**Unit – I: Distributed System paradigms**

**Lecture: - 01 – Definition & Goals:- [1, 3]**

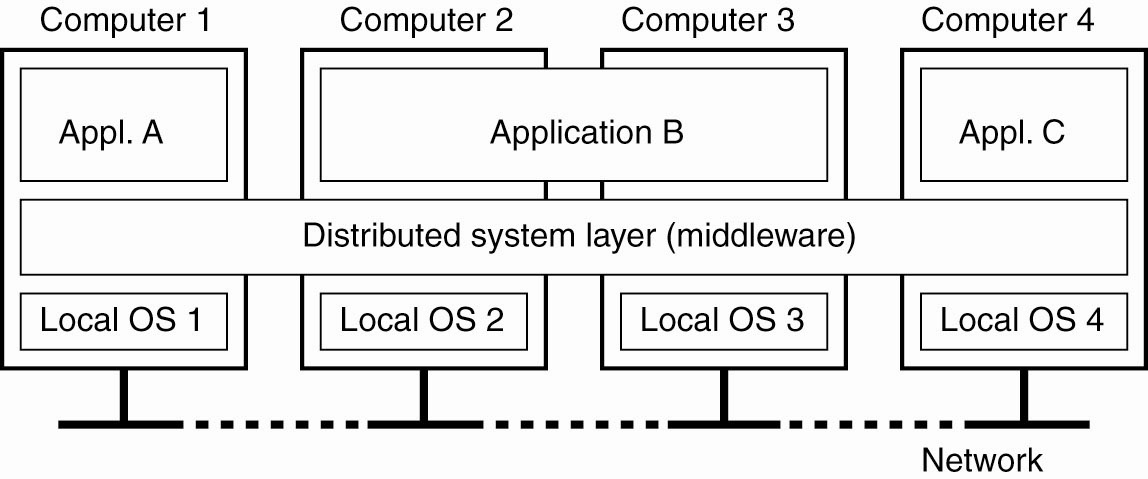
**Definition:- [1, 3]**

**Distributed System:**

* A collection of independent computers that appears to its users as a single coherent system.
* Features:
  + Message-based communication
  + Each runs its own local OS
* 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
* 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**. A distributed system organized as middleware

**Goals: [1, 3]**

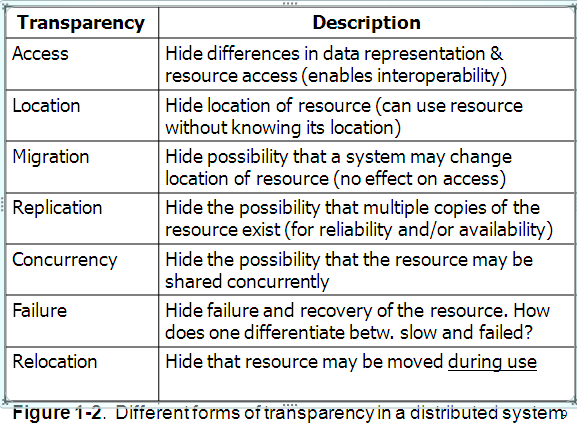
**Goal 1 – Resource Accessibility**

* 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
* Example: Sharing of a printer in small office
* 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:**



**Goal 3 – Openness:**

* An **open distributed system** “…offers services according to standard rules that describe the syntax and semantics of those services.”
* **Interface Definition/Description Languages (IDL):** used todescribe the interfaces between software components, usually in a distributed system
  + 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.
* **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.
* **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.
* 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

**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
* Centralized components + wide-area communication: waste of network bandwidth

