# Distributed Systems
# MTech, 2013
# Delhi Technological University
# Introduction
# Instructor: Divyashikha Sethia

# divyashikha@dce.edu
# Sites.google.com/site/divyashikhasethia




# Course Objective

Understanding of the issues involved in distributed computer systems and to investigate the fundamental characteristics of distributed computing systems, including their models, architectures and designs that exploit rapidly evolving technology.

# Course Introduction

**Prerequisites:**
- Operating System
- Computer Networks



# Course Outline

- **Unit1:**
  - **Introduction:** Definitions and motivation, characteristics, issues in design of DS
  - **Architecture:** Distributed System Models

- **Unit2: Inter-process communication:** Networking concepts, Sockets and streams, remote procedure calls, remote method invocation

- **Unit3: Time Synchronization:** Logical clocks, vector clocks, direct dependency clocks, matrix clocks

- **Unit 4: Resource Allocation Synchronization:** Distributed Shared Memory, Process Scheduling, Load Balancing & Load Sharing, Mutual Exclusion, Election algorithms.

# Course Outline

**Unit 5: Replication and Consistency**

Need for replication, consistency models and protocols

**Unit 6: Fault Tolerance**

–Basic concepts, reliability and availability, recovery

**Unit7: File Systems**

–Distributed file services, example file systems



# Text

**Text Books:**

Distributed Systems: Principles and Paradigms, 2nd Ed., Andrew S. Taenbaum and Maarten Van Steen, Prentice Hall, 2007.

**References:**

• Distributed Operating Systems: Concepts and Design, P.K.Sihna, PHI, 2007.
•Distributed Operating Systems and Algorithms. R. Chow, T. Johnson. Addison-Wesley Publishing Company, 1997. ISBN 0-201-49838-3.
•Distributed Systems: Concepts and Design, 4th Ed by Coulouris, G, Dollimore, J., and Kindberg, T., Addison-Wesley, 2006.

# Evaluation

**MTech:**

- Mid Semester: 30
- Internal (Presentation/Programming): 20
- End Semester : 100

**BTech:**

- Mid Semester: 20
- Internal (Quiz/Tests): 10
- End Semester: 70



# Definition of a Distributed System (1)

- A distributed system is:

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

- Utilizes the power of :
  - Advancement in computing power of microprocessor
  - High speed network communication between computers

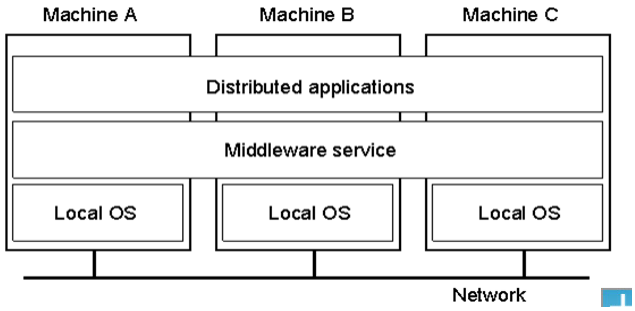
# Some uses..

- Pool of processors dynamically assigned to users to execute a job in the best manner possible

- Large bank with branch offices over world, each office with:
  - Master computer to store local accounts and handle local transactions
  - Ability to communicate with other branch offices and central computer
  - Transactions can be made without regard to where customer account is located.



# Definition of a Distributed System (2)




- Middleware to support heterogeneous computers and networks and offering a single system overview.
- Middleware layer extends over multiple machines.
- Applications distributed over computers.

# Goals

- Resource Availability
- Distribution Transparency
- Openness
- Scalability



# Resource Availability

- Access to Remote resources by all users eg: printers, computers, storage
- Helps collaborate and exchange information
  - exchanging files, mail. documents, audio, and video
  - electronic commerce
- Threat to Security

# Transparency in a Distributed System

| 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 |

Different forms of transparency in a distributed system.



