two different systems or applications to work together
  - A process that needs a service should be able to talk to any process that provides the service.
  - Multiple implementations of the same service may be provided, as long as the interface is maintained
- **Portability**: an application designed to run on one distributed system can run on another system which implements the same interface.
- **Extensibility**: Easy to add new components, features

Goal 4 - Scalability

- Dimensions that may scale:
  - With respect to size
  - With respect to geographical distribution
  - With respect to the number of administrative organizations spanned
- A scalable system still performs well as it scales up along any of the three dimensions.

Size Scalability

- Scalability is negatively affected when the system is based on
  - Centralized server: one for all users
  - Centralized data: a single data base for all users
  - Centralized algorithms: one site collects all information, processes it, distributes the results to all sites.
    - Complete knowledge: good
    - Time and network traffic: bad

Decentralized Algorithms

- No machine has complete information about the system state
- Machines make decisions based only on local information
- Failure of a single machine doesn’t ruin the algorithm
- There is no assumption that a global clock exists.

Geographic Scalability

- Early distributed systems ran on LANs, relied on **synchronous communication.**
  - May be too slow for wide-area networks
  - Wide-area communication is unreliable, point-to-point;
  - Unpredictable time delays may even affect correctness
- LAN communication is based on broadcast.
  - Consider how this affects an attempt to locate a particular kind of service
- Centralized components + wide-area communication: waste of network bandwidth

Scalability - Administrative

- Different domains may have different policies about resource usage, management, security, etc.
- Trust often stops at administrative boundaries
  - Requires protection from malicious attacks

Scaling Techniques

- Scalability affects performance more than anything else.
- Three techniques to improve scalability:
  - Hiding communication latencies
  - Distribution
  - Replication

Hiding Communication Delays

- Structure applications to use **asynchronous communication** (no blocking for replies)
  - While waiting for one answer, do something else; *e.g.*, create one thread to wait for the reply and let other threads continue to process or schedule another task
- Download part of the computation to the requesting platform to speed up processing
  - Filling in forms to access a DB: send a separate message for each field, or download form/code and submit finished version.
  - i.e., shorten the wait times

Distribution

- Instead of one centralized service, divide into parts and distribute geographically
- Each part handles one aspect of the job
  - Example: DNS namespace is organized as a tree of domains; each domain is divided into zones; names in each zone are handled by a different name server
  - WWW consists of many (millions?) of servers

Replication

- Replication: multiple identical copies of something
  - Replicated objects may also be distributed, but aren’t necessarily.
- Replication
  - Increases availability
  - Improves performance through load balancing
  - May avoid latency by improving proximity of resource

Summary Goals for Distribution

- Resource accessibility
  - For sharing and enhanced performance
- Distribution transparency
  - For easier use
- Openness
  - To support interoperability, portability, extensibility
- Scalability
  - With respect to size (number of users), geographic distribution, administrative domains

Issues/Pitfalls of Distribution

- Requirement for advanced software to realize the potential benefits.
- Security and privacy concerns regarding network communication
- Replication of data and services provides fault tolerance and availability, but at a cost.
- Network reliability, security, heterogeneity, topology
- Latency and bandwidth
- Administrative domains

Distributed Systems

- Early distributed systems emphasized the single system image – often tried to make a networked set of computers look like an ordinary general purpose computer
- Examples: Amoeba, Sprite, NOW, Condor (distributed batch system), …

Examples of distributed systems/applications of distributed computing :

- Intranets, Internet, WWW, email
- Telecommunication networks: Telephone networks and Cellular networks.
- The network of branch office computers -Information system to handle automatic processing of orders,
- Real-time process control: Aircraft control systems,
- Electronic banking,
- Airline reservation systems,
- Sensor networks,
- Mobile and Pervasive Computing systems.

