Chapter one

Characterization of distributed systems

1.1 Introduction

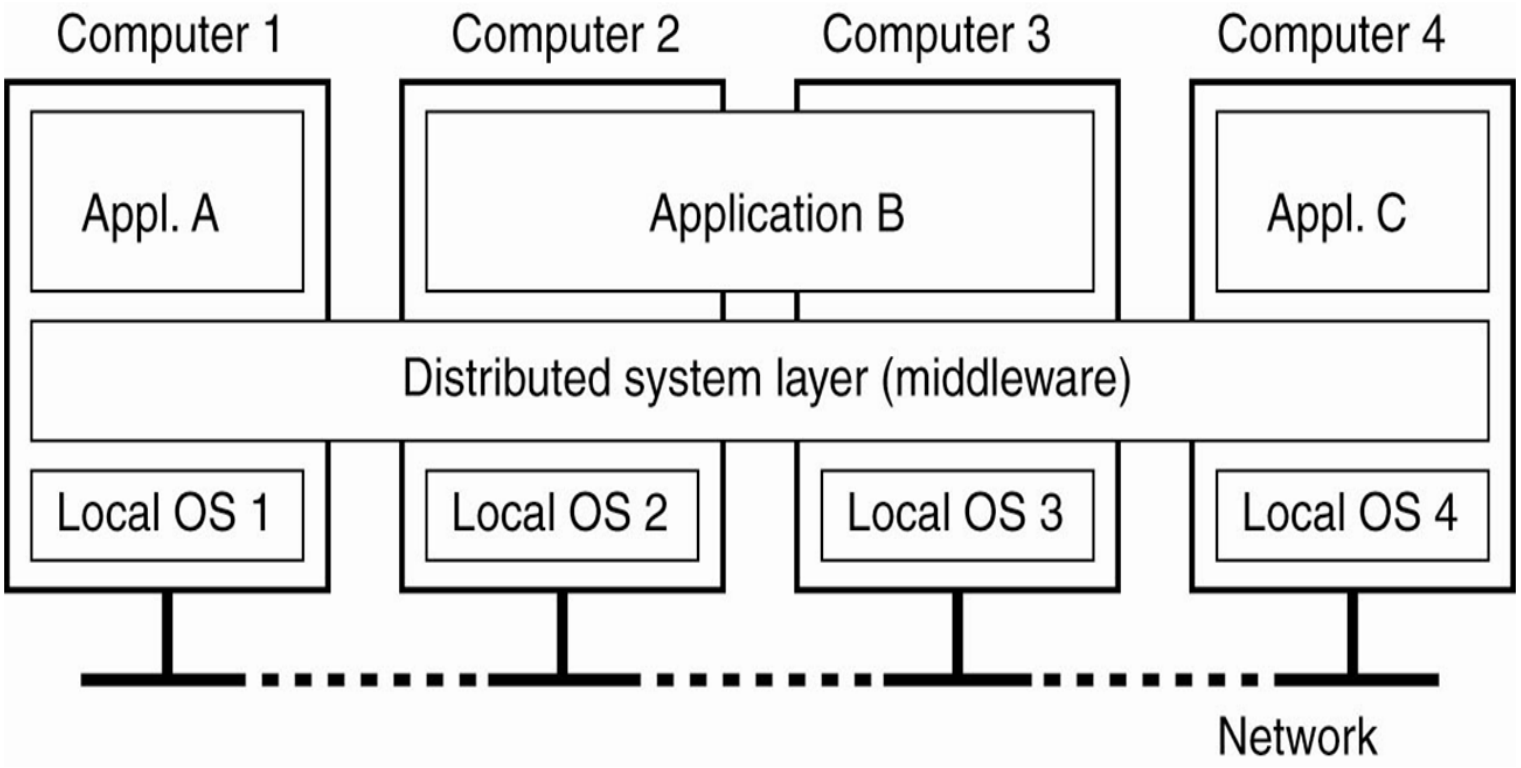
- computer systems are undergoing a revolution
- Since 1945 -1985
  - computers were large and expensive
  - slow in processing instructions
  - there was lack of a way to connect them.
  - operated independently from one another.
- Around the mid-1980s, two major developments in technology began to change
  - development of cheap and powerful microprocessor-based computers (8,16,32 and 64 bits)
  - the invention of high-speed computer networks(LAN and WAN)
- This results introduction of distributed systems