# Openness

- Offers services according to standard rules in the form of interfaces described by Interface Definition Languages (IDL)
- Allows an arbitrary process to talk to another process
- Extensible: add new components or replace old ones

# Scalabity

- Add more users and resources
- Geographical scalability wrt to distribution of users resources across different places
- Administrative scalability



# Scalability Problems when users increase

| 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 |

Examples of scalability limitations.

# Centralized algorithm

- Collect information about load on machines and lines and compute optimal routes for communication.
- Collecting and Transporting information to centralized server and then distributing information after analysis overloads network.
- Distributed algorithms characteristics:
  - No machine with complete information about system state
  - Machine decision based on local information only
  - Algorithm not ruined by failure of a single machine
  - Global clock



# Geographic scalability

- Client server architecture which requires synchronous communication works well on a LAN.
  - client, blocks until a reply is sent back
- Communication delays in WAN for response can be longer
- Unreliability of communication over WAN
  - local-area networks generally provide highly reliable communication facilities based on broadcasting

# Administrative scalability

- Scaling across multiple independent administrative domains resolving conflicting policies for resource usage, management, security
- Security issues across domains



# Scaling Techniques

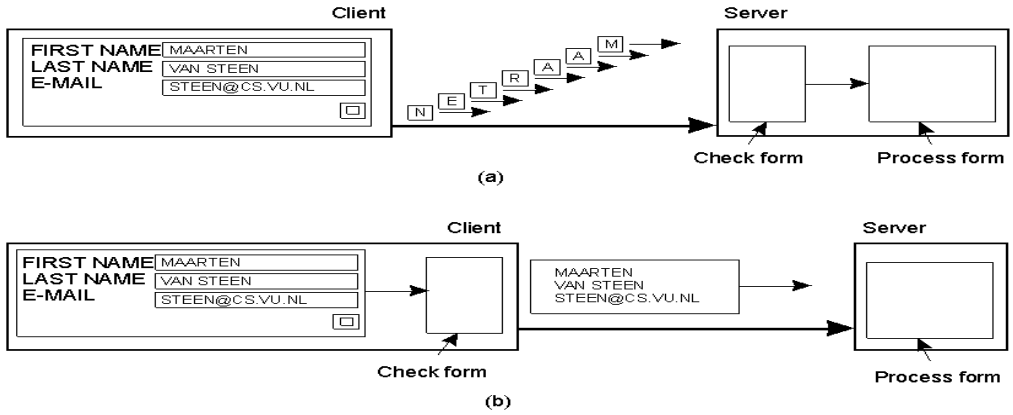
- Hiding Communication latencies
- Distribution
- Replication

# Scaling Techniques – Hiding Communication latencies

- Avoid waiting for responses to remote service requests (asynchronous communication)
- Reply invokes interrupt to complete the previous request
- Alternative to start a new thread of control to perform request – only single thread blocked for reply while others are in process
- Not suitable for interactive applications requiring quick response time.
  eg: accessing databases using forms
- Solution: interactive form filling to be done at client side rather than server side



# Hiding Communication latencies(2)




The difference between letting:
a) a server or b) a client check forms as they are being filled

# Scaling Techniques – Hiding Communication latencies (3)

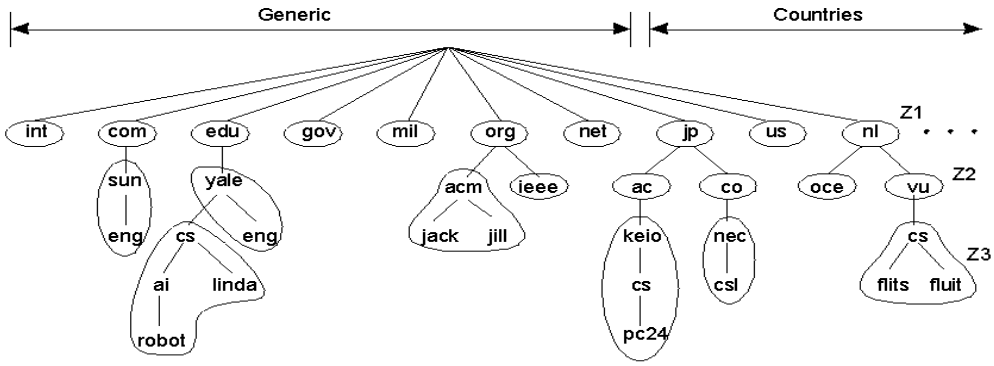
- Server may check for syntactic errors before accepting an entry
- Better solution: code for filling in form, and possibly checking entries, to the client, and have the client return a completed form
Eg: Web in the form of Java applets and Javascript