**Lecture: - 02 –Goals:- [1, 3]**

**Scalability – Administrative:**

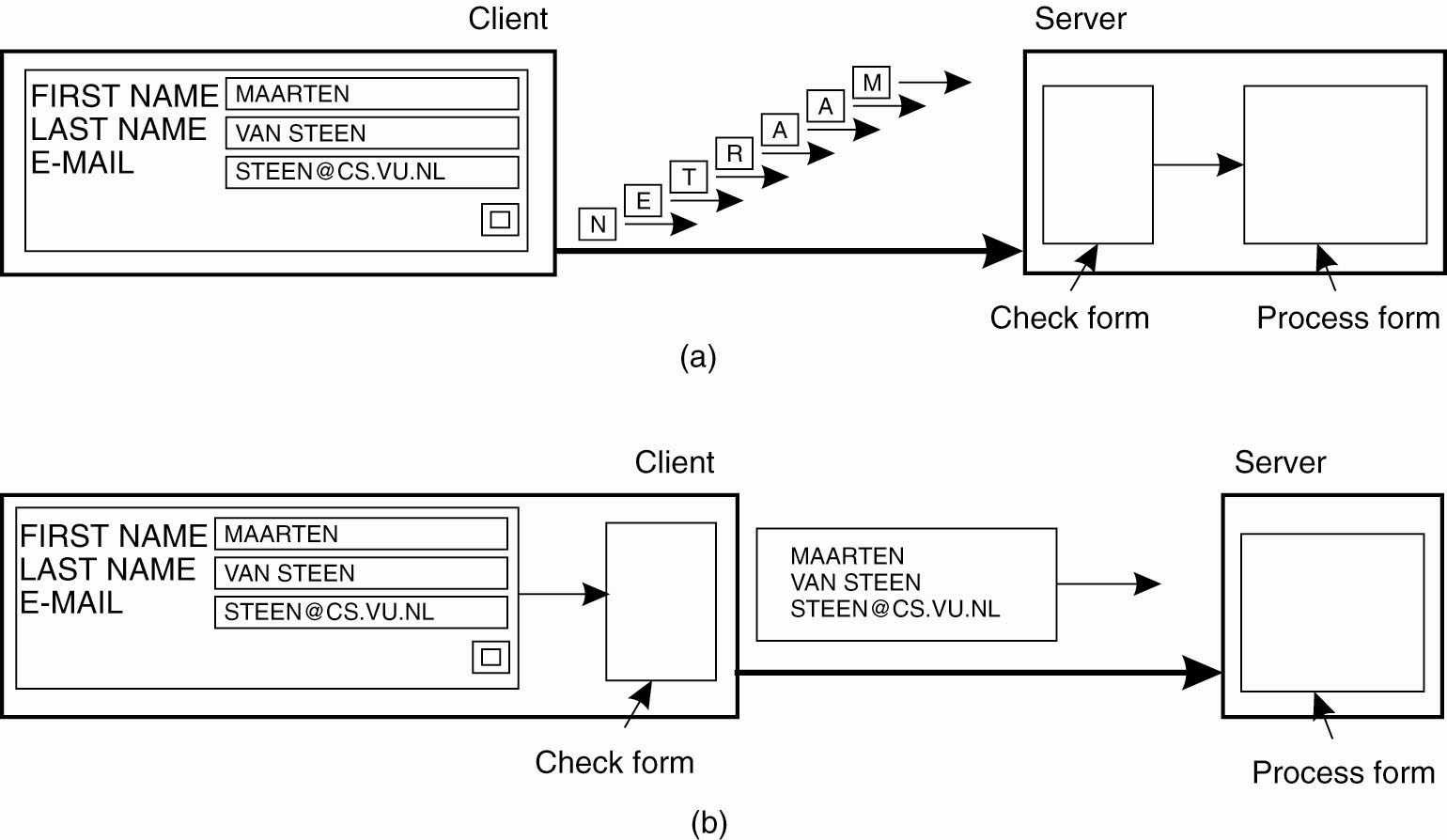
* Different domains may have different policies about resource usage, management, security, etc.
* Trust often stops at administrative boundaries
  + Requires protection from malicious attacks

