MS (DEIM)
Fall Semester 2024

Distributed Systems (CS-555)

Lecture #1

Introduction to Distributed Systems

Dr Syed Zaffar Qasim
Assistant Professor (CIS)

Distributed Computer Systems

A collection of independent entities appearing to its users as a single coherent system in which software components communicate and coordinate their actions only by passing messages.

- A node may be a complete computer with a full complement of peripherals.
- A typical distributed system consists of thousands of machines, loosely cooperating over the internet.
- Loose coupling of distributed systems is both a strength and a weakness.
  - **Strength**: Computers can be used for a variety of different applications.
  - **Weakness**: Programming the application is difficult due to *lack of* any common underlying platform.

Different features of Distributed Systems

- **Concurrency** Concurrently executing programs sharing resources such as web pages or files.
  - Coordination - an important and recurring topic.
- **No common physical clock** Synchronization of clocks not possible.
  - Programs coordinate their actions by exchanging messages.
- **No shared memory** Key feature that requires message-passing for communication.
  - It may be noted that a distributed system may still provide the abstraction of a common address space via the *distributed shared memory abstraction*.

3

Different features of Distributed Systems

- **Geographical separation** However, it is not necessary for the nodes to be on a WAN.
  - Recently, the *network/cluster of workstations (NOW/COW)* configuration connecting workstations on a LAN is also being increasingly regarded as a small distributed system.
    - ❖ because low-cost, high-speed, off-the-shelf processors now available.
  - The ***Google search engine*** based on the NOW architecture.
- **Autonomy and heterogeneity** Typically the computers are semi-autonomous and loosely coupled while they cooperate to address a problem collectively.
  - Have *different speeds*, may each run *different OS*, each has its *own file system* and be under *different administration*.

4

Different features of Distributed Systems

- Wide range of computers, from *weakly coupled systems* such as WANs, to *strongly coupled systems* such as LANs, to *very strongly coupled systems* such as multiprocessor systems.

- **Independent Failures** Distributed systems can fail in new ways:-
  - ***Faults in network*** result in isolation of computers but the later don't stop working.
    - ❖Programs may not be able to detect whether network failed or becomes unusually slow.
  - ***Failure of a computer*** or unexpected crash of a program is not immediately known to other components.
    - ❖Each component can fail independently.

5

Motivation

- Motivation for constructing and using distributed systems stems from *desire to share resources.*

➢ **Resource**
- Characterizes a wide range of things that can be usefully shared in a networked computer system.

➢ Extends from
- Hardware components such as disks and printers

to

- Software defined entities such as files, databases and data objects of all kinds.
  - ❖It includes the *streams of video frames* that emerge from a digital video camera and the *audio-connection* that a mobile phone call represents.

6

MIDDLEWARE (MW)

- Distributed systems often organized by means of a layer of software logically placed between
  - a higher-level layer consisting of users and applications and
  - a layer underneath consisting of operating systems and basic communication facilities.
- The middleware layer extends over multiple machines, and offers each application the same interface.

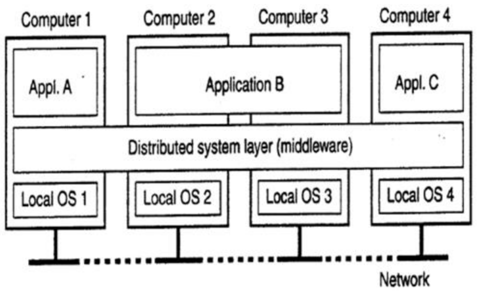


Fig 1: General Model of distributed system

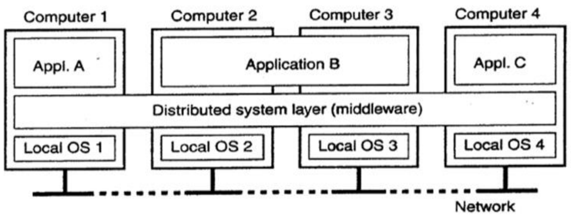
MIDDLEWARE (MW)

- **Purpose**: The middleware provides

  A. masking the heterogeneity of the underlying hardware, networks, operating systems and programming languages while offering a single-system view

  B. programming abstraction (data structures and operations)

  ❖ uniform computation model for use by the programmers of servers and distributed applications.



- Allows communication of
  - different applications to each other
  - components of a single distributed application to each other.

MIDDLEWARE (MW)

- Most middleware is implemented over the Internet protocols which themselves mask the difference of the underlying networks.
- But all MW deals with the difference in OS and hardware.
- Middleware is like the OS of a distributed system.
- Example: **Common Object Request Broker Architecture (CORBA)**
  - Rather than a document or a file, everything is an object.
  - CORBA is a client-server system in which client processes on client machines can invoke operations on objects located on (possibly remote) server machines.
- Some middleware such as Java *Remote Method Invocation* (RMI) supports only a single programming language.

