Distributed System

Introduction

There has been a great revolution in computer systems. In the initial days, computer systems were huge and also very expensive. Because of this reason few firms had less number of computers and those systems were operated independently as there was a lack of knowledge to connect them. The development in technology changed such situations and in these developments, the initial development was the advancement of microprocessors.

The next advancement was the invention of computer networks which had high speed like the local area networks. The output of these applied sciences made easy to connect many computers to a network which has high speed. These connected systems are called as distributed systems or canned computer networks.

Definition

A distributed system is defined as a group of independent computers which looks to its users as a single system which is coherent.

The above-explained definition has many vital aspects and two vital aspects of them are as follows:

- The initial aspect is that the distributed system has components which are autonomous and here the components are nothing but the computer systems.
- The next aspect is that the users of it think that they are managing with a single system.

It means that in a way or other, the autonomous computers need to collaborate. The process of collaborating lies in the developing of distributed systems.

Goals of a Distributed System

It is not that distributed systems are built just because people have the possibility of building it. Every distributed system has few goals which have to be achieved during its building process and few vital goals of a distributed system are as follows:

- Making Resources Accessible
- Distribution transparency
- Openness
- Scalability
- Pitfall

1. **Making Resources Accessible:** Few typical examples of resources are as below:
    - Networks
    - Web pages
    - Files
    - Data
    - Storage facilities
    - Computers
    - Printers
2. **Transparency:** The concept of transparency is applicable to many aspects of a distributed system and the vital types of transparencies are as follows:
    - Failure
    - Concurrency
    - Access
    - Location
    - Migration
    - Replication
    - Relocation
3. **Openness:** The aim of openness has the following activity:
    - Separating policy from mechanism
4. **Scalability:** The scalability includes the below activities:
    - Scalability problems
    - Scaling techniques
5. **Pitfalls:** The pitfalls explain about the mistakes which are made during the advancement of distributed system for the foremost time. These mistakes are formulated as false assumptions and they are as follows:
    - It has the reliable network.
    - The network is fully secured.

- The network is homogeneous.
- It has the zero latency.
- Only one administrator is present.
- The cost of transport is zero.
- It has infinite bandwidth.
- The topology does not change.