# Scaling Techniques –Distribution (1)

- Splitting components into smaller parts and spreading them across system
- Eg: Domain Name System name space is hierarchically organized as tree of domains, which are divided into nonoverlapping zones,

# Scaling Techniques –Distribution (2)




An example of dividing the DNS name space into zones and distributing the naming service

Resolving "nl. vu.cs.flits": Traverse Z3,Z2,Z1



# Scaling Techniques –Distribution (3)

**World Wide Web:**

- Web is physically distributed across a large number of servers, each handling a number of Web documents.
- The name of the server handling a document is encoded into that document's URL.
- It is only because of this distribution of documents that the Web has been capable of scaling to its current size

# Scaling Techniques - Replication

- Replicate components across distributed system – increasing availability and balancing load between components for better performance
  eg: Geographically close copy can hide communication latency

- Caching is a form of replication – copy of resource in the proximity of the client accessing the resource
  - Difference from replication: on-demand rather than planned

- Leads to consistency problems
  - update must be immediately be propagated to all other copies
  - Two concurrent updates must be made in the same order.

- Requires global synchronization



# Scalability Conclusion

- Size scalability is the least problematic

- Geographical scalability is a much tougher problem

- Administrative scalability seems to be the most difficult one, because need to solve nontechnical problems
  - peer-to-peer technology demonstrates what can be achieved if end users simply take over control

# Advantages over Centralized systems

- Resource utilization provides better performance for the price.
- Provide more computing power compared to centralized mainframe
- Computer supported cooperative work involving spatially separated machines
- Higher reliability
- Incremental growth in computing power



# Advantages over PC

| | |
|---|---|
| Data sharing | Allow many users access to a common data base |
| Device sharing | Allow many users to share expensive peripherals like color printers |
| Communication | Make human-to-human communication easier, for example, by electronic mail |
| Flexibility | Spread the workload over the available machines in the most cost effective way |

# Disadvantages of DS

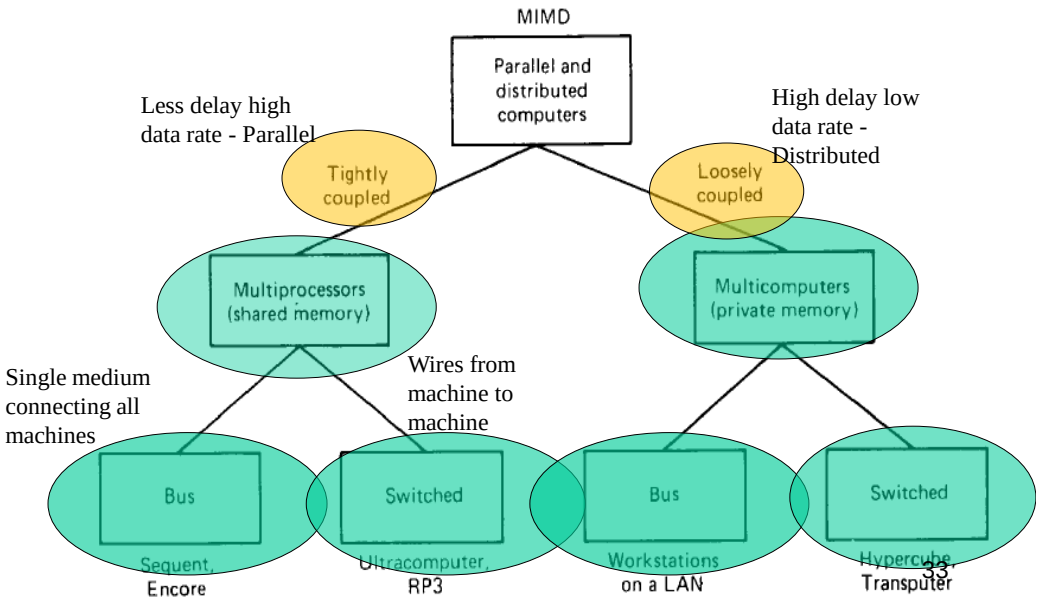
| | |
|---|---|
| Software | Little software exists at present for distributed systems |
| Networking | The network can saturate or cause other problems |
| Security | Easy access also applies to secret data |