Types of Distributed system

1. **Distributed Computing Systems :** An important class of distributed systems is the one used for high-performance computing tasks.
   a. **Cluster Computing Systems**
   b. **Grid Computing Systems**

Cluster Computing system:

- In cluster computing the underlying hardware consists of a collection of similar workstations or PCs (**homogenous**), closely connected by means of a high-speed local-area network.
- In addition, each node runs the same operating system.

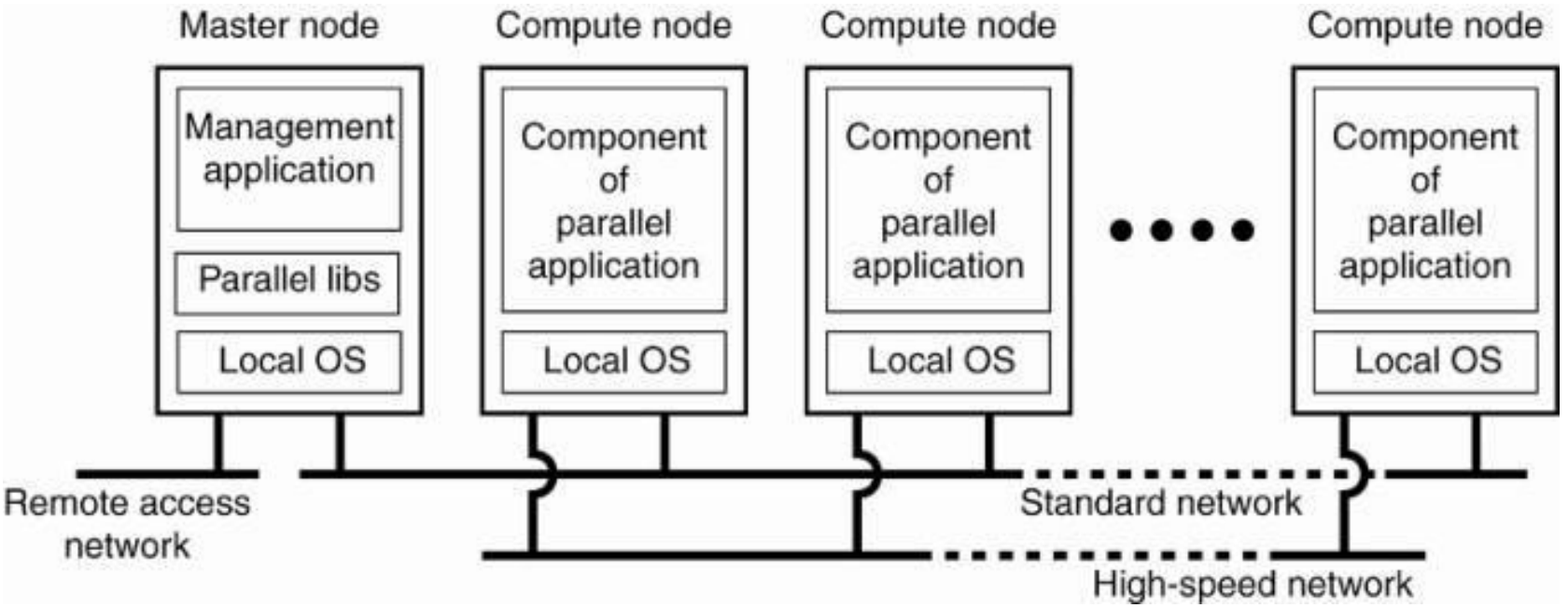

Figure 1-3. An example of a cluster computing system.

- Each cluster consists of a collection of compute nodes that are controlled and accessed by means of a single master node.
- The master typically handles the allocation of nodes to a particular parallel program, maintains a batch queue of submitted jobs, and provides an interface for the users of the system.
- Some popular implementations of cluster computing are the **Google search engine, Earthquake Simulation, Petroleum Reservoir Simulation, and Weather Forecasting systems**.