Types of Distributed System

- **Distributed Computing Systems**

Used for high performance computing

Cluster Computing Systems

mainly for parallel programming
homogeneous platforms

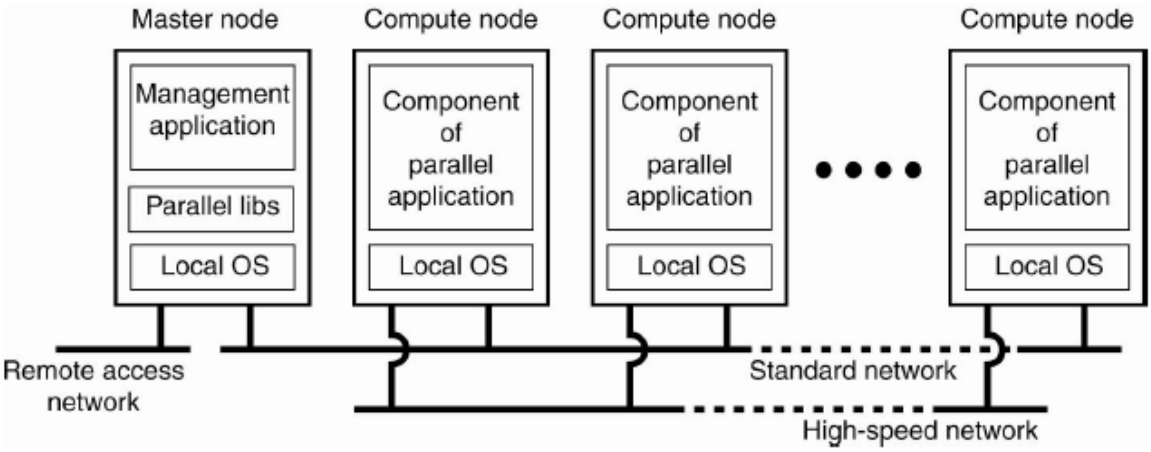

Figure 6

e.g Beowulf -- Linux-based cluster

Grid Computing Systems

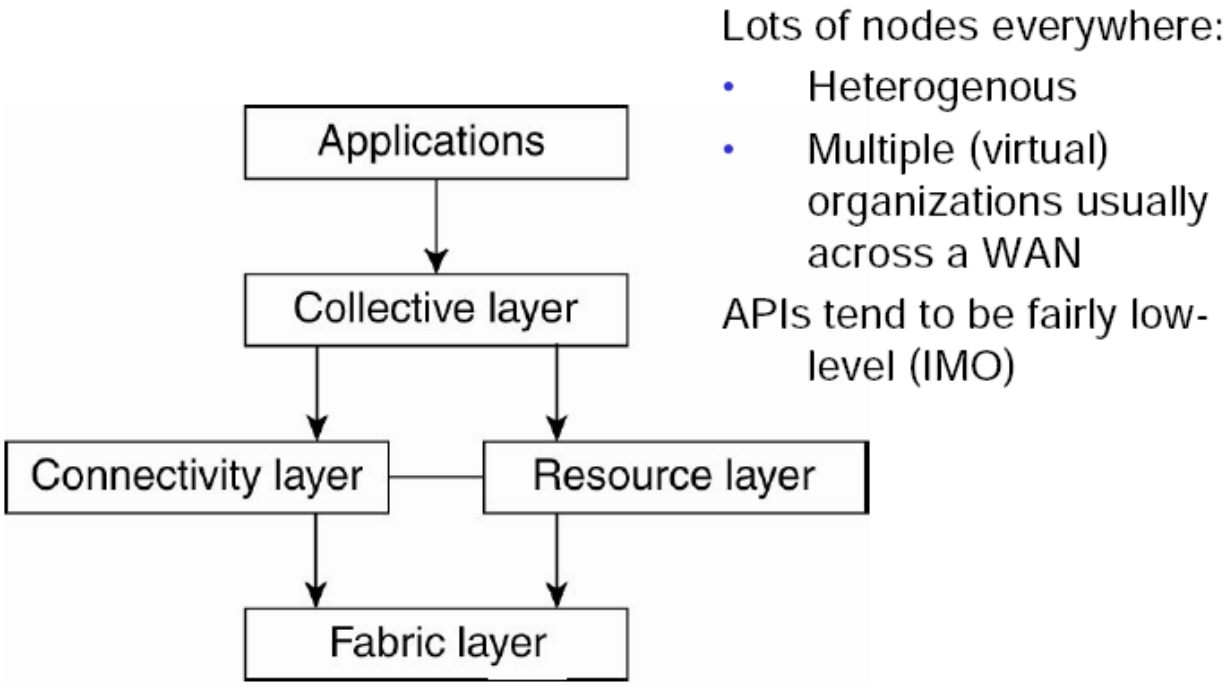

Figure 7

- fabric -- provides interfaces to local resources at a specific site
- connectivity -- consists of communication protocols for supporting grid transactions that may use multiple resources
  also authenticate users to use certain resources
- resource -- responsible for managing a single resource; use info from connectivity layer to call fabric layer
- collective -- handles access to multiple resources, typically consisting of services for discovering and allocating resources

e.g. SETI (Search for Extraterrestrial Intelligence)@Home project: thousands of people are sharing the unused processor cycles of their PCs in the vast search for signs of "rational" signals from outer space.

- **Distributed Information Systems**

Transaction Processing Systems

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 8 Example primitives for transactions.

```
BEGIN_TRANSACTION(server, transaction);
READ(transaction, file-1, data);
WRITE(transaction, file-2, data);
newData := MODIFIED(data);
IF WRONG(newData) THEN
ABORT_TRANSACTION(transaction);
ELSE
WRITE(transaction, file-2, newData);
END IF;
END_TRANSACTION(transaction);
```

Characteristic properties of transactions:

- **Atomic**: To the outside world, 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.

A nested transaction can be constructed from a number of subtransactions.

