Administrarea Bazelor de Date
Managementul în Tehnologia Informaţiei

Sisteme Informatice şi Standarde Deschise (SISD)

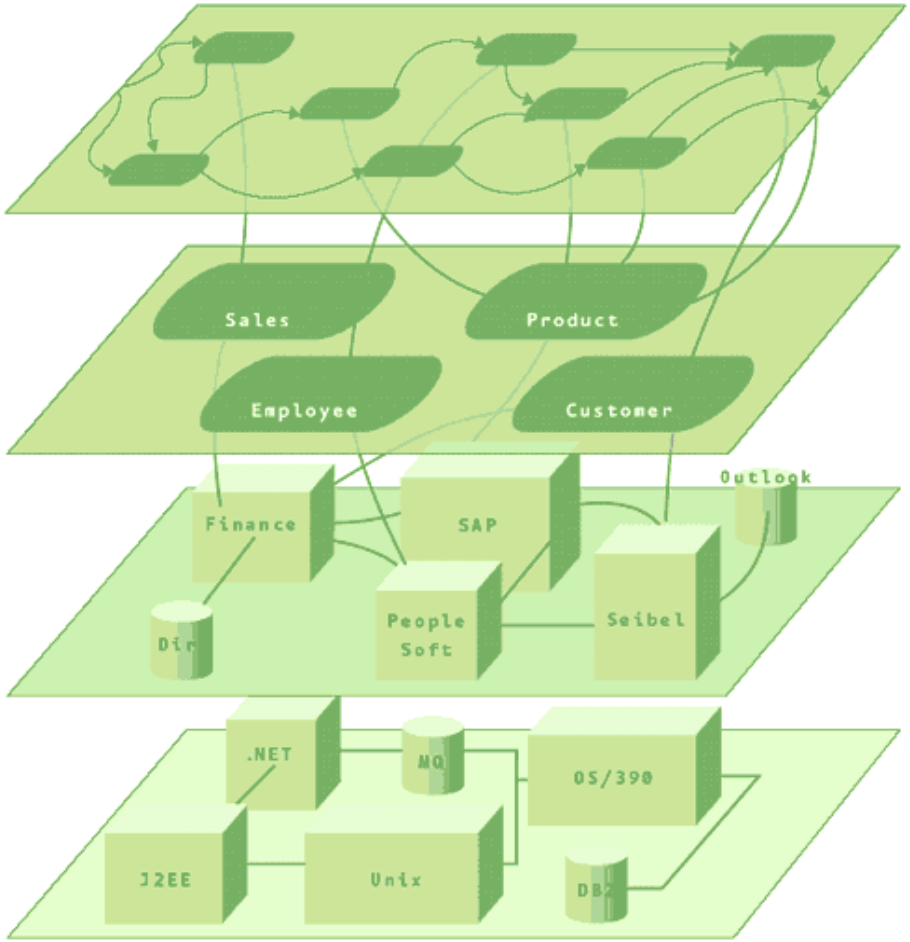
2009-2010

Curs 2

Introducere în sisteme informatice şi standarde deschise

Information systems

- **Information Systems** (IS) refers to the interaction between people, processes, and technology.



Business Process Layer

Business Service Layer

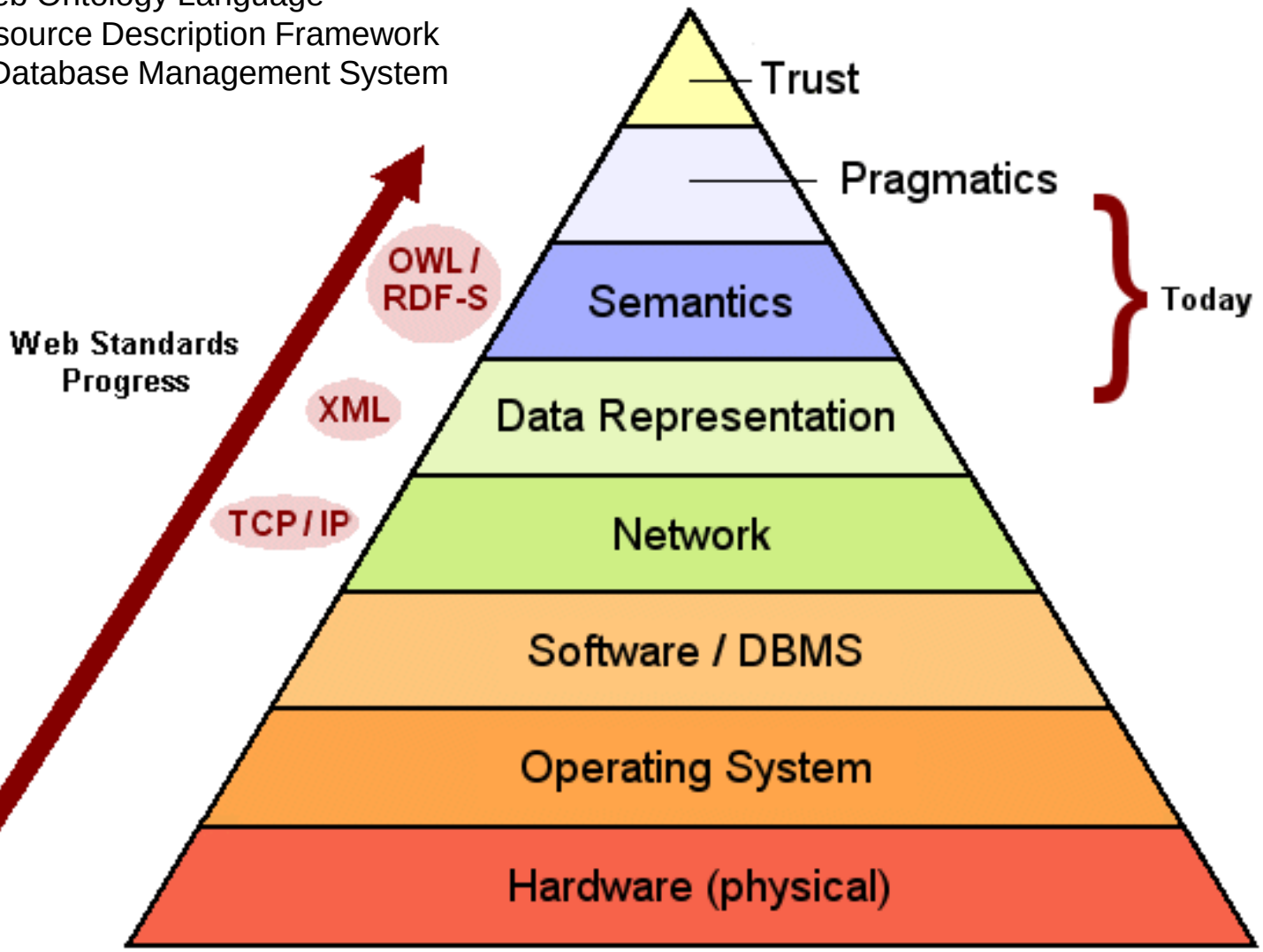
Application Layer

Technology Layer

Sursa: http://msdn.microsoft.com

Information systems pyramid

OWL - Web Ontology Language
RDF - Resource Description Framework
DBMS – Database Management System

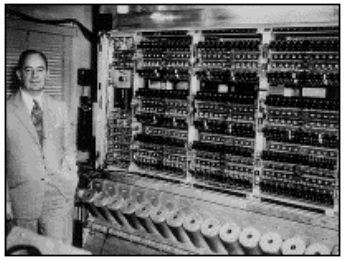


Types of information systems

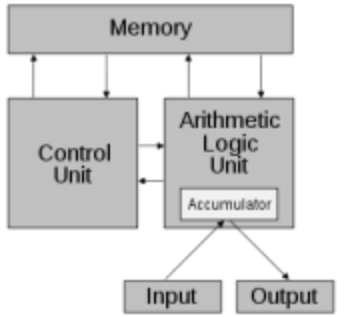
- **Transaction processing systems**
  - Tasks execution, data storage, communication
- **Management information systems**
  - Internal control of applications
- **Decision support systems**
  - support business and organizational decision-making activities
- **Executive information systems**
  - facilitate and support the information and decision-making