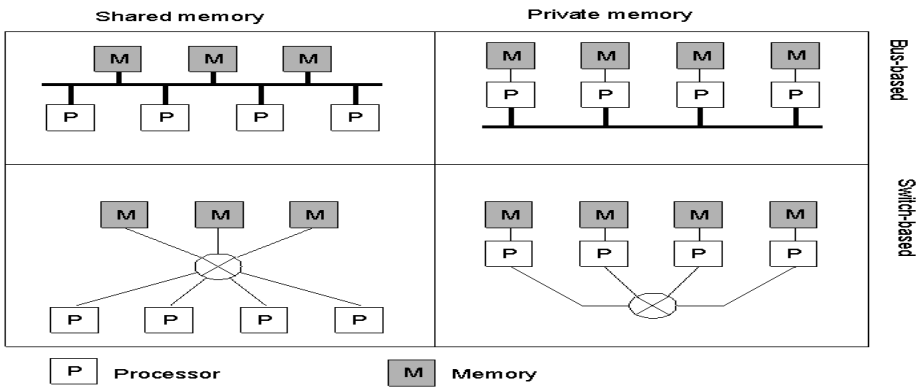
# Classification

- SISD: Single Instruction and Single stream (traditional computer)
- SIMD: Single Instruction and Multiple Stream (Array of processors with one instruction to be processed on multiple data handled by multiple data units in parallel) eg: supercomputers
- MIMD: Multiple instruction and multiple streams – group of independent computers each with its program counter, program and data. (Distributed Systems)

# MIMD Classification

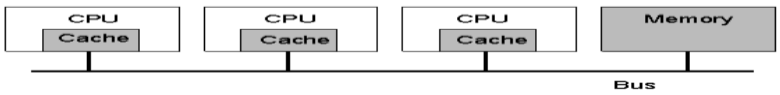



# Hardware Concepts




Different basic organizations and memories in distributed computer systems

# Multiprocessors (1)




A bus-based multiprocessor.

- Bus has address lines, data lines and control lines in parallel.
- Single coherent memory ( memory written is readable to others coherently after a minute delay)
- Cache memory to increase performance
- Write through cache for uniformity of caches (i.e word written to cache is written to memory as well so that new CPU accessing it gets updated value)
- Limited can have at most 64 CPUs

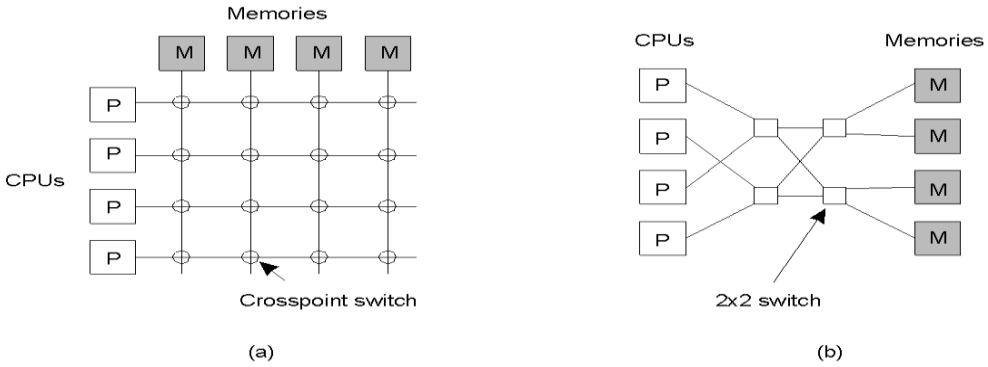


# Multiprocessors (3)

- **Write through cache** for uniformity of caches (i.e word written to cache is written to memory as well so that new CPU accessing it gets updated value)

- **Snoopy cache** – cache constantly monitors the bus for any write occurring to a memory address that is in the cache and updates it.