Grid Computing Systems:

- Grid computing is a computing infrastructure that combines computer resources spread over different geographical locations(**heterogeneity)** to achieve a common goal.
- Users and resources from different organizations are brought together to allow collaboration, Such a collaboration is realized in the form of a **virtual organization.**
- The processes belonging to the same virtual organization have access rights to the resources that are provided to that organization.
- For example, meteorologists use grid computing for weather modeling.
- The architecture consists of four layers.
    - The **Fabric layer** provides interfaces to local resources at a specific site.
    - The **connectivity layer** consists of communication protocols for supporting grid transactions that span the usage of multiple resources. For example, protocols are needed to transfer data between resources, or to simply access a resource from a remote location.

- The **resource layer** is responsible for managing a single resource. It uses the functions provided by the connectivity layer and calls directly the interfaces made available by the fabric layer.
- The **collective layer** deals with handling access to multiple resources and typically consists of services for resource discovery, allocation and scheduling of tasks onto multiple resources, data replication, and so on.
- The **application layer** consists of the applications that operate within a virtual organization and which make use of the grid computing environment.

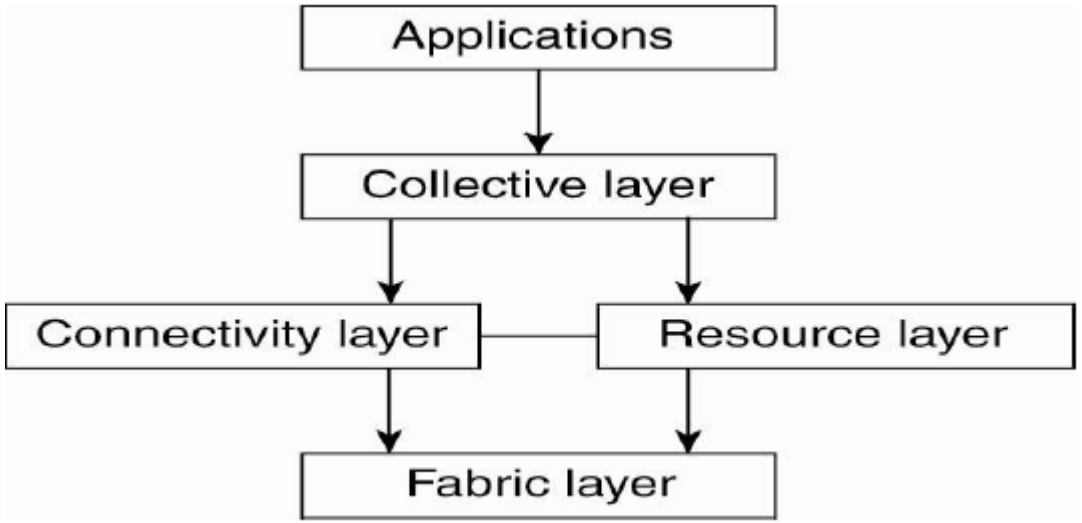

Figure 1-4. A layered architecture for grid computing systems.

2. Distributed Information Systems: Focus on interoperability

a. **Transaction Processing Systems**

b. **Enterprise Application Integration (Exchange info via RPC or RMI)**

Transaction processing system:

- Captures and processes every single **transaction** that takes place within the organization.

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

Figure 1-5. Example primitives for transactions.

Characteristic properties of transactions: **(ACID** properties)

- **Atomic:** 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.