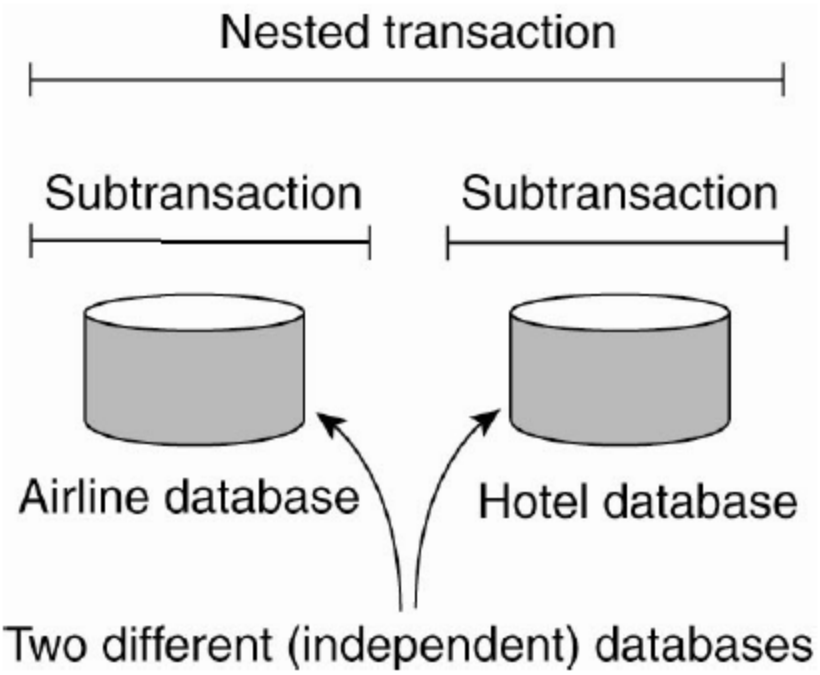

Figure 9. A nested transaction.

'durable' applies to the top level only

Transaction processing (TP) monitor allows an application to access multiple server/database.

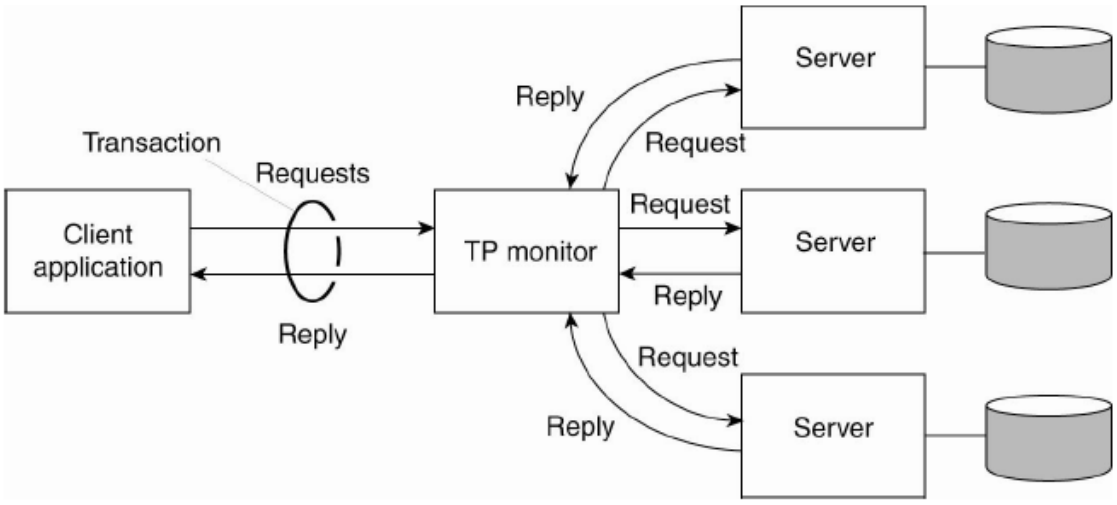

Figure 10. The role of a TP monitor in distributed systems.

Enterprise Application Integration

The more applications became decoupled from the databases they were built upon, the more we needed facilities to integrate applications independent from their databases; we want application components to communicate directly with each other.