# Multiprocessors (2)




a) A crossbar switch
b) An omega switching network



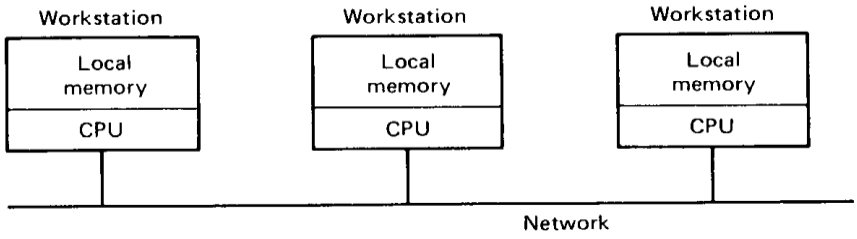
# Multiprocessor

- **Crossbar Switch:**
  - Every intersection between CPU and memory is a physical switch that can be opened or closed
- Many CPUs can access memory at the same time provided memory locations are different
- Disadvantage – n CPU, n memory require $n^2$ switches

- **Omega network:**
- Contains 2x2 switch
- n CPU n memory location require $\log_2 n$ switching stages
- Delay due to switching stages

- Conclusion: tightly coupled, shared memory multiprocessor is difficult and expensive

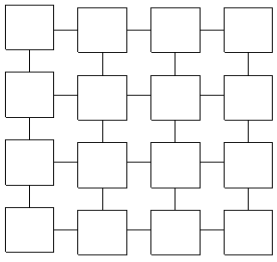
# Bus Based Multicomputer

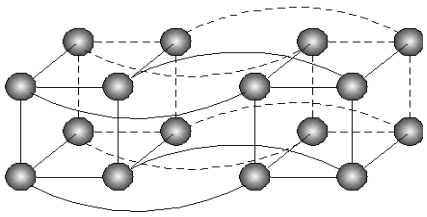



- Each computer has own memory and communicate across bus
- Since traffic is less compared to multiprocessor bus can be a lower speed LAN



# Homogeneous Multicomputer Systems




a) Grid

b) Hypercube

# Types of Distributed Systems

- **Distributed Computing System**
- used for high-performance computing tasks

- **Distributed Information System –**
- interoperatabilty & communication of network applications in organization

- **Distributed Pervasive Systems**

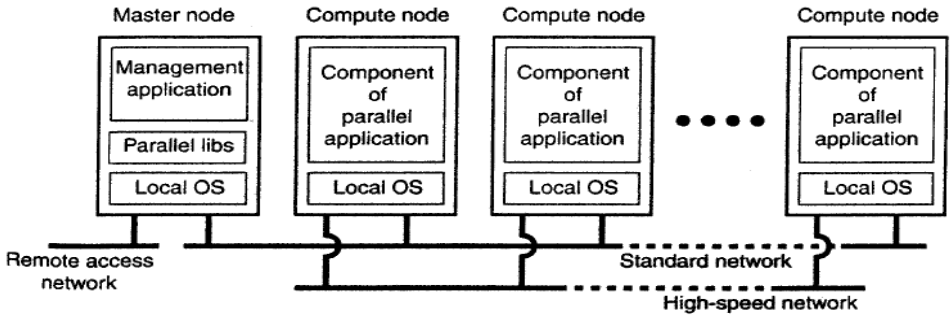


# Distributed Computing System(1)

**i) Cluster computing**
- underlying hardware consists of similar workstations closely connected by means of a highspeed LAN
- each node runs same OS

**ii) Grid computing**
-consists of distributed systems often constructed as federation of computer systems, where each system may fall under different Administrative domain, and may have different hardware, software, and deployed network technology.