1.2. Definition of Distributed System

- Is **a *collection of independent computers*** *that* ***appears to its users as a single coherent system***
  - *It* consists of components that are **autonomous**
  - **users** (be the people or programs) think that they are dealing with a single system
- *Is a system in which* ***hardware or software components located at networked computers communicate and coordinate their actions only by passing messages****.*
  - This def. of DS has the following consequences
    - Concurrency
    - No global clock
    - Independent Failure

- Concurrency
  - In computer networks, concurrent program execution is a norm.
  - I can do my work on my computer while you do your work on yours, sharing resources such as web pages or files when necessary
- No global clock
  - When programs need to cooperate they coordinate their actions by exchanging messages.
  - There is no single global notion of the correct time, the only communication is by sending messages through a network
- Independent failures
  - Each component of the system can fail independently, leaving the others still running

- *A system designed to support the development of applications and services which can exploit a physical architecture consisting of multiple, autonomous processing elements that do not share primary memory but cooperate by sending asynchronous messages over a communication network.*



1.3. Challenges in designing a Distributed System

- *Heterogeneity:*
  - Users access services and run applications over a heterogeneous collection of computers and networks.
    - networks, computer hardware, operating systems, programming languages, implementations by different developers.
    - ***Middleware***: applies to a software layer that provides a programming abstraction as well as masking the heterogeneity of the underlying networks, hardware, operating systems and programming languages. (CORBA: common object request broker and RMI: Java remote method invocation)(implemented in Over the IP)
    - ***Internet protocol*** masks the differences of networks
    - ***Programs*** written by d/t developers use common standards to communicate

- *Openness:*
  - is determined primarily by the degree to which **new resource-sharing services can be added** and be made available for use by a variety of client programs.
- *Security*:
  - has three components: **confidentiality** (protection against disclosure to unauthorized individuals), **integrity** (protection against alteration or corruption), and **availability** (protection against interference with the means to access the resources).
- *Scalable:*
  - remains effective when there is a significant increase in the number of resources and the number of users

- *Failure handling*
  - handling of failures is particularly difficult like detecting failures(checksum), masking failures(retransmitting and replicating), tolerating failures(try again later) and recovery from failures(roll back) since systems fail independently
- *Concurrency*
  - a possibility that several clients can access a shared resource at the same time. These multiple requests may conflict with one another and produce inconsistent results.
- *Transparency*
  - the system is perceived as a whole rather than as a collection of independent components
- *Quality of service*
  - Once we provide the users with the functionalities then ask them about the quality of services
  - Non-functional properties: reliability, security, performance, availability and flexibility

1.4. Goals of Distributed System

- Making Resources Accessible
- Distribution Transparency
- Openness in a Distributed System
- Scalability in Distributed Systems

Making Resources easily Accessible

- Resources include things like printers, computers, storage facilities, data, files, Web pages, and networks
- The reason for sharing resources is economy i.e. to share costly resources such as supercomputers, high-performance storage systems, printers, and other expensive peripherals to make exchange of information easy

Distribution Transparency

- to hide the fact that its processes and resources are physically distributed across multiple computers

Transparency	Description
Access	Hide differences in data representation and how a resource is accessed.
Location	Hide where a resource is physically 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.

Openness in a Distributed System

- An open distributed system is a system that offers services according to standard rules that describe the syntax and semantics of those services.
- In computer networks, standard rules govern the format, contents, and meaning of messages sent and received(protocols)
- In distributed systems, services are generally specified through interfaces, which are often described in an Interface Definition Language (IDL)
- Extensibility, interoperability and portability are also goals for open distributed system.
- **portability** :an application developed for a distributed system A can be executed without modification, on a different distributed system B that implements the same interfaces as A.
- **Interoperability**: the extent by which two implementations of systems or components from different manufacturers can co-exist and work together by merely relying on each other's services.