**Scaling Techniques:**

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

**Hiding Communication Delays:**

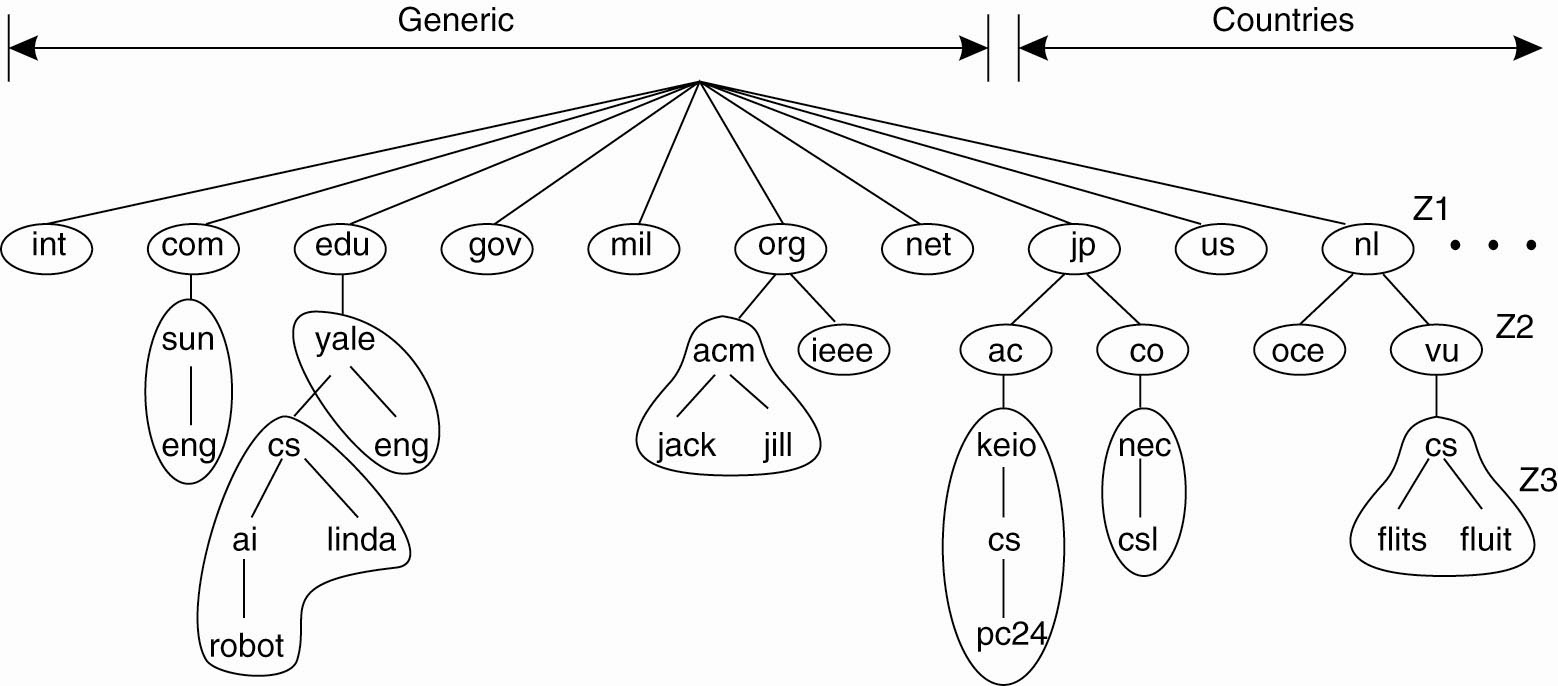
* Structure applications to use **asynchronous communication** (no blocking for replies)
  + While waiting for one answer, do something else; *e.g*., create one thread to wait for the reply and let other threads continue to process or schedule another task
  + Download part of the computation to the requesting platform to speed up processing



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

**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



**Figure.** An example of dividing the DNS name space into zones.

**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.

**Caching:**

* Caching is a form of replication
  + Normally creates a (temporary) replica of something closer to the user
* Replication is often more permanent
* User (client system) decides to cache, server system decides to replicate

Both lead to consistency problems

**Issues/Pitfalls of Distribution:**

Assumptions that everyone makes when developing distributed application for first time:

* 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.

**Text & Reference Books**

1. Distributed Systems Principles and Paradigms- A. S. Tanenbaum (2nd Edition), (Pearson Education)

2. Distributed Operating Systems - P. K. Sinha (PHI)

3. Distributed Systems – Concepts & Design by George Coulouris, Jean Dollimore, (Pearson Education)

**Lecture-: 03 - Types of Distributed Systems: [1, 3]**

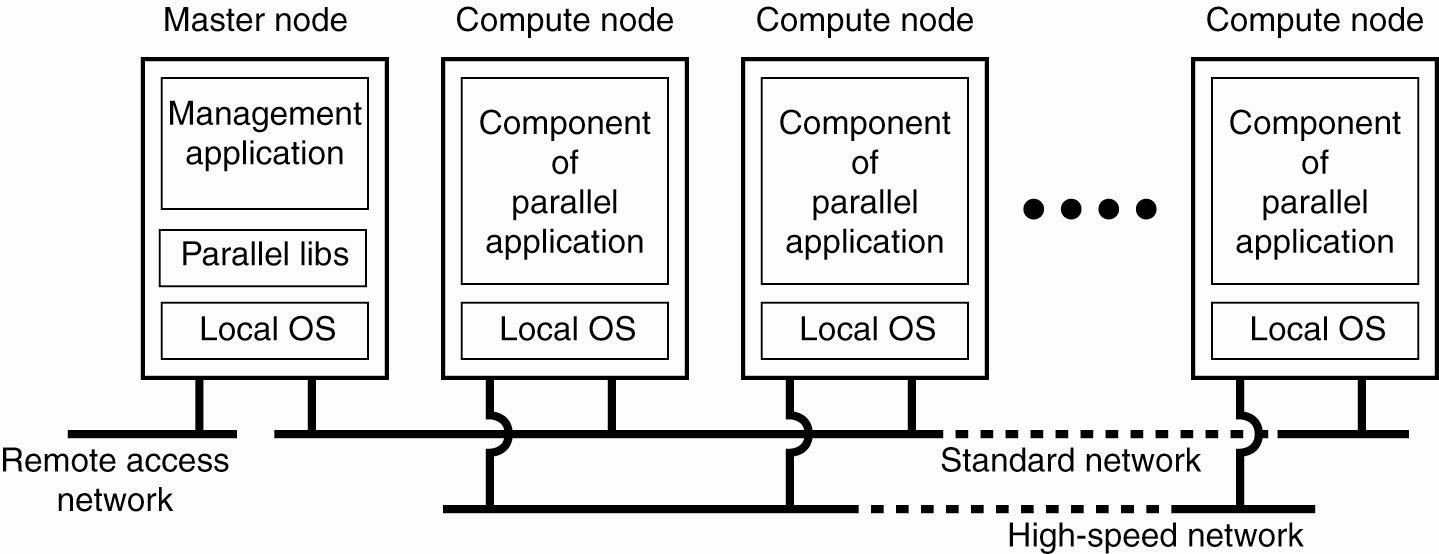
* **Distributed Computing Systems**
  + Clusters
  + Grids
* **Distributed Information Systems**
  + Transaction Processing Systems