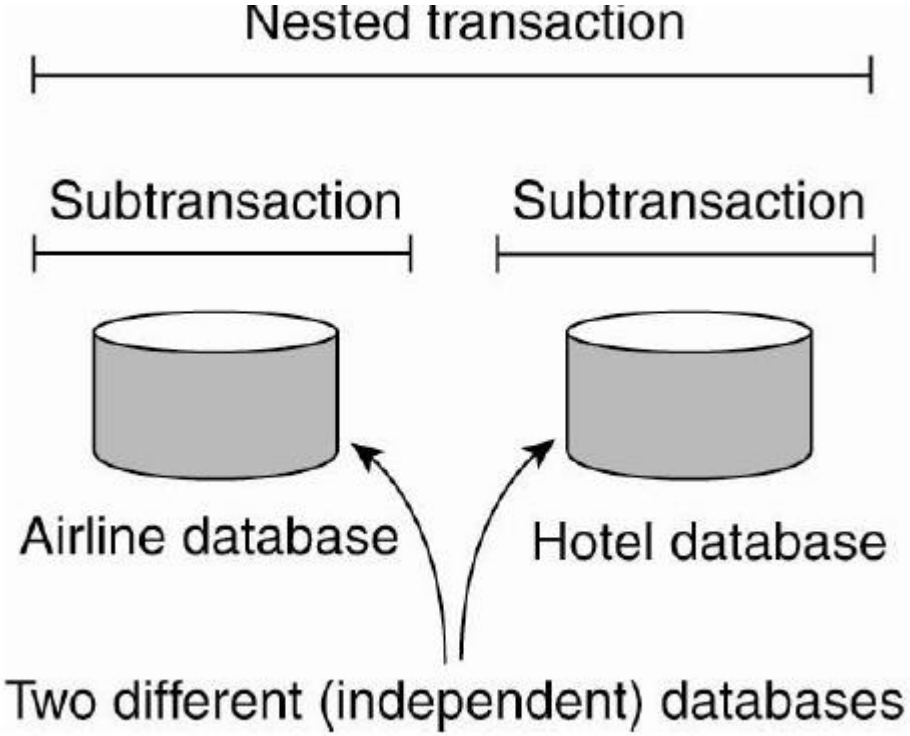

Figure 1-6. A nested transaction.