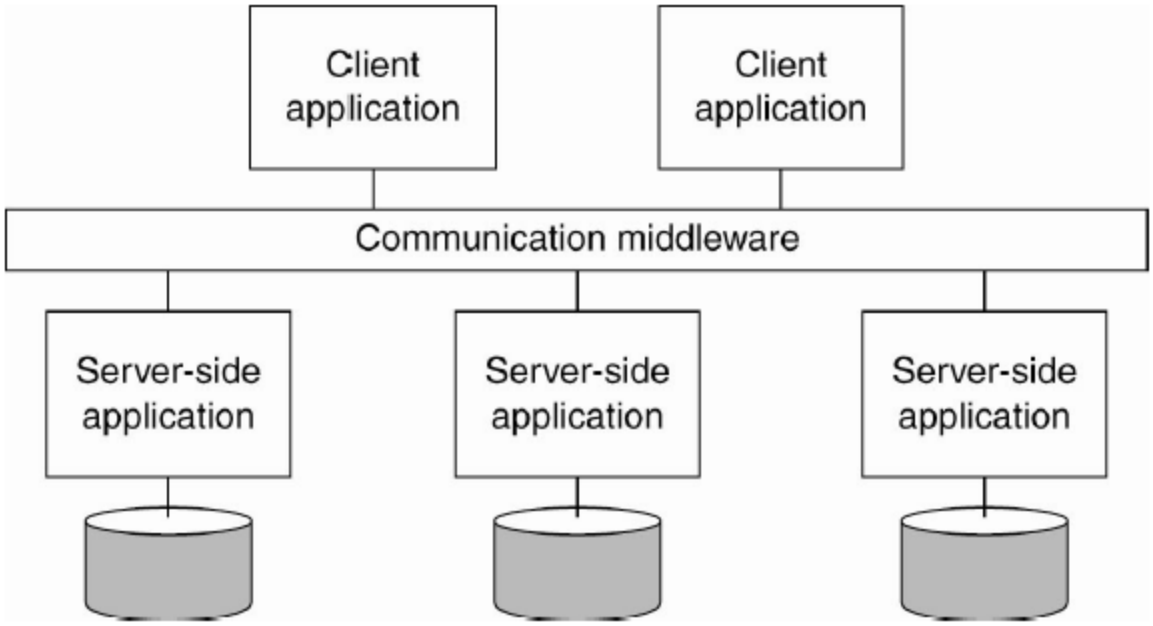

Figure 11. Middleware as a communication facilitator in enterprise application integration

RPC and RMI (remote method invocation) are examples of middleware; RMI operates on objects.
Both RPC and RMI require caller and callee be up and running during communication.

Message-oriented middleware (MOM) lets applications send messages to logical contact points. (**publish/subscribe**)

- ## Distributed Pervasive Systems

Small nodes, mobile, usually embedded, battery-powered
Requirements for pervasive systems

- Embrace contextual changes.
- Encourage ad hoc composition.
- Recognize sharing as the default.

Electronic Health Care Systems
Great example of pervasive system
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?

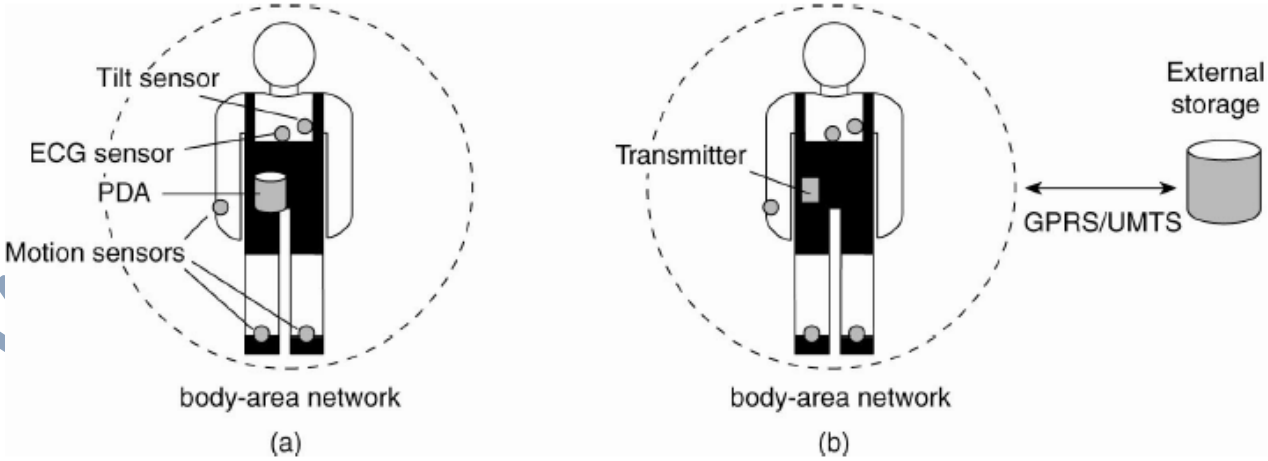

Figure 12.
Monitoring a person in a pervasive electronic (ECG (Electrocardiogram) sensors) health care system, using

(a) a local hub or
(b) a continuous wireless connection

Sensor Networks
Characteristics: (10s to 1000s, simple CPUs, often battery)
Used for processing information; mostly use wireless communicaions

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?