Enterprise Application Integration

**Distributed Computing Systems [1, 3]**

**Cluster Computing:**

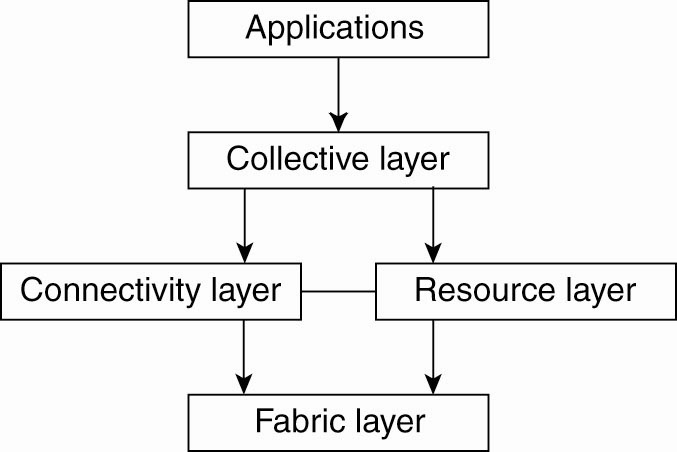
* A collection of similar processors (PCs, workstations) running the same operating system, connected by a high-speed LAN.
* Parallel computing capabilities using inexpensive PC hardware



**Figure**. An example of a cluster computing system

**Grid Computing Systems: [1, 3]**

* Highly heterogeneous with respect to hardware, software, networks, security policies, etc.
* Grids support virtual organizations: a team work of users who pool resources (servers, storage, databases) and share them.



**Figure.** A layered architecture for grid computing systems

**Fabric layer**: interfaces to local resources at a specific site

**Connectivity layer**: protocols to support usage of *multiple resources* for a single application; e.g., access a remote resource or transfer data between resources; and protocols to provide security

**Resource layer** manages a *single resource,* using functions supplied by the connectivity layer

**Collective layer:** resource discovery, allocation, scheduling, etc.

**Applications**: use the grid resources

The collective, connectivity and resource layers together form the middleware layer for a grid.

**Text & Reference Books**

1. Distributed Systems Principles and Paradigms- A. S. Tanenbaum (2nd Edition), (Pearson Education)

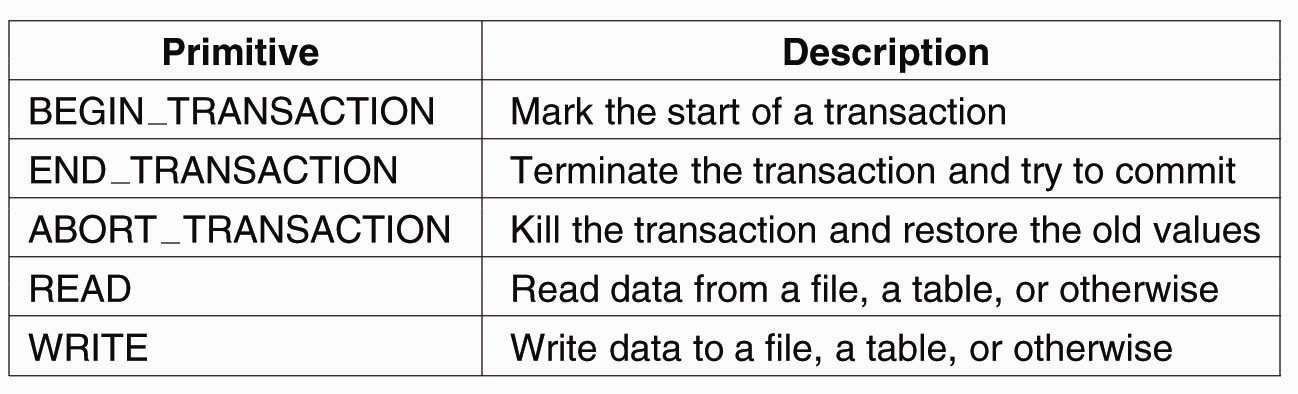
2. Distributed Operating Systems - P. K. Sinha (PHI)