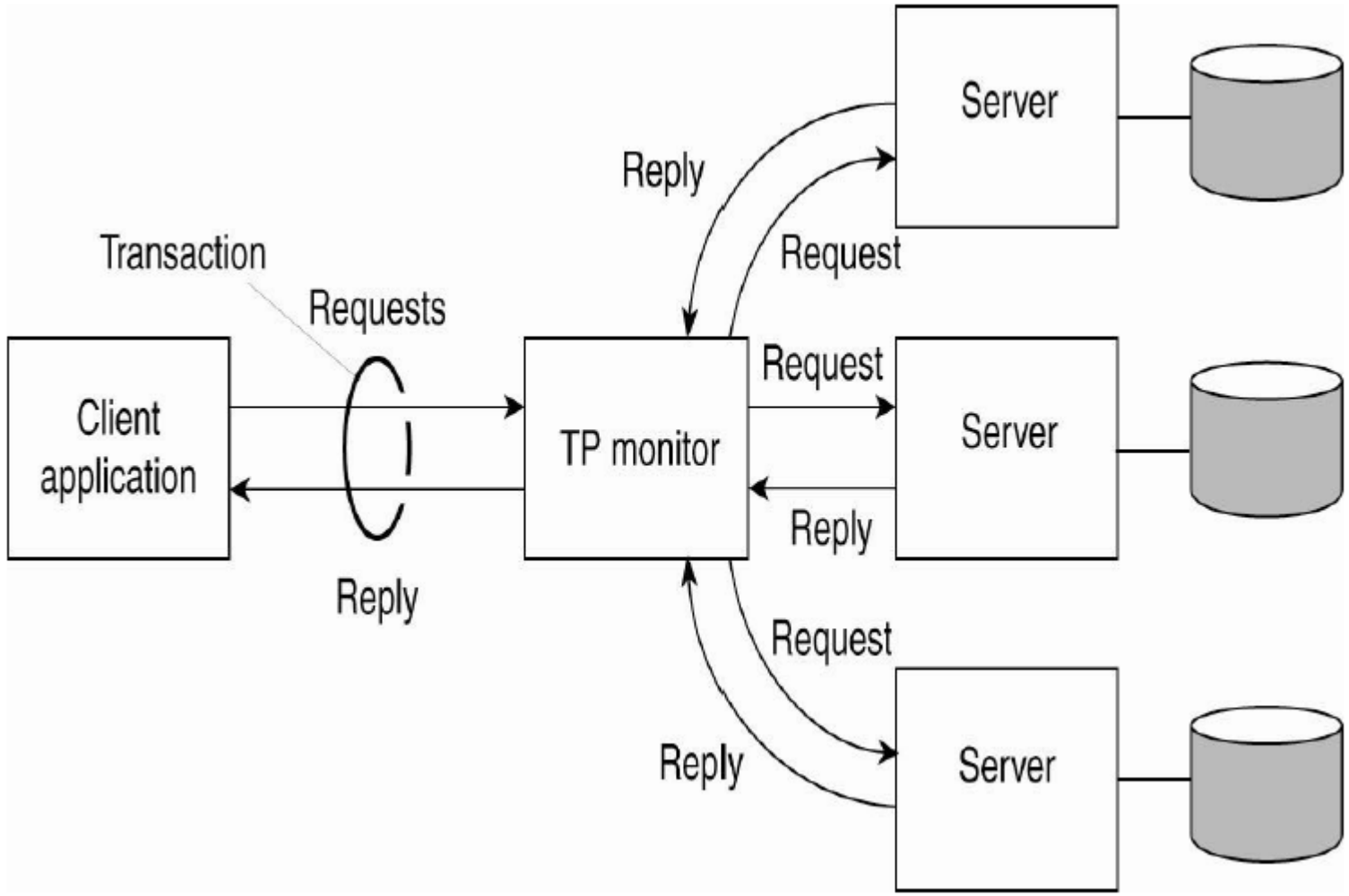

Figure 1-7. The role of a TP monitor in distributed systems.

Enterprise Application Integration:

- Enterprise Application Integration (EAI) is a collection of technologies and services that form an intermediary to enable the integration of independent systems and applications. EAI acts as middleware to link applications.

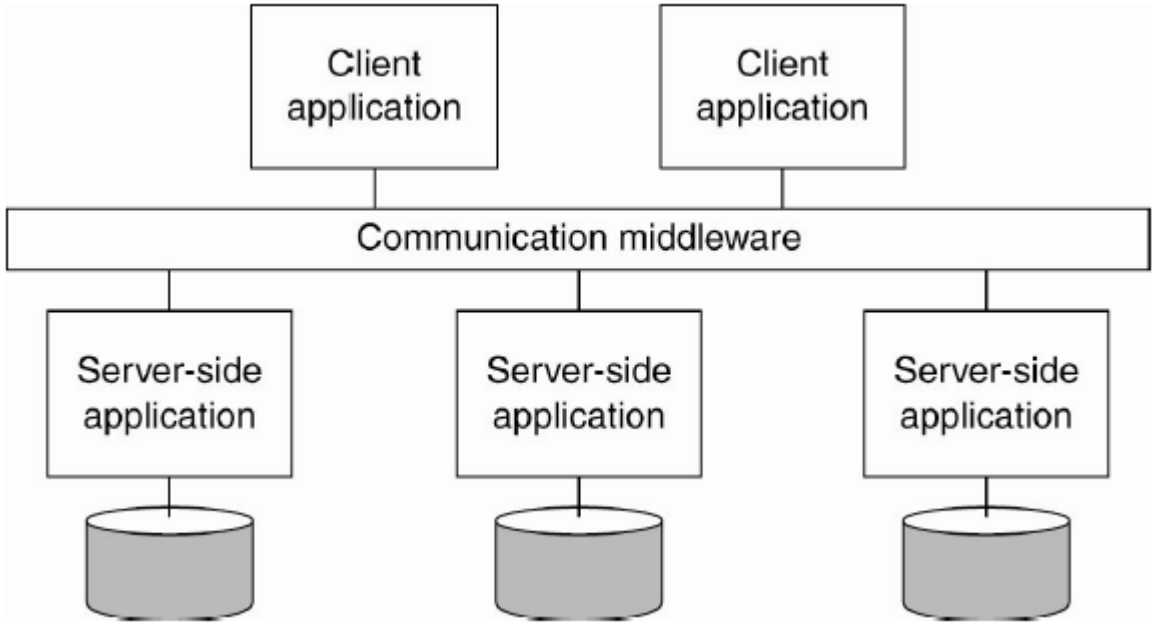

Figure 1-8. Middleware as a communication facilitator in enterprise application integration.