# Cluster Computing




- Build supercomputer by simply hooking up collection of simple computers in high-speed network.
- Cluster computing is used for parallel programming in which single (compute intensive) program is run in parallel on multiple machines



# Linux Cluster

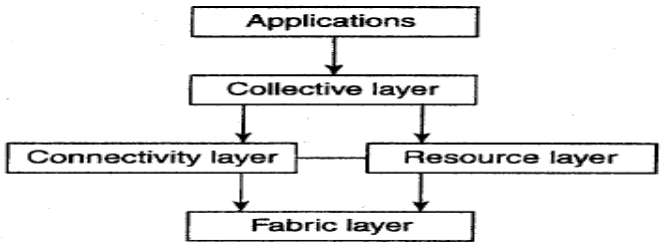
- Each cluster consists of collection of compute nodes controlled and accessed by master node that:
  - allocates nodes to particular parallel program
  - maintains a batch queue of submitted jobs
  - provides interface for users of system.
- Master runs middleware for execution of programs and management of cluster, while compute nodes only need a standard OS.
- Middleware has libraries for executing parallel programs and effectively provides advanced message-based communication facilities, handling faulty processes, security, etc.

# Grid Computing

- Computing systems have a high degree of heterogeneity
- Resources from different organizations are brought together to allow the collaboration
- Software provides access to resources from different administrative domains, users and applications that belong to a specific virtual organization

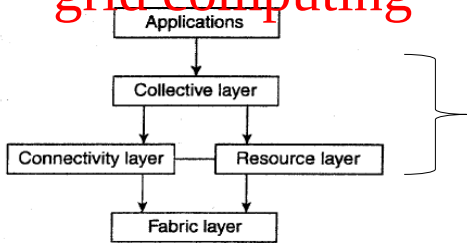


# Layered architecture for grid computing




- **Fabric layer**
  - Provides interfaces to local resources at a specific site.
  - Allows sharing of resources within a virtual organization
- **Connectivity Layer**
  - Communication protocol for grid transactions for usage of multiple resources
  - Security protocols to authenticate users and resources

# Layered architecture for grid computing




•**Resource Layer**
- Responsible for managing a single resource
- Functions for obtaining configuration information on a specific resource, perform specific operations such as creating a process or reading data

•**Collective layer**
- Access multiple resources

-Services for resource discovery, allocation & scheduling tasks onto multiple resources, data replication

•**Application layer**
- Applications that operate within virtual organization &which use grid computing



# Difference between Cluster and Grid computing

**Characteristics of Grid Computing**
•Loosely coupled (Decentralization)
•Diversity and Dynamism
•Distributed Job Management & scheduling

**Characteristics of Cluster computing**
•Tightly coupled systems
•Single system image
•Centralized Job management & scheduling system

# Distributed vs Cloud Computing

- Distributed computing/distributed system involve breaking up a problem which can be solved by a group of computers working at the same time.

- Cloud computing usually refers to providing a service via the internet. That service can be pretty much anything, from business software that is accessed via the web to off-site storage or computing resources.



# Cloud vs Grid Computing

- **Grid computing:**

-Used in environments where users make few but large allocation requests. Eg: lab may have 1000 node cluster and users make allocations for all 1000, or 500, or 200, etc.
- only a few of these allocations can be serviced at a time and others need to be scheduled for when resources are released
-results in sophisticated batch job scheduling algorithms of parallel computations.

- **Cloud computing:**

- lots of small allocation requests. The Amazon EC2 accounts are limited to 20 servers each by default and lots and lots of users allocate up to 20 servers out of the pool of many thousands of servers at Amazon.
- Allocations are real-time and there is no provision for queuing allocations until someone else releases resources

# Distributed Information System

Networked applications can be integrated to form enterprise-wide information system

- Types:

i) **Transaction Processing Systems (Database – self study)**

- server running application (often including a database) and making it available to remote programs, called clients.