Consider sensor networks from a database perspective/API!

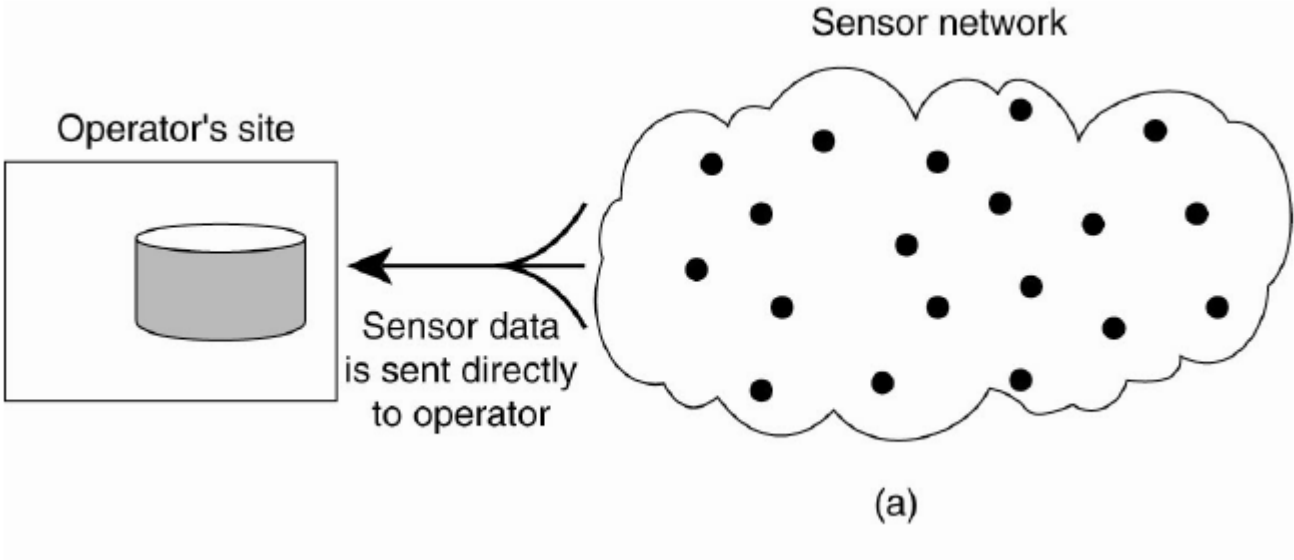

Figure 13a. Organizing a sensor network database, while storing and processing data (a) only at the operator's site or ...