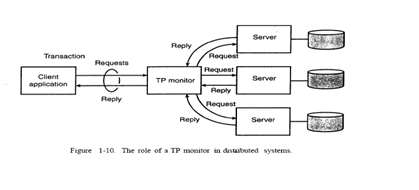
3. Distributed Systems – Concepts & Design by George Coulouris, Jean Dollimore, (Pearson Education)

**Lecture: - 03 - Distributed Information Systems: [1, 3]**

* Business-oriented.
* Systems to make a number of separate network applications interoperable.
* Two types discussed here:
  + Transaction processing systems
  + Enterprise application integration (EAI)

**Transaction Processing Systems: [1, 2, 3]**

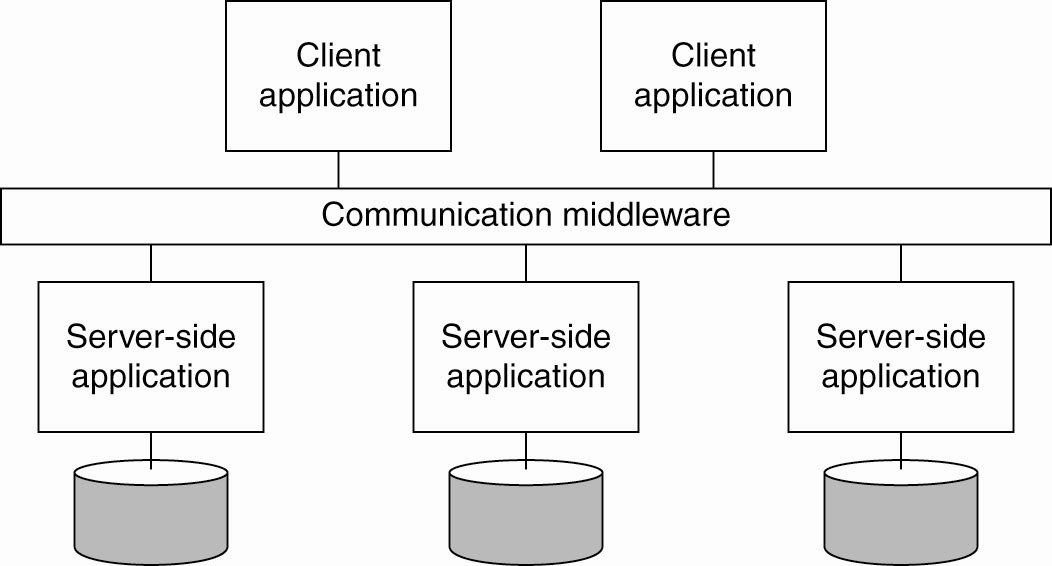
****



**Figure.** Example primitives for transactions

**Enterprise Application Integration: [1, 3]**

* Less structured than transaction-based systems
* EA components communicate directly
* May use different OSs, different DBs but need to interoperate sometimes.
* Communication mechanisms to support this include, Remote Procedure Call (RPC) and Remote Method Invocation (RMI)