- Remote Procedure Calls (RPC) and Remote Method Invocations (RMI) are an example of middleware. With RPC, application A calls application B by performing a local procedure call, the result of which is sent back to A by B with a message. RMI is an evolution from RPC.
- An RMI is essentially the same as an RPC, except that it operates on objects instead of functions.
- RPC and RMI have the disadvantage that the caller and callee both need to be up and running at the time of communication.

3. Distributed Pervasive Systems

- Usually small, battery-powered nodes used in embedded systems.
- Includes contextual change, i.e. the system perceives its environment to change continuously. For example: suddenly realized... lost the network
- Encourage ad-hoc components, i.e. many diverse devices in the system that are used in different ways by different users. Therefore, it should be easy to configure the application framework running on the device by the user or adjust automatically (but with control).
- Recognize sharing is the default

a. Home Systems (e.g. Smart phones, PDAs)
b. Electronic Health care systems (Heart monitors, BAN: Body Area Networks)
c. Sensor Networks (distributed Databases connected wirelessly)

Home Systems: (e.g. Smart phones, PDAs)

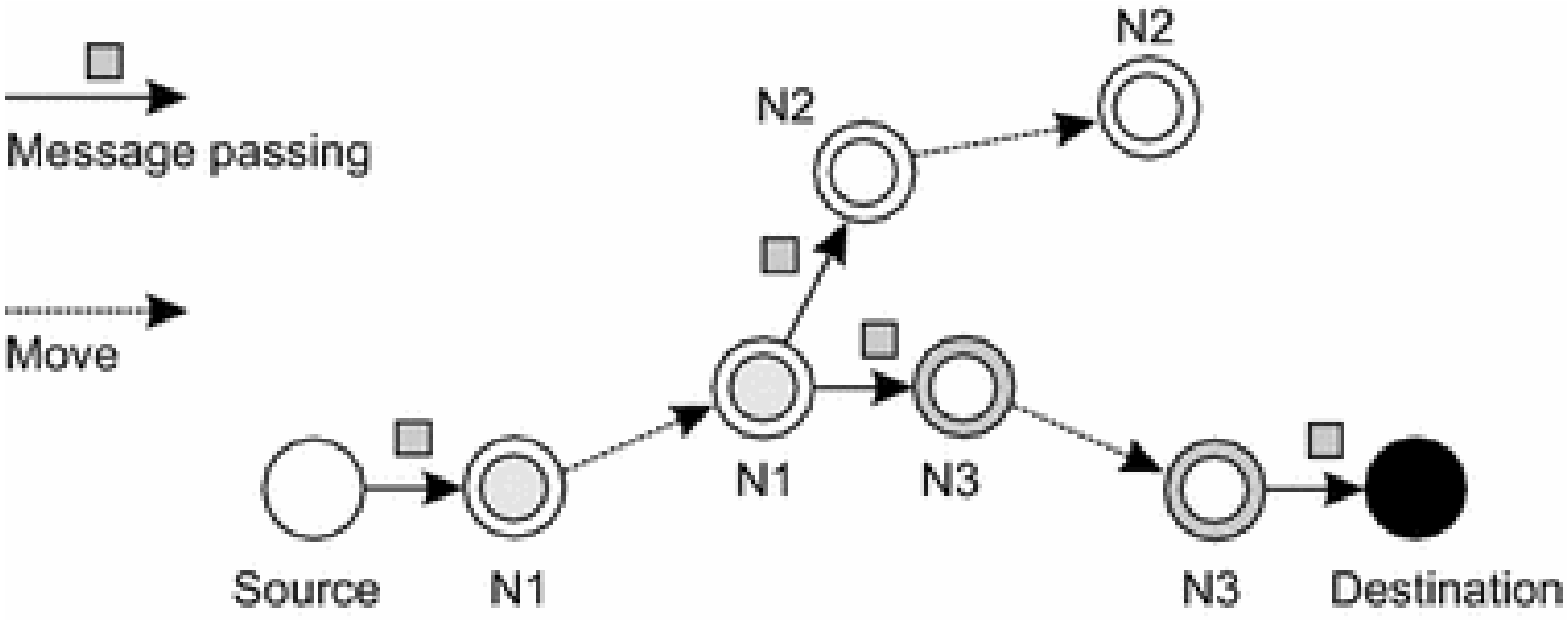

Figure 1-9. Passing messages in a (mobile) disruption-tolerant network.

Electronic Health Care Systems:

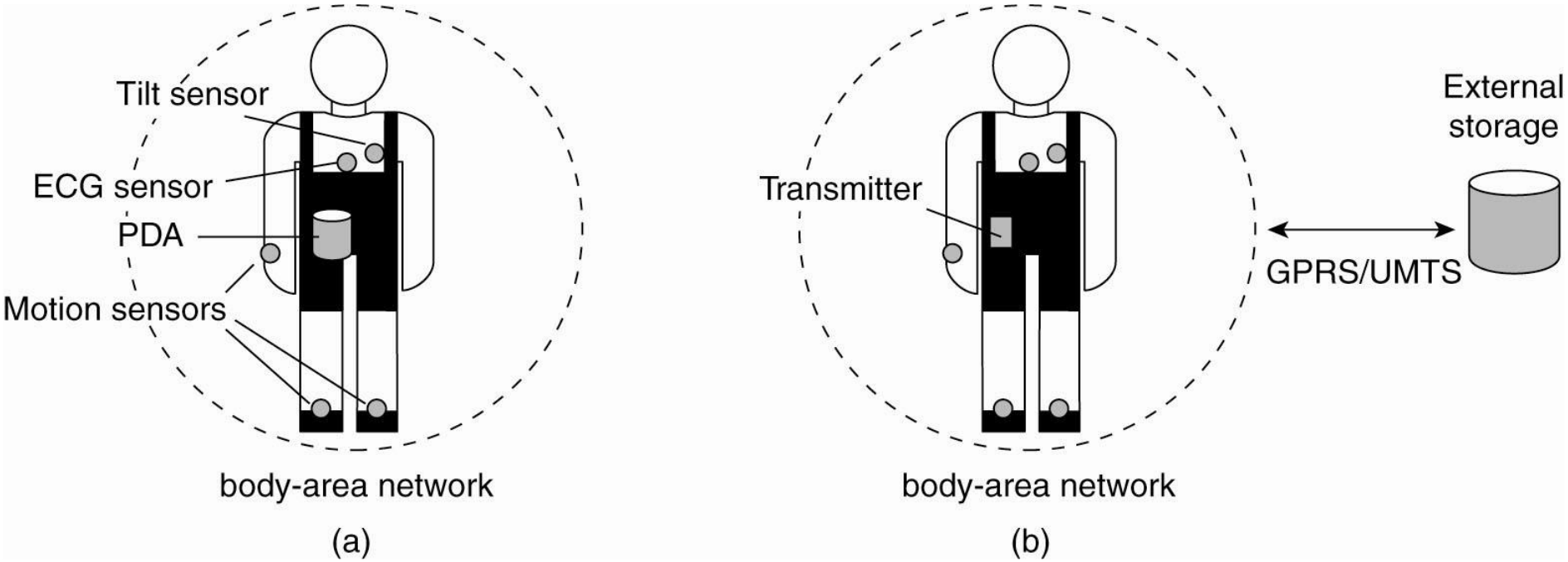

Figure 1-10. Monitoring a person in a pervasive electronic health care system, using (a) a local hub or -(b) a continuous wireless connection.