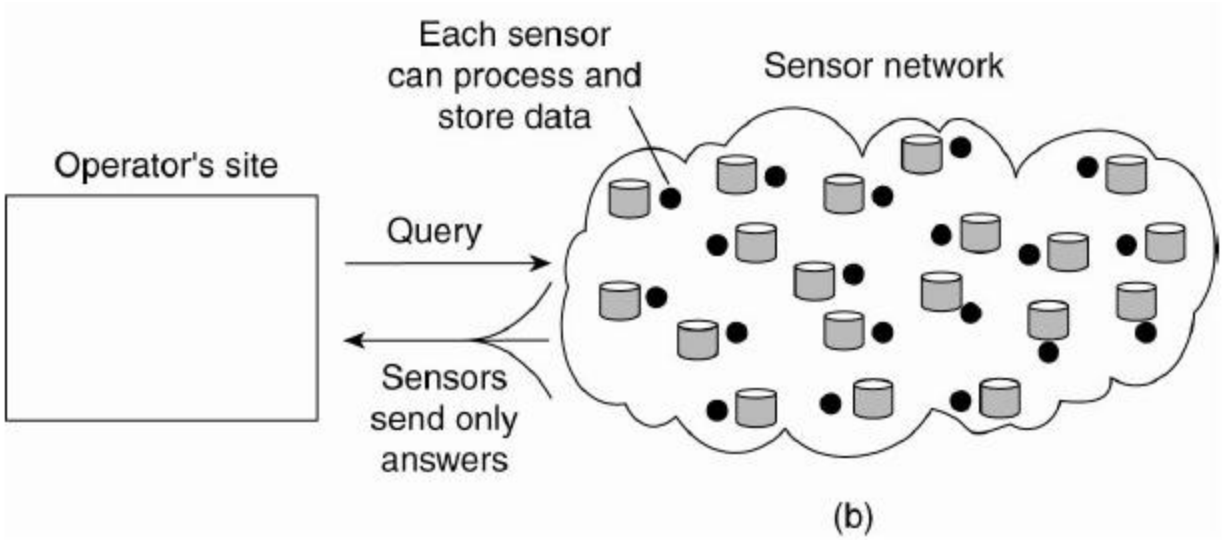

Figure 13b. Organizing a sensor network database, while storing and processing data ... or (b) only at the sensors.

Architectural Styles

The vital architectural styles of distributed systems are as follows:

- Layered architecture
- Object-based architecture
- Event-based architecture
- Data-centered architecture

The diagrammatic representation of a layered architecture if as below:

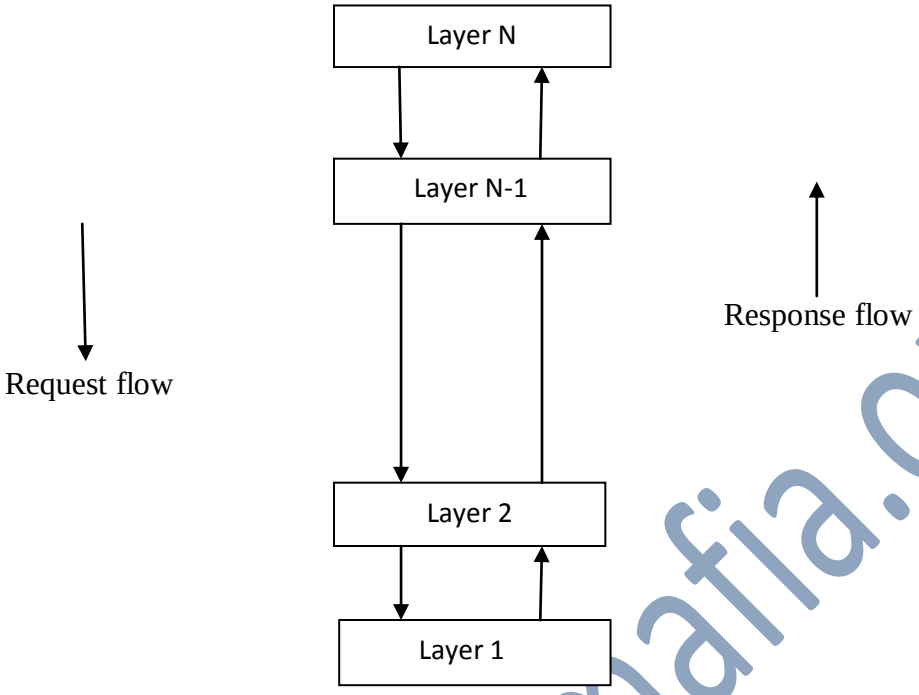

Fig1: layered architectural style

The below diagram explains about the object-based architectural style:

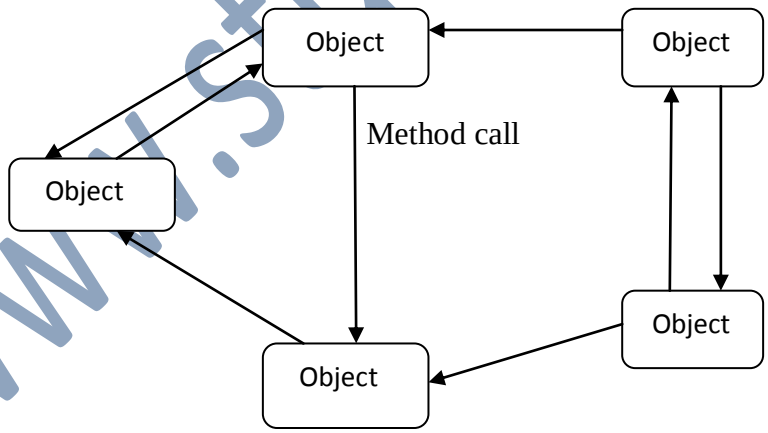

Fig2: object-based architectural style

The diagrammatic representation of event-based architecture is as below:

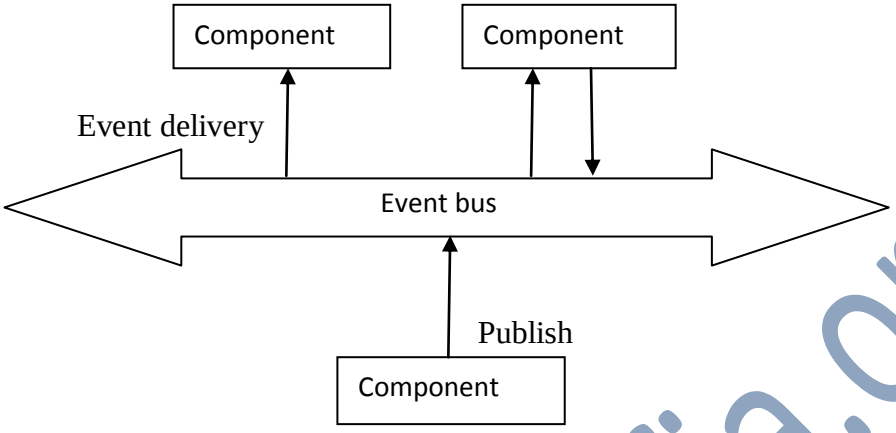

Fig3: event-based architecture

The diagrammatic representation of shared data based architecture is as follows:

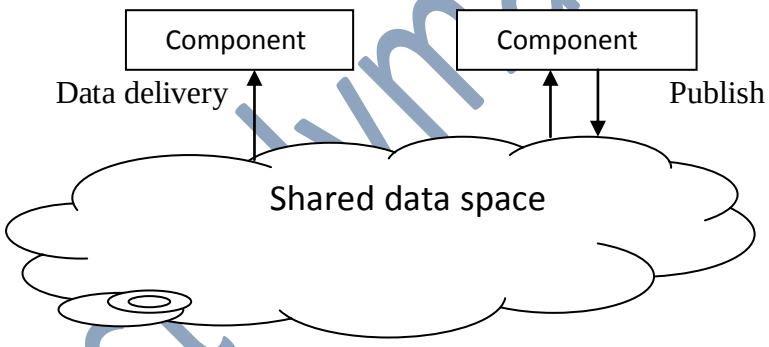

Fig4: shared data based architecture

Characteristics of Distributed System

- Data set can be split in to fragments and can be distributed across different nodes within network.
- Individual data fragments can be replicated and allocated across different nodes.
- Data at each site is under control of a DBMS.
- DBMS at each site can handle local applications autonomously.
- Each DBMS site will participate in at least one global application.

Differences between shared nothing Parallel System and Distributed system are :

1. In distributed system, databases are geographically separated, they are administered separately and have slower interconnection.
2. In distributed systems, we differentiate between local and global transactions. Local transaction is one that accesses data in the single site at that the transaction was initiated. Global transaction is one which either accesses data in a site different from the one at which the transaction was initiated or accesses data in several different sites.

Advantages and Disadvantages of Distributed System

Advantages of Distributed System

- **Sharing Data:** There is a provision in the environment where user at one site may be able to access the data residing at other sites.
- **Autonomy:** Because of sharing data by means of data distribution each site is able to retain a degree of control over data that are stored locally.
- In distributed system there is a global database administrator responsible for the entire system. A part of global data base administrator responsibilities is delegated to local data base administrator for each site. Depending upon the design of distributed database
- each local database administrator may have different degree of local autonomy.
- **Availability:** If one site fails in a distributed system, the remaining sites may be able to continue operating. Thus a failure of a site doesn't necessarily imply the shutdown of the System.

Disadvantages of Distributed Systems

The added complexity required to ensure proper co-ordination among the sites, is the major disadvantage. This increased complexity takes various forms :

- **Software Development Cost**: It is more difficult to implement a distributed database system; thus it is more costly.
- **Greater Potential for Bugs**: Since the sites that constitute the distributed database system operate parallel, it is harder to ensure the correctness of algorithms, especially operation during failures of part of the system, and recovery from failures. The potential exists for extremely subtle bugs.
- **Increased Processing Overhead**: The exchange of information and additional computation required to achieve intersite co-ordination are a form of overhead that does not arise in centralized system.

REFERENCES

1. www.google.com
2. www.wikipedia.org
3. www.studymafia.org