****

**Figure:** Middleware as a communication facilitator in EAI.

**Text & Reference Books**

1. Distributed Systems Principles and Paradigms- A. S. Tanenbaum (2nd Edition), (Pearson Education)

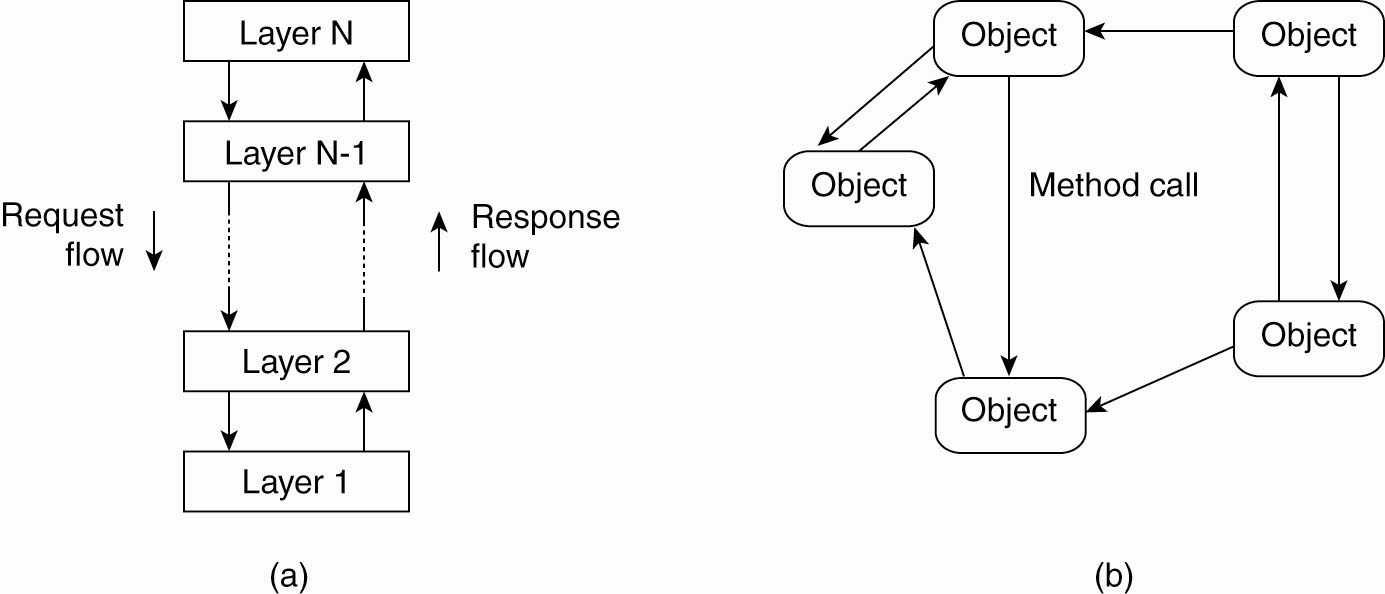
2. Distributed Operating Systems - P. K. Sinha (PHI)

3. Distributed Systems – Concepts & Design by George Coulouris, Jean Dollimore, (Pearson Education)

**Lecture: - 04 – Architectural Styles for Distributed Systems: [1, 3]**

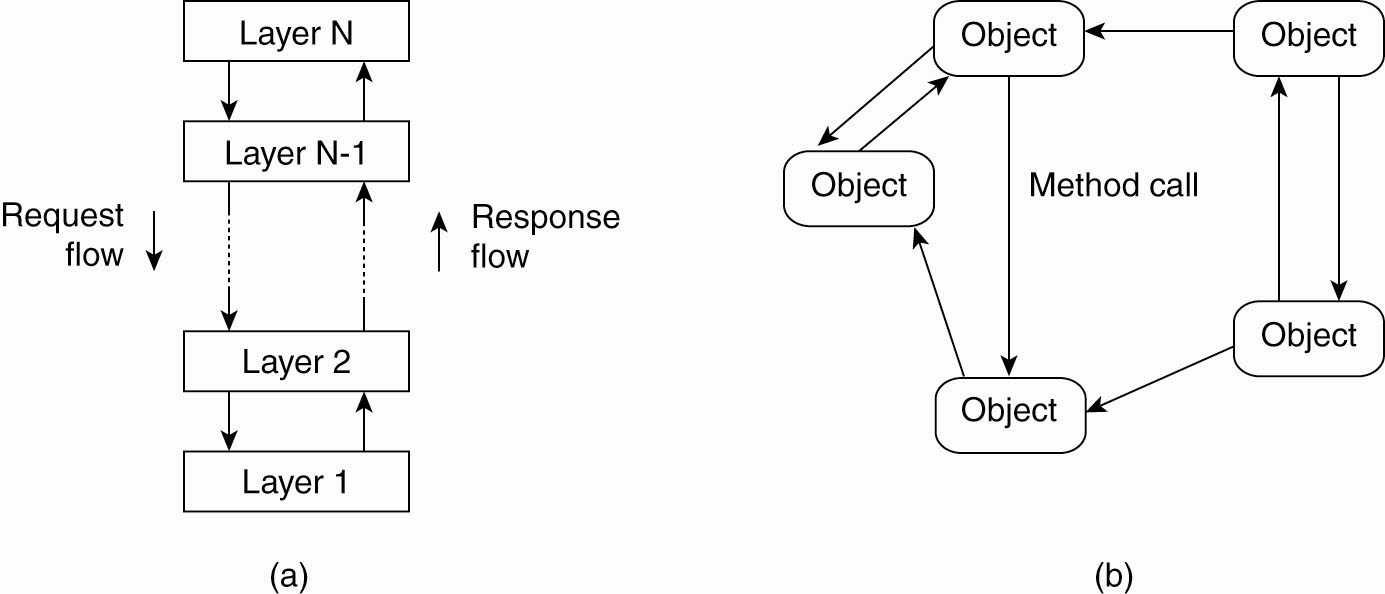
* An architectural style describes a particular way to configure a collection of components and connectors.
  + Component - a module with well-defined interfaces; reusable, replaceable.
  + Connector – communication link between modules.
* Architectures suitable for distributed systems:
  + Layered architectures
  + Object-based architectures
  + Data-centered architectures
  + Event-based architectures