Scalability in Distributed Systems

- Distributed systems operate effectively and efficiently at many different scales, ranging from a small intranet to the Internet.
- A Distributed System is described as scalable if it will remain effective when there is a significant increase in the number of resources and the number of users
- Allows the system and applications to expand in scale without change to the system structure or the application algorithms.
- Scalability
  - can easily add more users and resources to the system
  - geographically scalable system is one in which the users and resources may lie far apart.
  - system can be administratively scalable, spanning that it can still be easy to manage even if it spans many independent administrative organizations
- Techniques to achieve scalability includes
  - Scaling up, hiding the communication, partitioning and distribution, replication, catching

1.5. Types of Distributed System

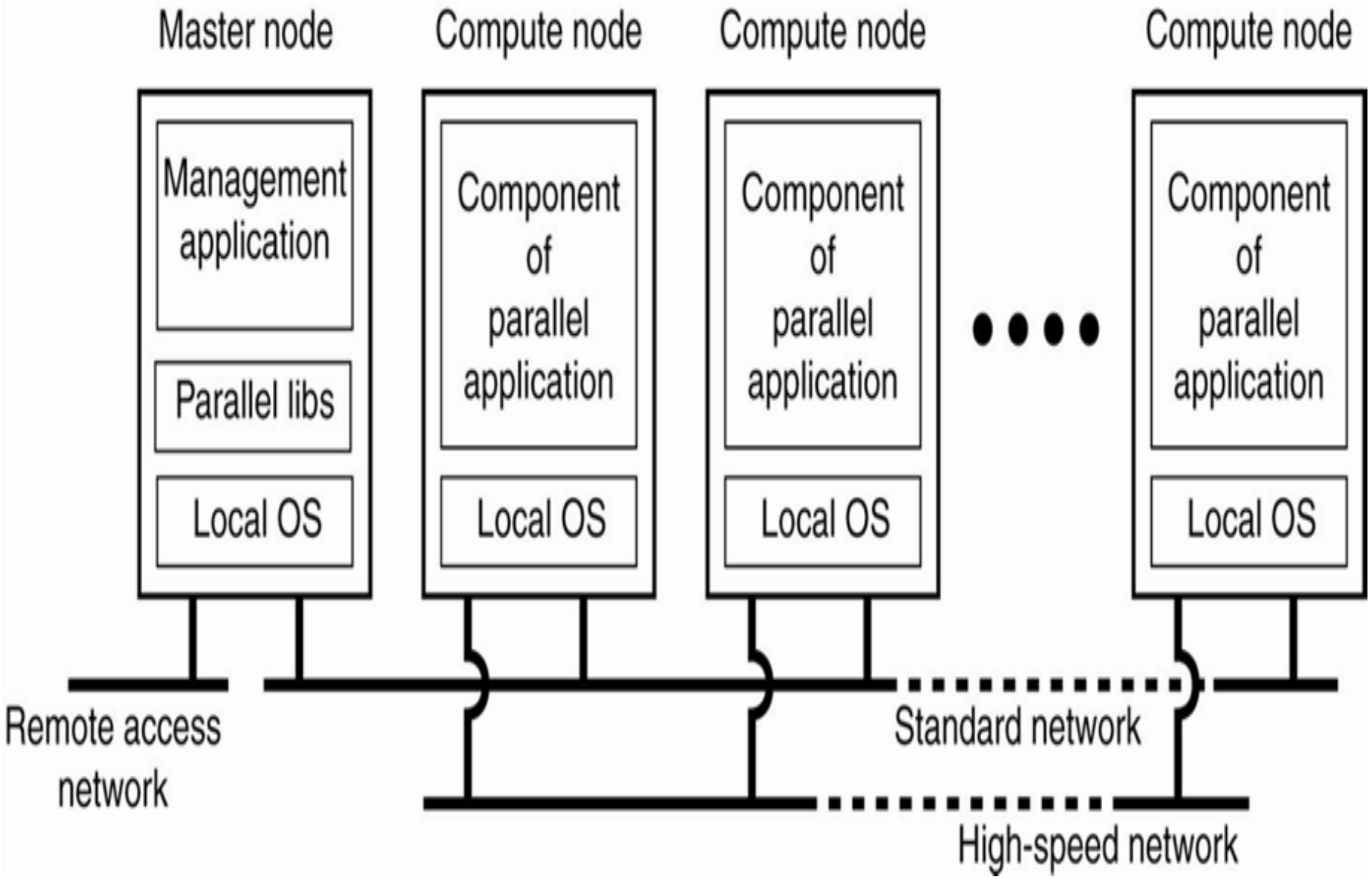
- Distributed Computing Systems
- Distributed Information Systems
- Distributed Pervasive Systems