- Data warehouses
- Enterprise resource planning
- Enterprise systems
- Expert systems
- Global information system
- Office Automation

IT Systems



- **von Neumann architecture**
  - ENIAC (1946) – first modern electronic computer (Electronic Numerical Integrator And Computer)
  - TRADIC (1954) - first solid-state (or transistor) computer (**TRA**nsistor **DI**gital **C**omputer)



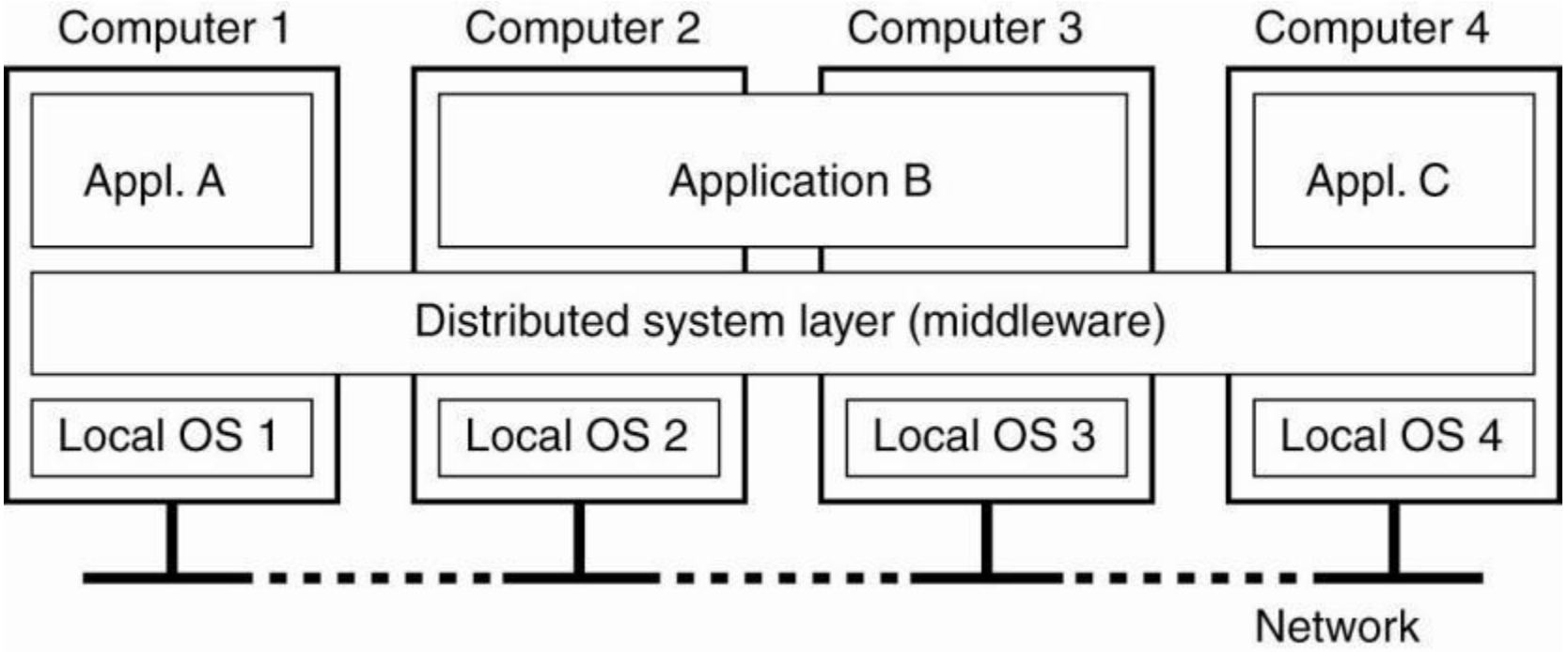
- **Bare hardware**
  - no operating system
- **Computer operators**
- **Device drivers and library functions (1950s)**
- **Microprocessors (1960s)**

- **Operating Systems & Programming Languages (1970s)**
  - DOS (DOS/360, DOS/VS) and Unix (Ken Thompson, Bell Labs)
  - ADA, FORTRAN, ALGOL, LISP, C, SmallTalk, Prolog, SQL
- **Distributed Systems**

What Is A Distributed System?

- A collection of independent computers that appears to its users as a single coherent system.
- Characteristics:
  - No shared memory – message-based communication
  - Each runs its own local OS
  - Heterogeneity
- Single-system image:
  - Hide internal organization, communication details
  - Provide uniform interface

Definition of a Distributed System



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

Source: A.Tanenbaum

Why Have Distributed Systems?

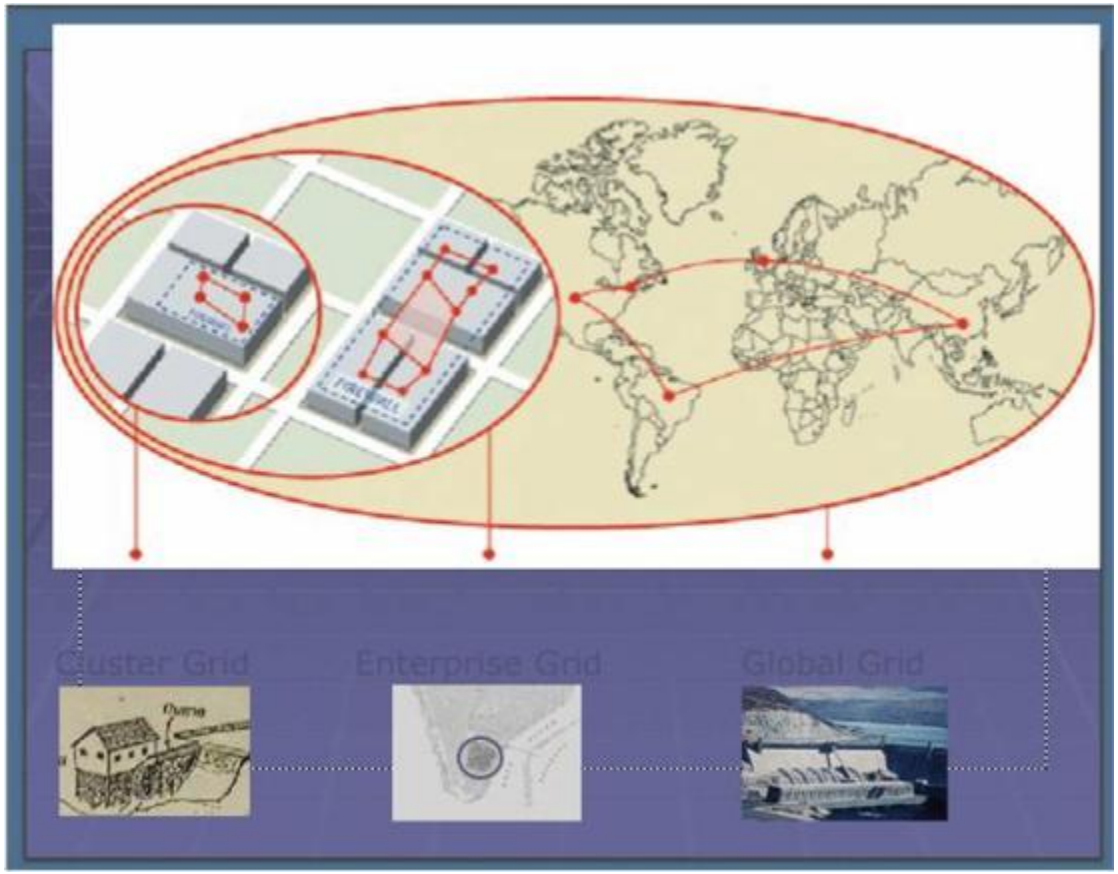
- Share resources at remote sites
- Price/performance advantages
  - Support applications with large computational requirements
- Cooperative communities – collaborate with colleagues at a distance.