**Layered architectural style [1, 2, 3]**

****

**Figure (a)** Layered architectural style

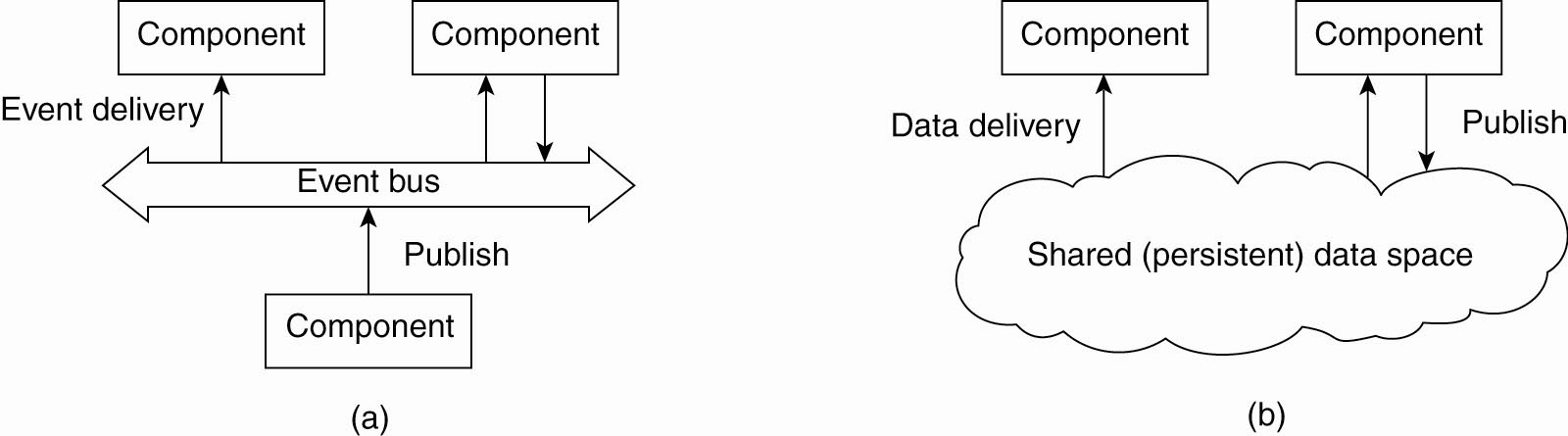
**The object-based architectural style. [1, 3]**



**Figure (b)** The object-based architectural style.

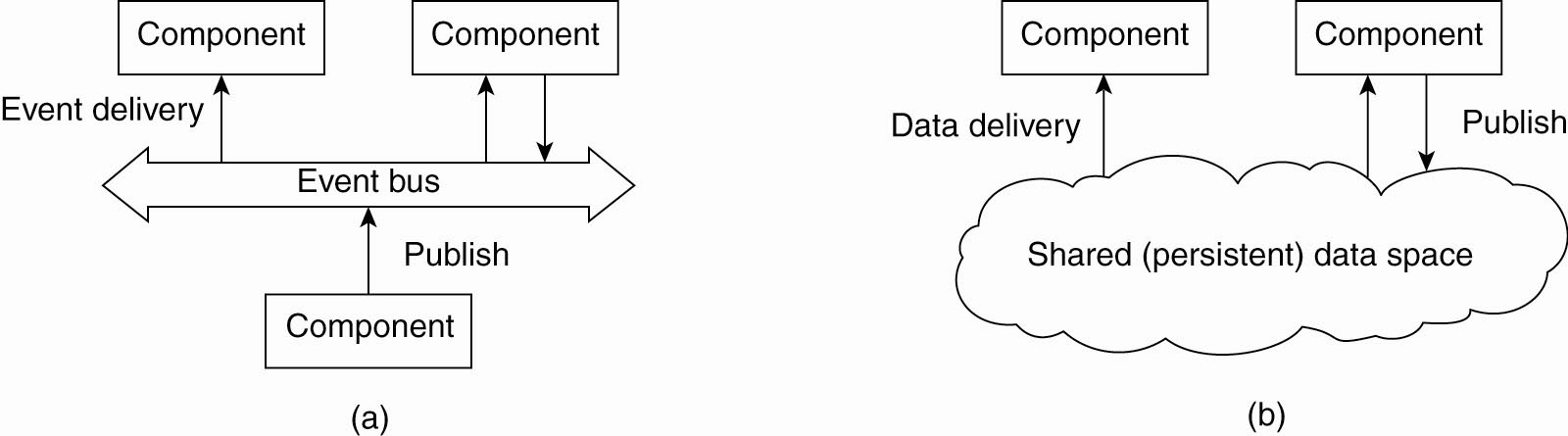
**Data-Centered Architectures: [1, 2, 3]**