Distributed Computing Systems

- used for high-performance computing tasks
  - Cluster computing
  - Grid computing
  - Cloud computing

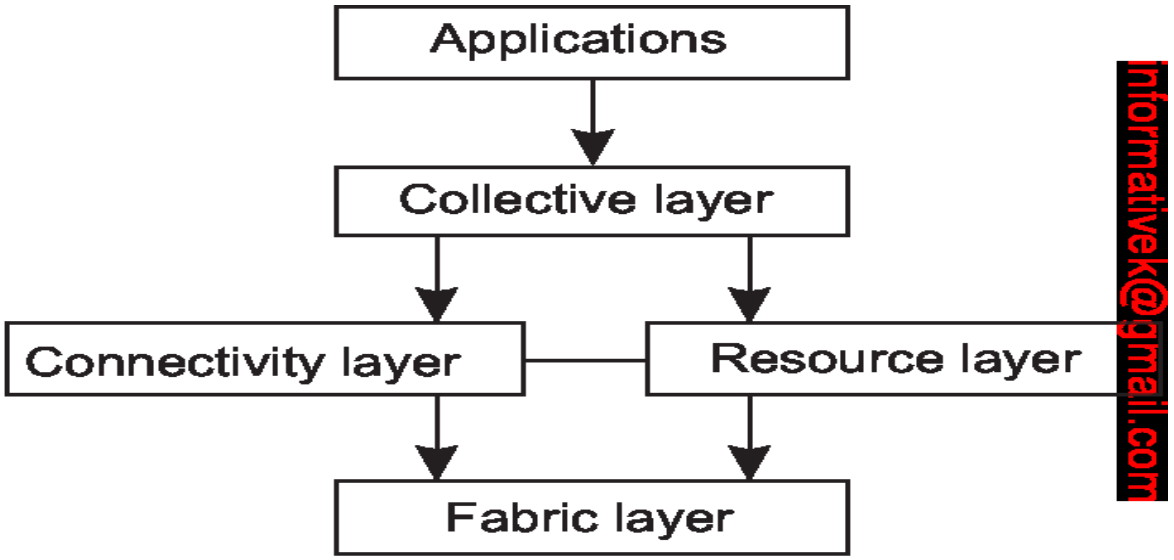
Cluster computing

- consists of a collection of similar workstations or PCs (homogeneous), closely connected by means of a high speed local-area network
- each node runs the same operating system
- used for parallel programming in which a single compute intensive program is run in parallel on multiple machines
- compute nodes of each cluster are controlled and accessed by means of a single master node
- The master
  - 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
  - runs the middleware needed for the execution of programs and management of the cluster


Master node
Compute node
Compute node
Compute node
Management application
Parallel libs
Local OS
Component of parallel application
Local OS
Component of parallel application
Local OS
Component of parallel application
Local OS
Remote access network
Standard network
High-speed network