Potential Advantages of Distribution

- **Reliability**: time between failures is large
- **Availability**: the system is almost always ready to satisfy a user's request.
  - An available system is not necessarily reliable: e.g., a system that is down one millisecond/hour has an availability of ~99.99%, but is not very reliable.
- **Fault Tolerance**: The ability to continue to operate in spite of failures: hardware or software

Distributed System Goals

- Resource Accessibility
- Distribution Transparency
- Openness
- Scalability



Goals – Resource Availability

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

Goals – Distribution Transparency

- Software hides some of the details of the system to make it more user friendly.
- A distributed system that appears to its users & applications to be a single computer system is said to be *transparent*.
  - Users & app's should be able to access remote resources in the same way they access local resources.

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
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
Relocation	Hide that resource may be moved during use

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

Goal - Openness

- An **open distributed system** "…offers services according to standard rules that describe the syntax and semantics of those services."
  - Compare to network protocols
- **Interface Definition Language (IDL):** used to describe the interface to a distributed system (parameters, return values, etc.)
  - Difficulty: semantics –what do the services do?
- **Interoperability**: the ability of two different systems or applications to work together "… by relying on each other's services as specified by a common standard."
  - Any process that needs a service should be able to communicate with a process that provides the service.
  - Multiple implementations of the same service may be provided, as long as the interface is maintained
- **Portability:** the ability of an application designed to run on one distributed system to run on another system which implements the same interface.
- **Extensibility:** Easy to add new components, features

Goal - Scalability

- Dimensions that may scale
  - With respect to size
  - With respect to geographical distribution
  - With respect to the number of administrative organizations it spans
- 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 network communication is unreliable, point-to-point; LAN communication is based on broadcast.
  - Consider how this affects an attempt to locate a particular kind of service

Scalability - Administrative

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

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; 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.

Distribution

- Instead of one centralized service, divide into parts and distribute geographically
  - 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

Scaling Technique - Replication

- Replication: multiple identical copies of something
- Replication
  - Increases availability
  - Improves performance through load balancing
  - May avoid latency by improving proximity of resource

Caching

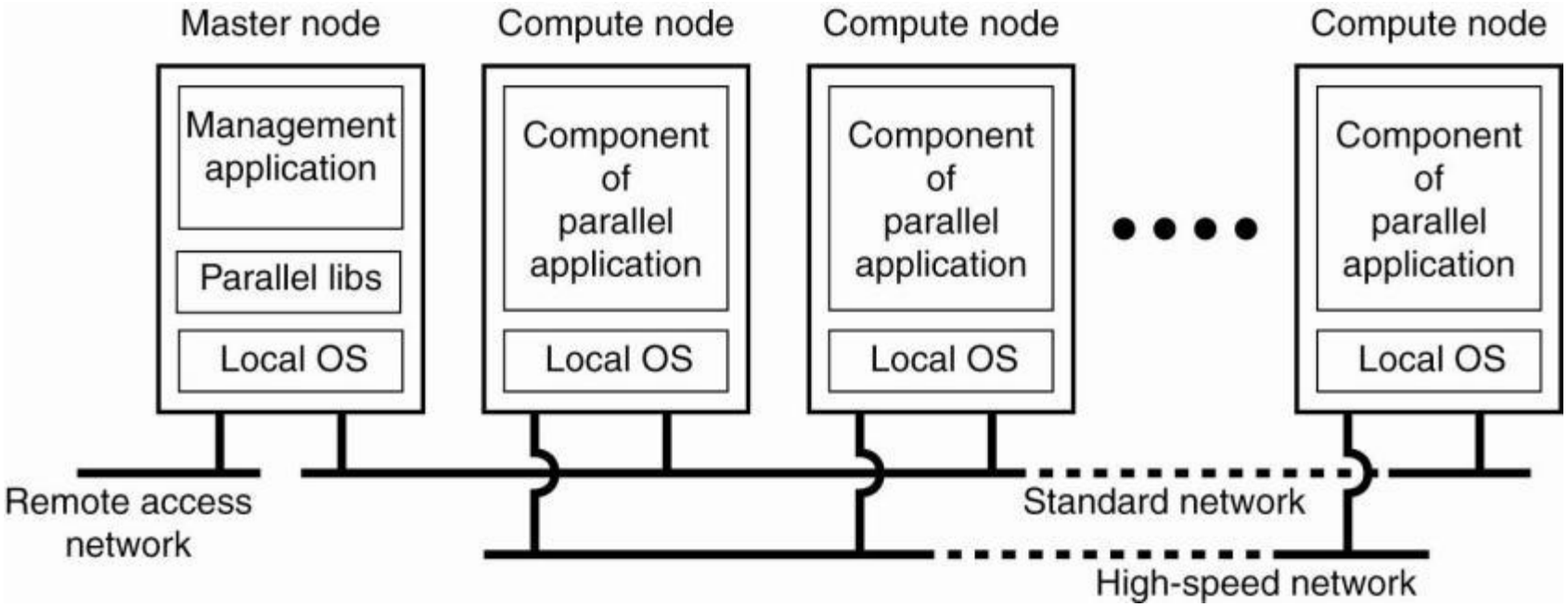
- Caching is a form of replication
  - Normally creates a (temporary) replica of something closer to the user
- User decides to cache, system decides to replicate
- Replication is more permanent
- Both lead to **consistency** problems

Types of Distributed Systems

- Distributed Computing Systems
  - Clusters
  - Grids
- Distributed Information Systems
  - Transaction Processing Systems
  - Enterprise Application Integration
- Distributed Embedded Systems
  - Home systems
  - Health care systems
  - Sensor networks

Cluster Computing

- A collection of similar processors (PCs, workstations) running the same operating system, connected by a high-speed network.
- Parallel computing capabilities using inexpensive PC hardware
- Used to run parallel programs or in server farms (e.g., at banks)
- Microsoft, Sun, and others sell clustering software and you can also buy turnkey systems

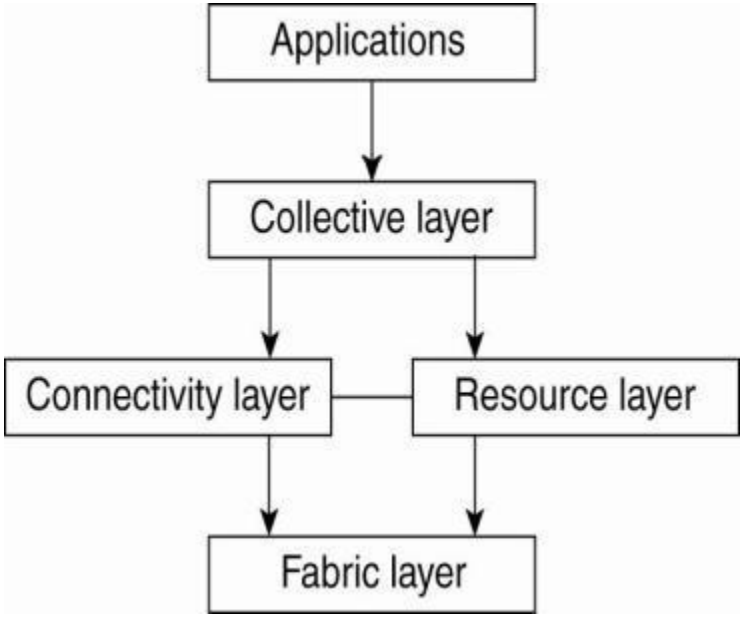


Grid Computing Systems

- Modeled loosely on the electrical grid.
- Highly heterogeneous with respect to hardware, software, networks, security policies, etc.
- Grids support **virtual organizations**: a collaboration of users who pool resources (servers, storage, databases) and share them
- Grid software is concerned with managing sharing across administrative domains.