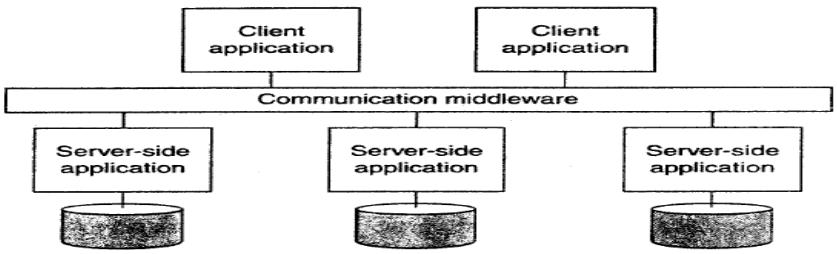
- clients wrap a number of requests, possibly for different servers, into a single larger request and have it executed as a distributed transaction

ii) **Enterprise Application Integration**

- integration should also take place by letting applications communicate directly with each other



# Enterprise Application Integration



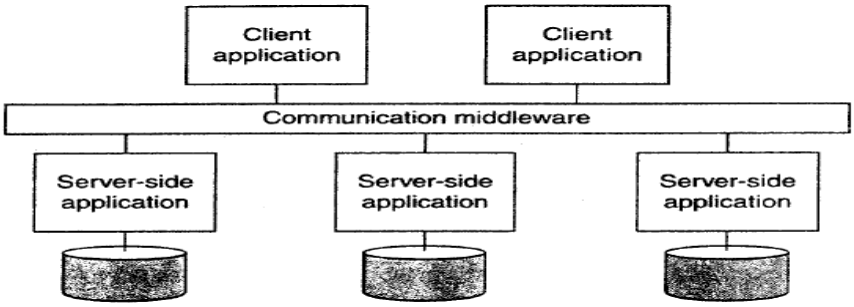

Integrate applications independent from their databases &communicate directly through Middleware in the form of :

i) **Remote procedure calls (RPC):**

- Application component can effectively send request to another application component by doing local procedure call, which results in request sent as message to callee.

- Result sent back and returned to application as result of procedure call

# Enterprise Application Integration




ii) **Remote Method Invocation ( RMI)**

- allow calls to remote objects instead of application
- Disadvantage: caller and callee need to be up and running at time of communication



# Message Oriented middleware (MOM)

- Disadvantage of RPC & RPI:
  - caller and callee need to be up and running at time of communication
  - need to know exactly how to refer to each other

- **Message Oriented Middleware (MOM)**
- applications send messages to logical contact points, described by means of subject and specifying type of message.
- Middleware takes care those messages are delivered to applications.
- Publish/subscribe systems form an important and expanding class of distributed systems

# Distributed Pervasive System

- Consists of devices -small, battery-powered, mobile, and having only a wireless connection hence not very stable
- Devices configured by their owners, but need to automatically discover environment
- Requirements:

1. **Embrace contextual changes**.
 - device aware that environment may change
 - on network disconnectivity application should react, possibly by automatically connecting to another network, or taking appropriate actions
2. **Encourage ad hoc composition**.
- easy to configure the suite of applications running on a device
3. **Recognize sharing as the default**.
- easily read, store, manage, and share information
- efficiently discover services



# Example of pervasive systems

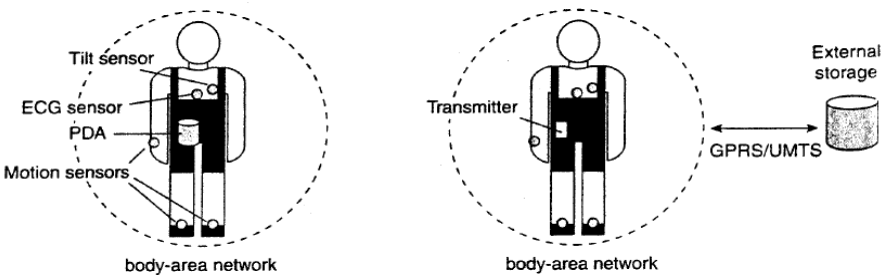
- **Home Systems:**

- consist of one or more PCs, gaming devices, smart phones, surveillance cameras, clocks, kitchen appliances hooked to distributed system
- Self configuring and self managing
- Universal Plug and Play (UPnP) standards by which devices automatically obtain IP addresses, can discover each other