* Main purpose: data access and update
* Processes interact by reading and modifying data in some shared repository (active or passive)
* Another example: web-based distributed systems where communication is through web services



**Figure. (a)** The event-based architectural style

**Shared data-space architectural style: [1, 3]**

****

**Figure. (b)** The shared data-space architectural style

**Text & Reference Books**

1. Distributed Systems Principles and Paradigms- A. S. Tanenbaum (2nd Edition), (Pearson Education)

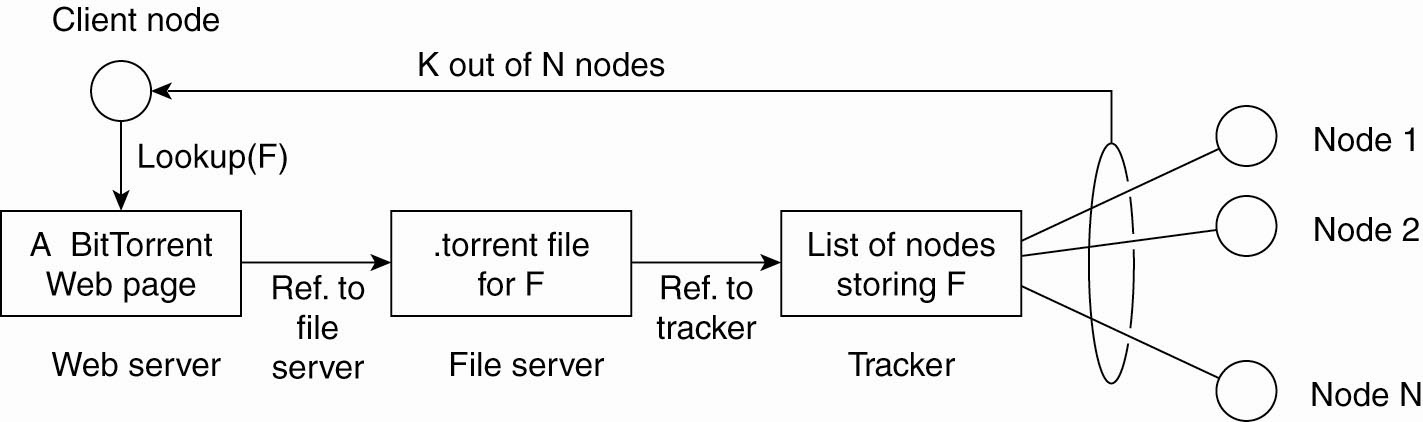
2. Distributed Operating Systems - P. K. Sinha (PHI)

3. Distributed Systems – Concepts & Design by George Coulouris, Jean Dollimore, (Pearson Education)

**Lecture: - 05 - System Architectures for Distributed Systems: [1, 3]**

* **Centralized:** traditional client-server structure
  + Vertical (or hierarchical) organization of communication and control paths (as in layered software architectures)
  + Logical separation of functions into client (requesting process) and server (responder)
* **Decentralized:** peer-to-peer
  + Horizontal rather than hierarchical comm. and control
  + Communication paths are less structured; symmetric functionality
* **Hybrid:** combine elements of C/S and P2P
  + Edge-server systems
  + Collaborative distributed systems.

[**BitTorrent**](http://www.bittorrent.org/)**:**



**Figure**. The principal working of BitTorrent.

* Clients contact a global directory (Web server) to locate a *.torrent* file with the information needed to locate a **tracker**; a server that can supply a list of active nodes that have chunks of the desired file.
* Using information from the tracker, clients can download the file in chunks from multiple sites in the network. Clients must also provide file chunks to other users.
* Designed to force users of file-sharing systems to participate in sharing.
* When a user downloads your file, he becomes in turn a server who can upload the file to other requesters.
* Share the load.

**Text & Reference Books**

1. Distributed Systems Principles and Paradigms- A. S. Tanenbaum (2nd Edition), (Pearson Education)

2. Distributed Operating Systems - P. K. Sinha (PHI)

3. Distributed Systems – Concepts & Design by George Coulouris, Jean Dollimore, (Pearson Education)