A Proposed Architecture for Grid Systems

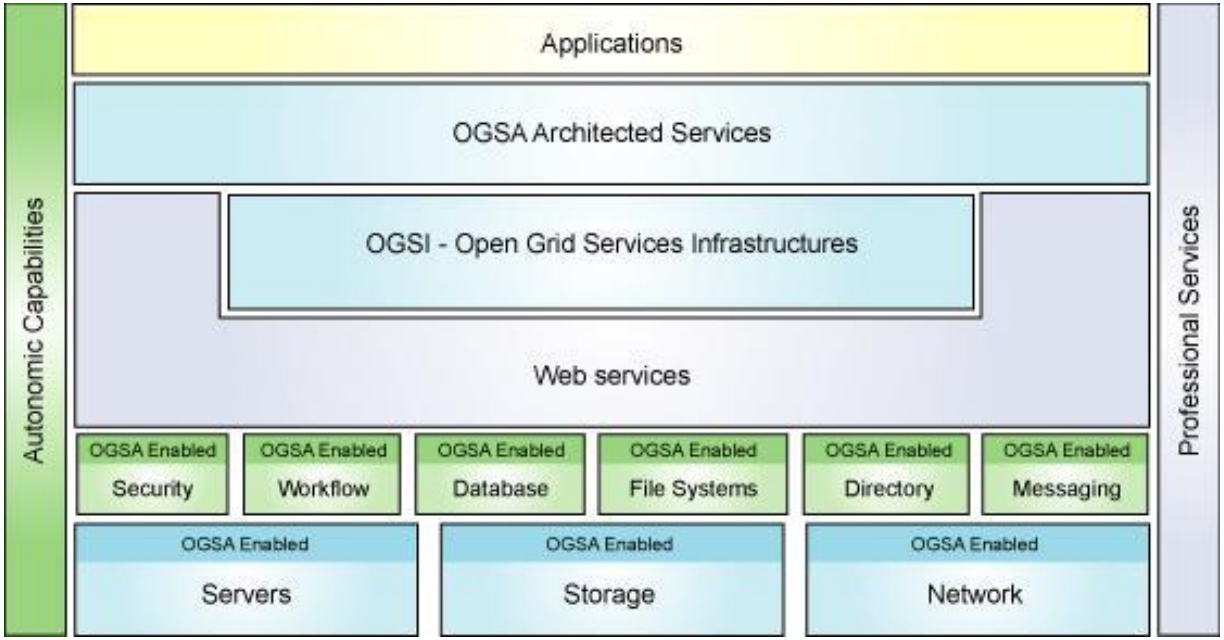
- **Fabric layer**: interfaces to local resources
- **Connectivity layer**: supports usage of *multiple resources* for a single application; e.g., access a remote resource or transfer data between sites
- **Resource layer** manages a *single resource*
- **Collective layer:** resource discovery, allocation, etc.
- **Applications**: use the grid resources
- The collective, connectivity and resource layers together form the middleware layer for a grid.



A layered architecture for grid computing systems

OGSA – Another Grid Architecture

- **Open Grid Services Architecture (OGSA)** is a service-oriented architecture
  - Sites that offer resources to share do so by offering specific Web services.
- The architecture of the OGSA model is more complex than the previous layered model.



Distributed Information Systems

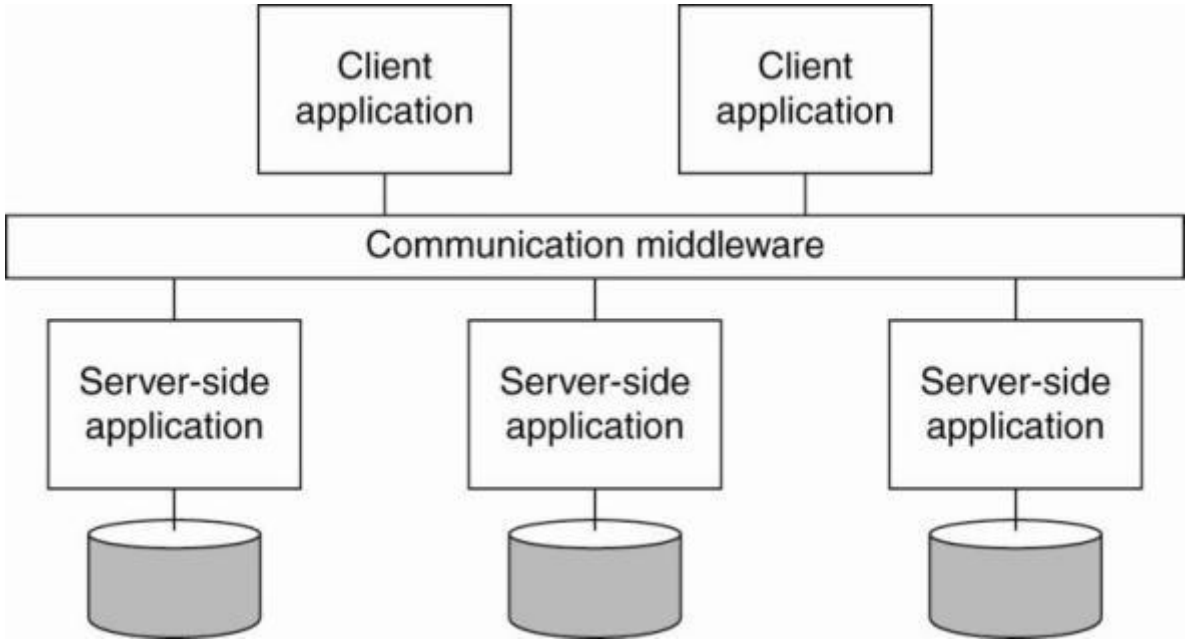
- Business-oriented
- Systems to make a number of separate network applications interoperable and build “enterprise-wide information systems”.
- Two types discussed here:
  - Transaction processing systems
  - Enterprise application integration

Transaction Processing Systems

- Provide a highly structured client-server approach for database applications
- Transactions are the communication model
- Obey the **ACID** properties:
  - **A**tomic: all or nothing
  - **C**onsistent: invariants are preserved
  - **I**solated (serializable)
  - **D**urable: committed operations can’t be undone