Sensor Networks:

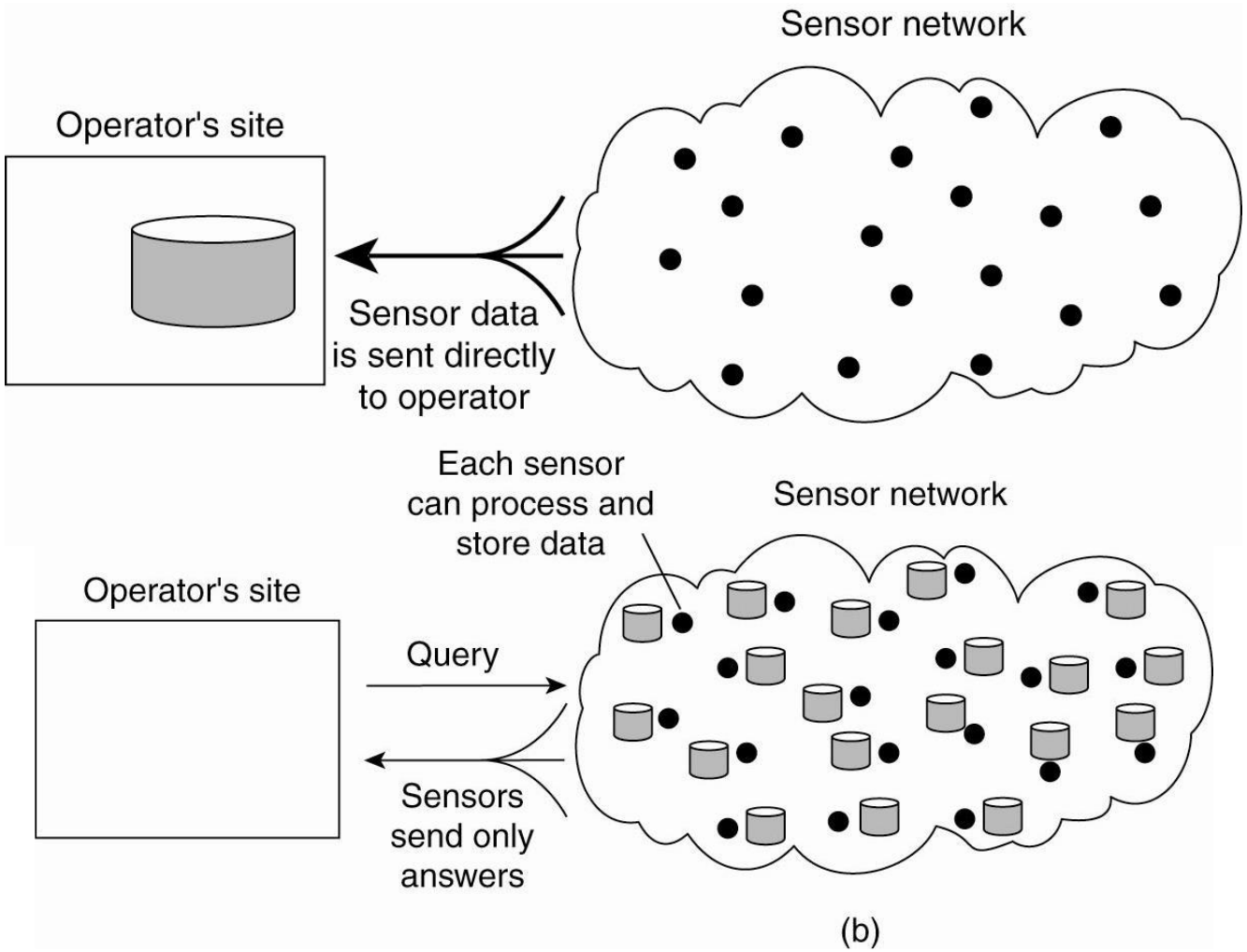

Figure 1-12. Organizing a sensor network database, while storing and processing data (a) only at the operator's site or (b) only at the sensors.