- **Electronic Health Care System**

- devices are being developed to monitor the well-being of individuals and to automatically contact physicians when needed and avoid hospitalization
- various sensors organized in a (preferably wireless) body-area network (BAN) minimally hindering a person, operating while person is on the move

# Pervasive Electronic Health Care



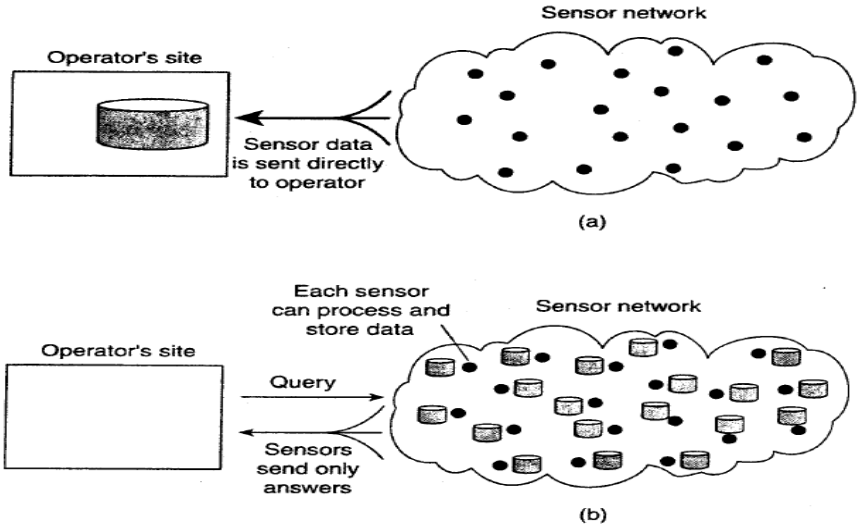

Monitoring a person in a pervasive electronic health care system, using:
(a) Local hub collects data as needed and offloads it to external storage device time to time.
(b) Continuous wireless connection hooked up to external storage



# Sensor Networks

- Used for processing information and form the basis for many medium-scale distributed systems
- Consists of tens to hundreds or thousands of relatively small nodes, each equipped with a sensing device, battery operated
- Distributed Database

# Sensor Network database




Storing and processing data:
(a) only at the operator's site or (b) only at the sensors.



# Sensor Network database

a) Sensors do not cooperate but simply send their data to centralized database located at operator's site
- may waste network resources and energy

b) Forward queries to relevant sensors and let each compute an answer, requiring operator to sensibly aggregate returned answers
- wasteful as it discards aggregation capabilities of sensors which would allow much less data to be returned to operator

# In-network processing

- Forward query to all sensor nodes along a tree encompassing all nodes and subsequently aggregate results as they are propagated back to root, where initiator is located

- Aggregation will take place where two or more branches of the tree come to together

- TinyDB, implements a declarative(database) interface to wireless sensor networks
-use any tree-based routing algorithm.
-intermediate node will collect and aggregate tree results from its children, along with its own findings, and send that toward root



# Summary

- Distributed systems consist of autonomous computers that work together to give the appearance of a single coherent system.
- Advantage:
 - Makes easier to integrate different applications running on different computers into a single system.
- Issues : Resource Availability, Distribution Transparency, Openness, Scalability
- Types of Systems:
 Distributed Computing System, Distributed Information System, Distributed Pervasive Systems

# Resources

Distributed Systems - Tanenbaum
http://www.jatit.org/research/introduction_grid_computing.htm
http://www.thepicky.com/tech/difference-cloud-computing-vs-grid-computing/