Enterprise Application Integration

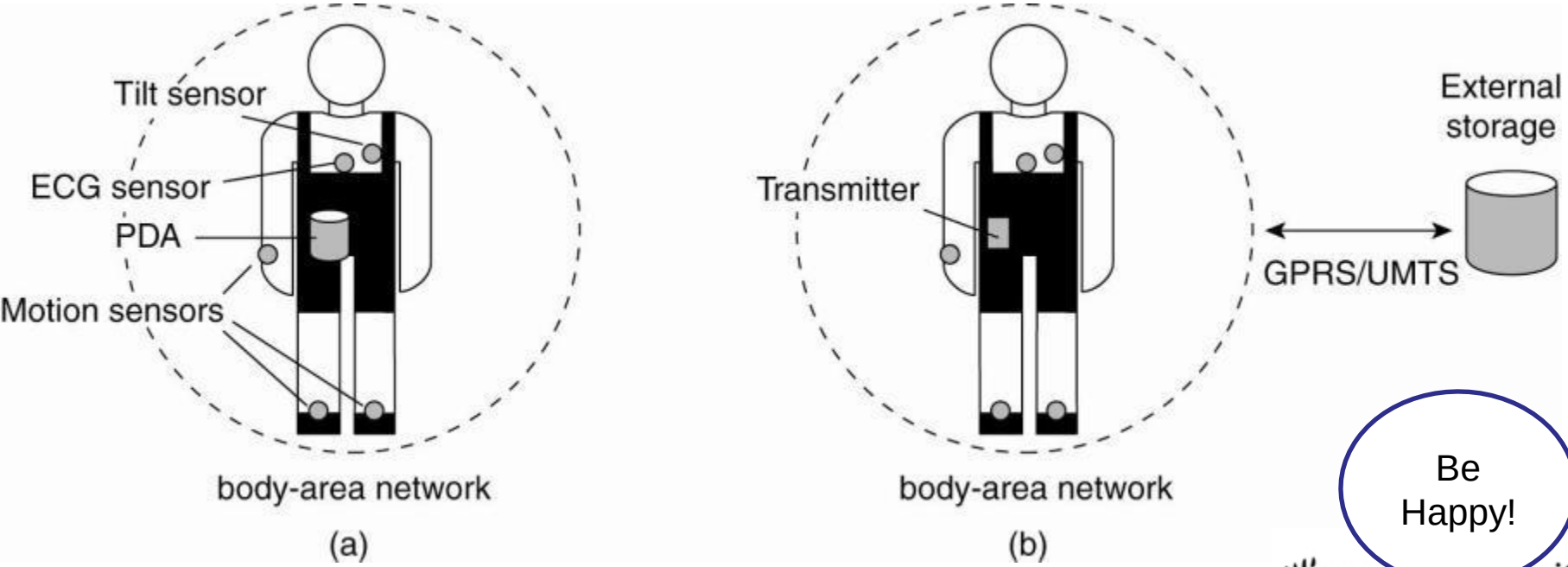
- Supports a less-structured approach (as compared to transaction-based systems)
- Application components are allowed to communicate directly
- Communication mechanisms to support this include CORBA, Remote Procedure Call (RPC) and Remote Method Invocation (RMI)
  - Both parties must be running to participate in this kind of communication



Distributed Pervasive Systems

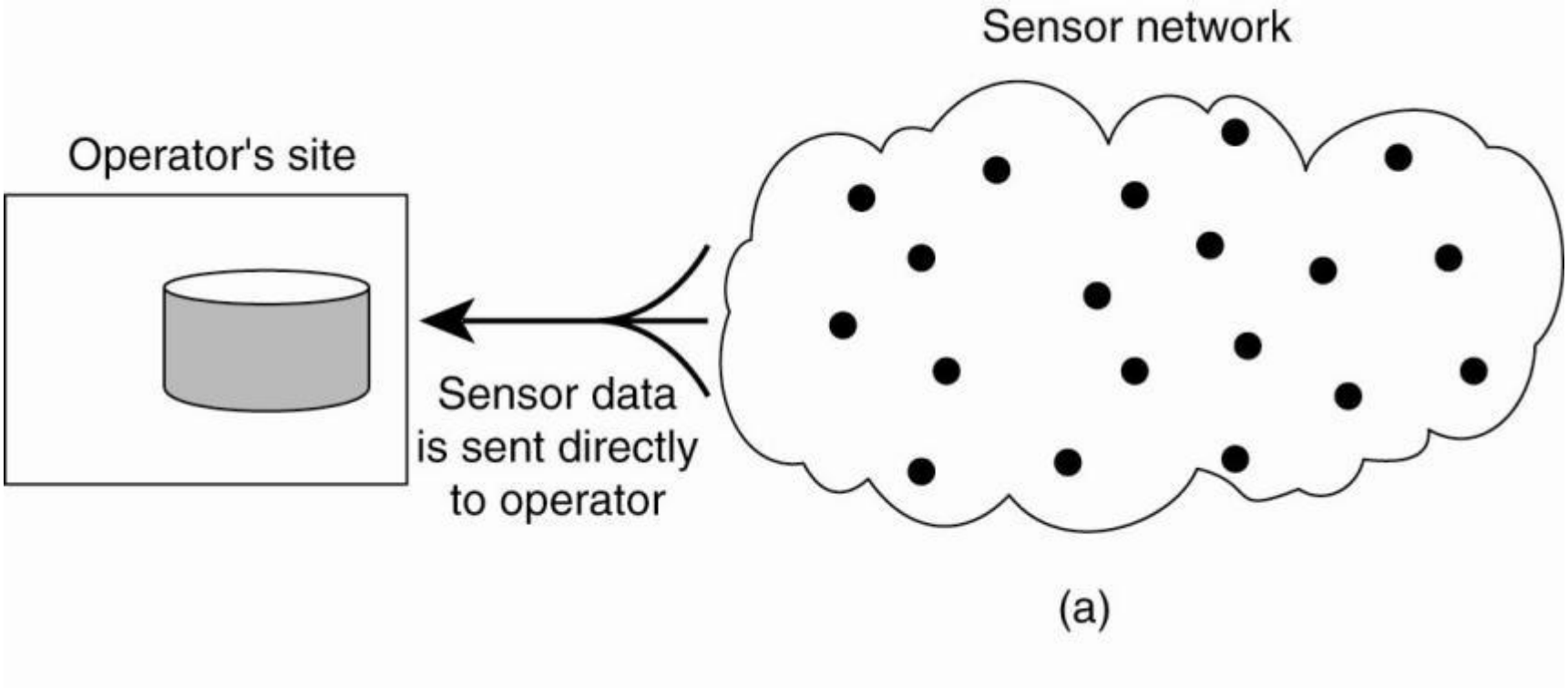
- The first two types of systems are characterized by their stability: nodes and network connections are more or less fixed
- This type of system is likely to incorporate small, battery-powered, mobile devices
  - Home systems
  - Electronic health care systems – patient monitoring
  - Sensor networks – data collection, surveillance