Grid computing

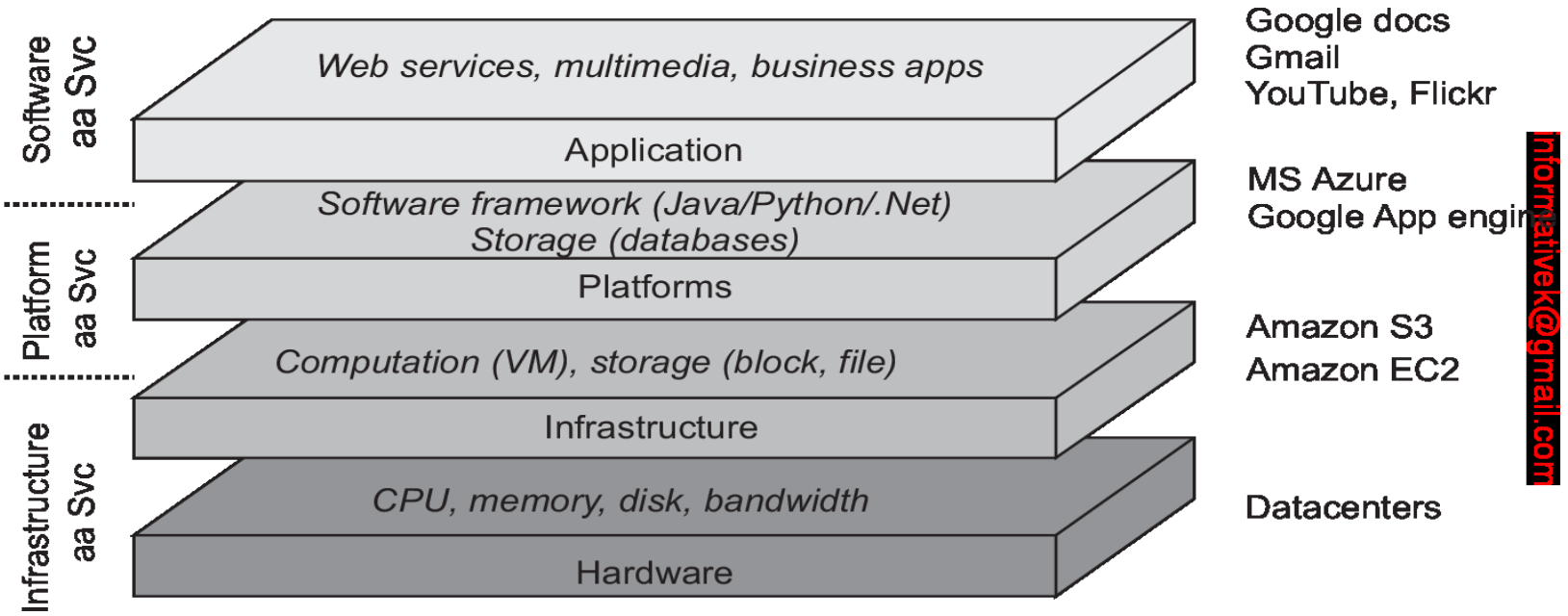
- consists of distributed systems that are often constructed as a federation of computer systems
- high degree of heterogeneity: no assumptions are made concerning hardware, operating systems, networks, administrative domains, security policies, etc.
- resources from different organizations are brought together to allow the collaboration of a group of people or institutions.



A layer architecture for a grid computing system

Cloud computing

- Simply *outsource* the entire infrastructure that is needed for compute-intensive applications
- proving the facilities to dynamically construct an infrastructure and compose what is needed from available services.
- Unlike grid computing, which is strongly associated with high-performance computing, cloud computing is much more than just proving lots of resources.
- Cloud-computing providers offer these layers to their customers through various interfaces (CL tools, Programming Interfaces and web interfaces), leading to three different types of services:
  - IaaS –covering hardware and Infrastructure layer
  - PaaS-covering the platform layer
  - SaaS in which their applications are covered



The organization of clouds

Distributed Information Systems

- a networked application simply consisted of a server running that application and making it available to remote programs (clients)
- Clients send requests and get responses from the server

Transaction processing systems

- operations on a database are usually carried out in the form of transactions

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

Transactions properties:

- *Atomic:* a transaction either happens completely or not at all
- *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

Distributed Pervasive Systems

- The last 2 types of DS are characterized by their stability; fixed nodes having high-quality connection to a network.
- DPS include mobile and embedded computing devices which are small, battery-powered, mobile, and with a wireless connection.
- instability is their default behavior
- three requirements for pervasive applications:
  - ***Embrace contextual changes***: a device is aware that its environment may change all the time,
  - ***Encourage ad hoc composition***: devices are used in different ways by different users
  - ***Recognize sharing as the default***: devices join a system to access or provide information
- Ex. Home systems, E-Healthcare system etc

1.6. Common mistakes

- Developing distributed systems is different than developing non-distributed ones.
- Developers with no experience typically make a number of false assumptions when first developing distributed applications.
- All of these assumptions hold for non-distributed systems, but typically do not hold for distributed systems.
  - The network is reliable
  - Latency is zero
  - Bandwidth is infinite
  - The network is secure
  - The topology does not change
  - There is one administrator
  - Transport cost is zero
  - Everything is homogeneous

End of chapter 1!

Questions?