Electronic Health Care Systems



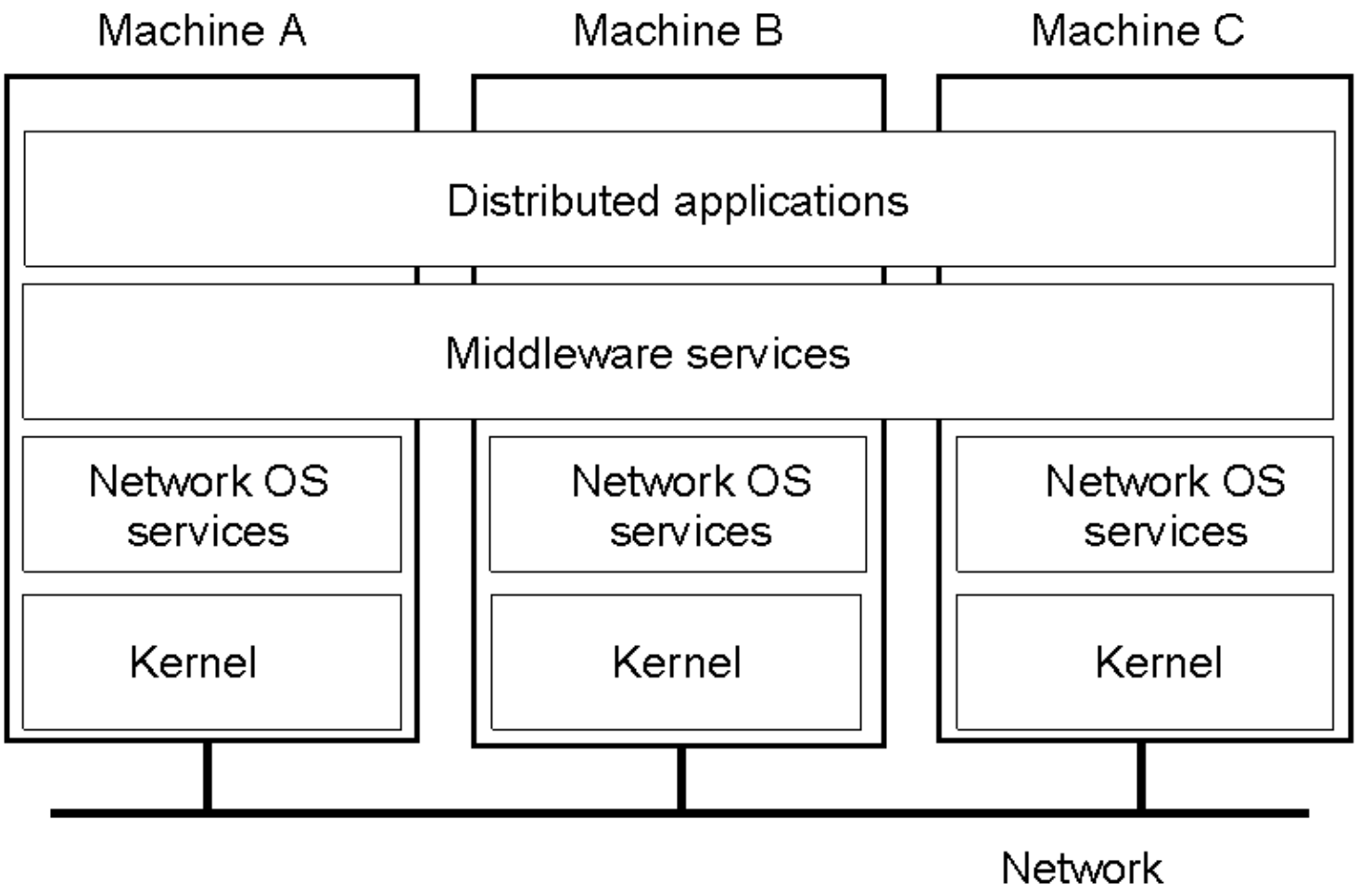
Monitoring a person in a pervasive electronic health care system, using (a) a local hub or (b) a continuous wireless connection.

Sensor Networks



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

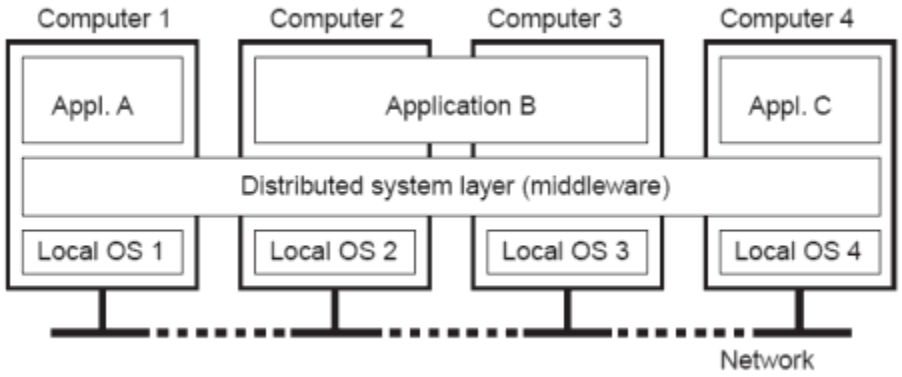
Middleware


Machine A
Machine B
Machine C
Distributed applications
Middleware services
Network OS services
Network OS services
Network OS services
Kernel
Kernel
Kernel
Network

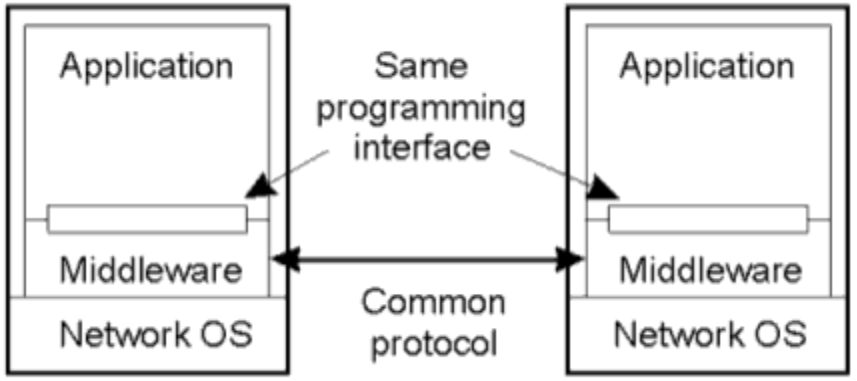
Middleware Examples

- CORBA (Common Object Request Broker Architecture)
- DCOM (Distributed Component Object Management)
- Sun’s ONC RPC (Remote Procedure Call)
- RMI (Remote Method Invocation)
- SOAP (Simple Object Access Protocol)

Middleware and Interoperability



- *Independent* hardware installations
- *Uniform* software layer (middleware)

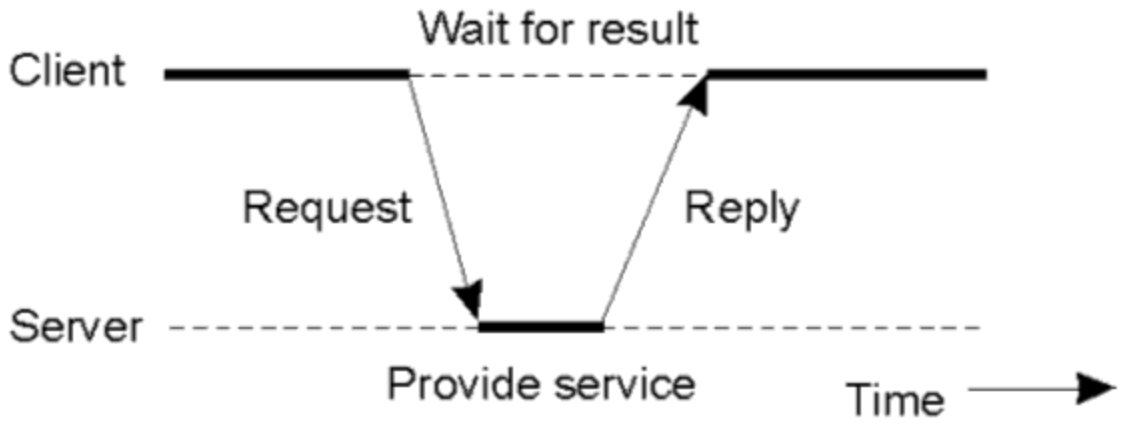


Interoperability provided by:

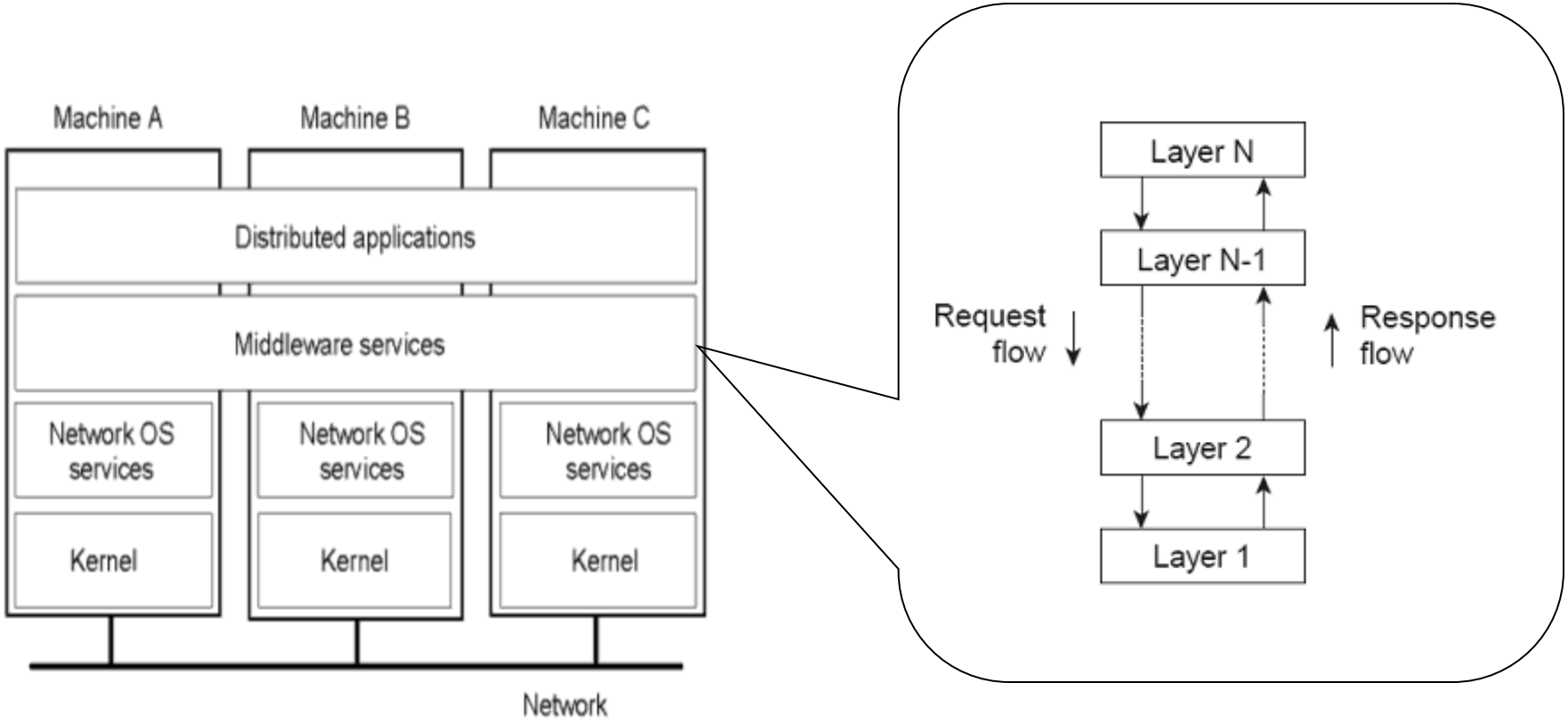
- *Protocols* used by each middleware layer
- *Interfaces* offered to applications

Architecture styles: Client/server

- Model:
  - Server: process implementing a certain service
  - Client: uses the service buy sending a request and waiting for the reply
- Main problem to deal with: unreliable communication
- Note: often both roles simultaneously for different services

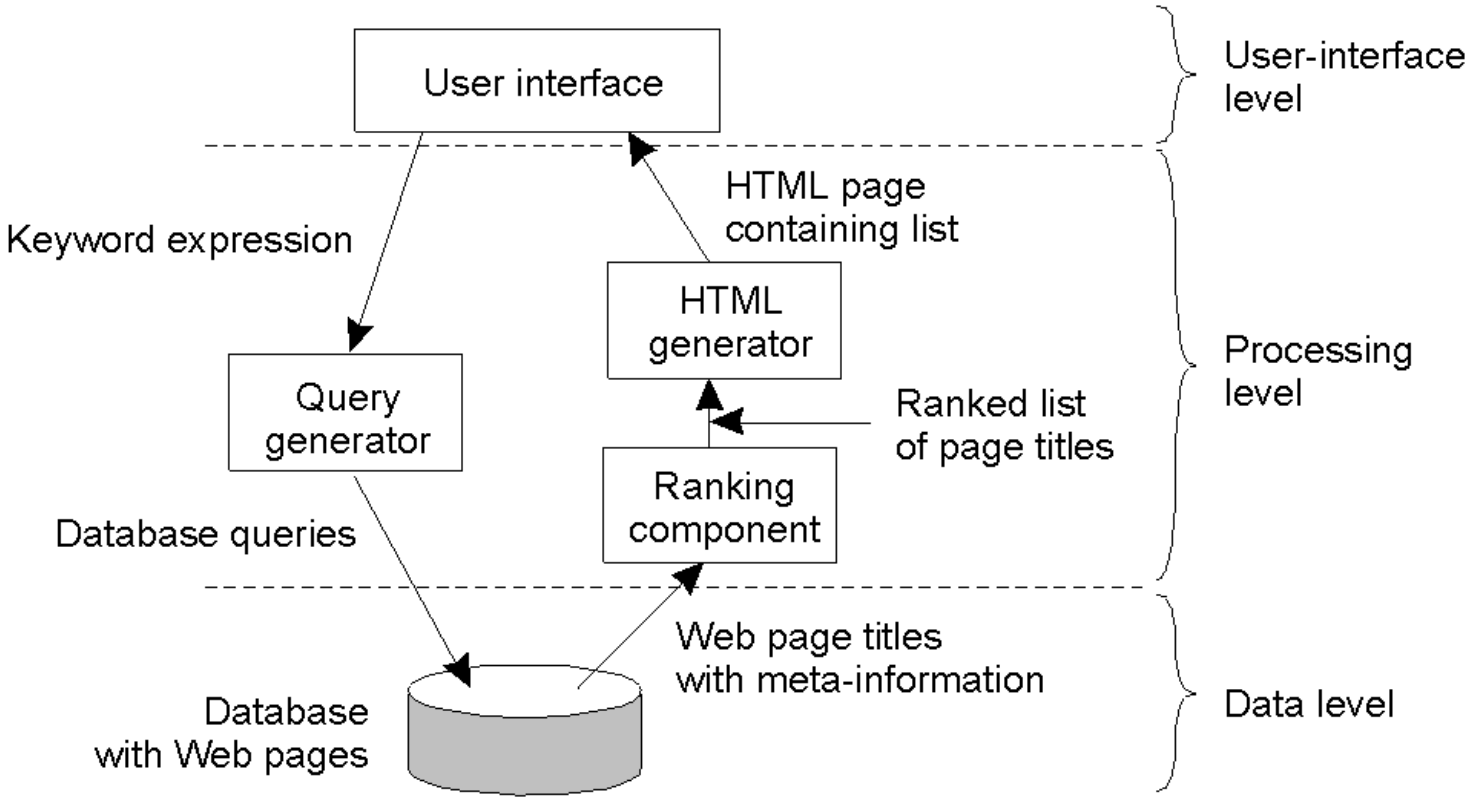


Architectural styles: Layered style



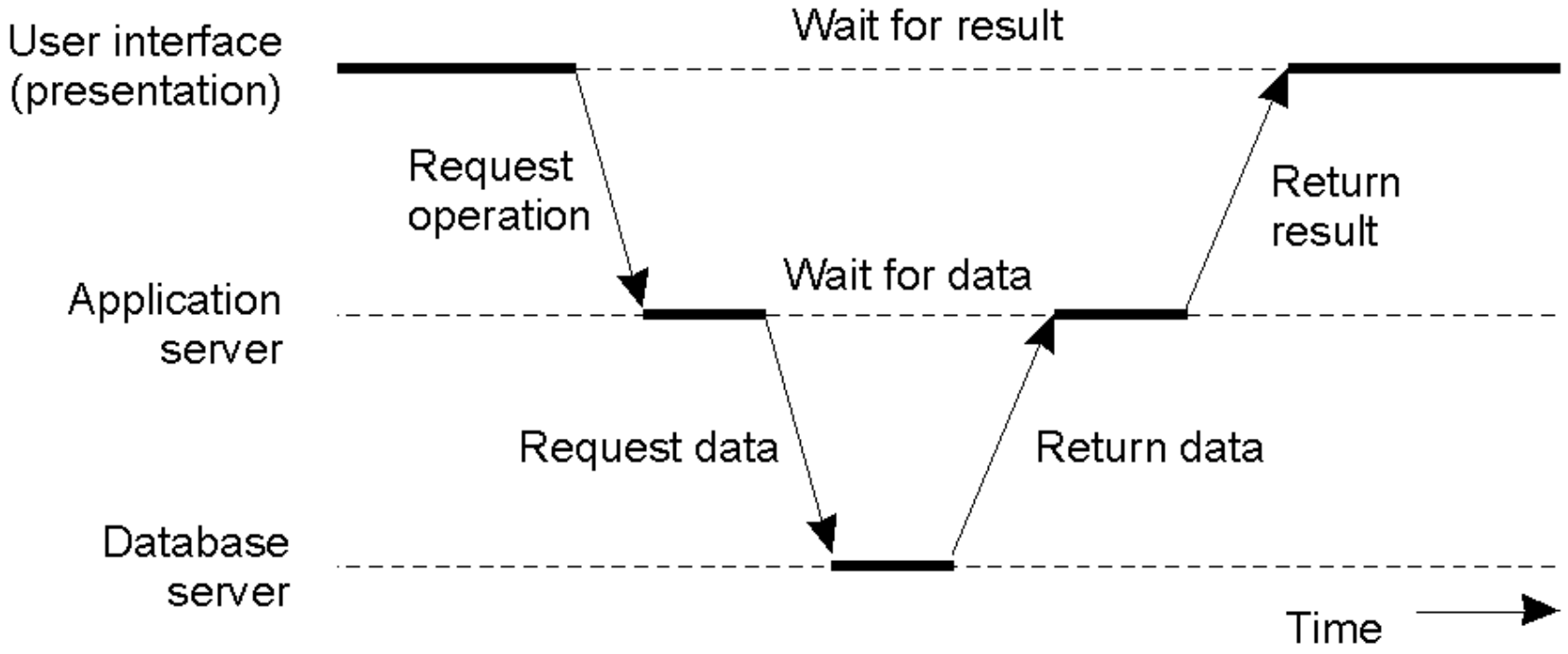
Architectural styles: Layered style example

- Layered style: three-layer (tier) architecture commonly used in many internet based applications today



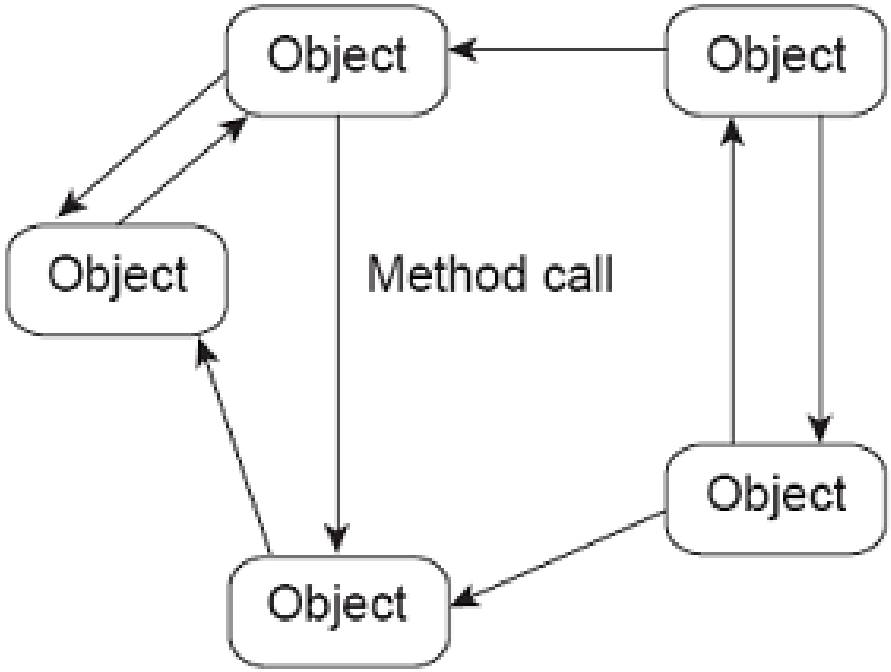
- General organization of an Internet search engine into three different layers

Architectural styles: Layered style example



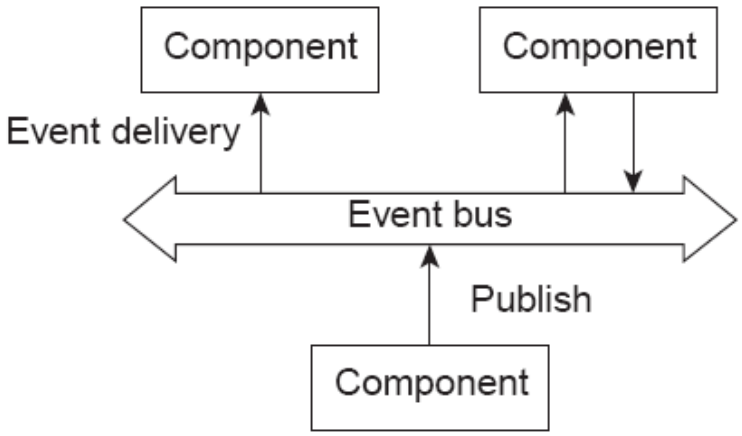
Architectural styles (II): object based

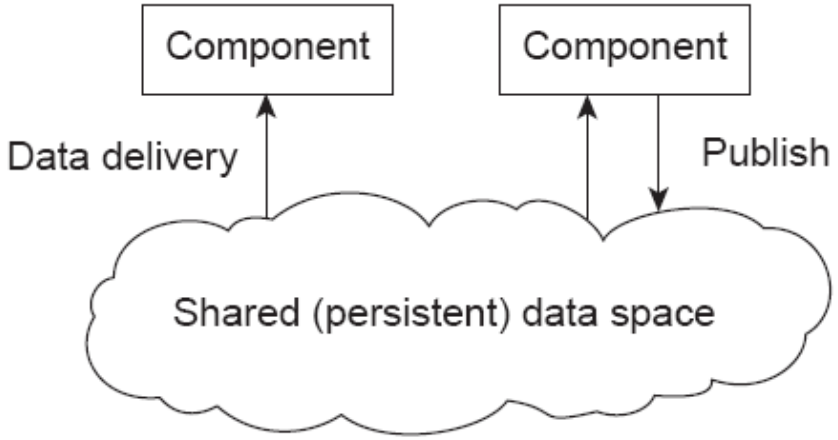
Idea**:** Organize into *logically different* components, and subsequently distribute those components over the various machines.



Architectural styles (III)

Alternative styles: Decouple processes in *space* (“anonymous”) and/or *time* (“asynchronous”)





Event-based

Data-centered architectures

Summary

- Introduction to Information Systems
  - Types of information systems
  - IS Pyramid
- Distributed Systems
  - Resource Accessibility
  - Distribution Transparency
  - Openness
  - Scalability
- Middleware
- Architectural styles for IS

Exam quizzes

- 1. Definiţi sistemele informatice. Care sunt tipurile de sisteme informatice din tehnologia informaţiei.
- 2. Daţi o definiţie cât mai cuprinzătoare pentru un sistem distribuit. Care sunt obiectivele urmărite în realizarea unui astfel de sistem.
- 3. Care sunt modelele arhitecturale importante pentru proiectarea sistemelor informatice. Daţi cate un exemplu de sistem informatic pentru fiecare caz în parte.
- 4. Descrieţi şi discutaţi noţiunea de *middleware*.