9

TRENDS IN DISTRIBUTED SYSTEMS

1. Distributed Computing Systems,
2. Distributed computing as a utility
3. Mobile and Ubiquitous computing

I. Distributed Computing Systems

- Used for high-performance computing tasks.
- Two subgroups:-

  i. *Cluster computer system*: a set of interconnected similar computers that cooperate closely to provide high-performance computing.
  - In addition, each node runs the same operating system.

  ii.Grid computing: DSs often constructed as a federation of computer systems,
  - each system may fall under a different administrative domain,
  - different hardware, software and network technology.

10

Cluster Computing Systems

- The modern trend is towards utilizing *commodity hardware* both for the computers and for the interconnecting networks.
- In virtually all cases, cluster computing is used for parallel programming in which a single (*compute intensive*) program is run in parallel on multiple machines.
- One well-known example of a cluster computer is formed by Linux-based Beowulf clusters, of which the general configuration is shown in Fig. 2.
- Each cluster consists of
  - a collection of ***compute nodes***
  - that are controlled and accessed by a single ***master node***.

11

Cluster Computing Systems

- The **Master**
  - provides an interface for the users of the system
  - maintains a batch queue of submitted jobs, and
  - handles allocation of nodes to a particular parallel program.
- The master actually runs the middleware needed for the execution of programs and management of the cluster, while the compute nodes often need nothing else but a standard OS.
- Many of master libraries effectively provide only advanced message-communication facilities, but are not capable of handling faulty processes, security, etc.

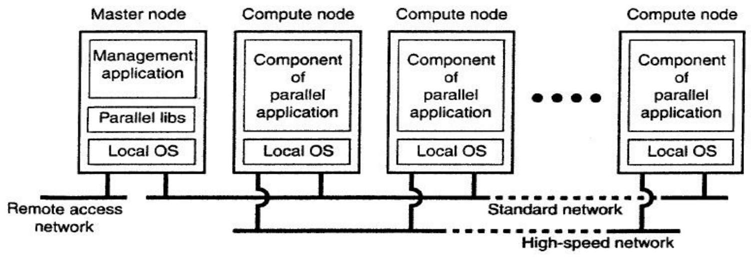


Fig 2

12

Cluster Computing Systems

- As an alternative to this hierarchical organization, a symmetric approach is followed in the MOSIX system.
  - MOSIX offers the ultimate distribution transparency
    - ❖To a process a cluster computer appears to be a single computer.
  - The high degree of transparency is provided by
    - ❖allowing processes to dynamically migrate between the nodes of the cluster.
  - *Process migration* allows a user to start an application on any node (referred to as the *home* node),
    - ❖after which it can transparently move to other nodes,
    - ❖for example, to make efficient use of resources.

13

Grid Computing Systems

- A characteristic feature of cluster computing: *homogeneity*.
- In contrast, grid computing systems have a high degree of heterogeneity.
- A key issue in a grid computing system
  - resources from different organizations are brought together to allow the collaboration of a group of people or institutions.
- Collaboration realized in the form of a **virtual organization**.
- Typically, resources consist of compute servers (including supercomputers, possibly implemented as cluster computers), storage facilities, and databases.
- In addition, special networked devices such as telescopes, sensors, etc., can be provided as well.

14

Grid Computing Systems

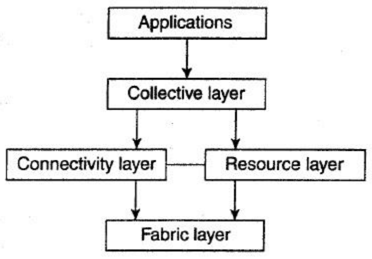
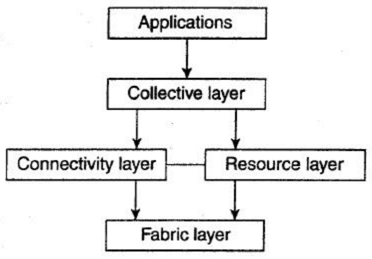


Fig 3

- The architecture consists of five layers.
- The lowest ***fabric layer*** provides interfaces to local resources at a specific site.
- These interfaces are tailored to allow sharing of resources within a virtual organization.
- Typically, they will provide functions for *querying the state and capabilities of a resource*, along with functions for *actual resource management* (e.g., locking resources).

15

Grid Computing Systems



- The *connectivity layer* consists of communication protocols (DNS/TCP/IP) for supporting grid transactions that span the usage of multiple resources.
- For example, protocols are needed to transfer data between resources, or to simply access a resource from a remote location.
- In addition, this layer will contain security protocols to authenticate users and resources.

16

Grid Computing Systems

- Note that in many cases, programs acting on behalf of the human users are authenticated.
- In this sense, delegating rights from a user to programs is the function to be supported in the connectivity layer.
- The *resource layer* is responsible for managing a single resource.
- It uses the functions provided by the connectivity layer and calls directly the interfaces of the fabric layer.
- For example, it will offer functions for
  - obtaining configuration information on a specific resource,
  - monitoring & control and
  - accounting for sharing operations on individual resources.
- The resource layer is thus *responsible for access control*, and hence will rely on the authentication performed as part of the connectivity layer.

17

Grid Computing Systems

- The next layer in the hierarchy is the *collective layer.*
- It deals with handling access to multiple resources
- Typically consists of services for resource discovery, allocation and scheduling of tasks onto multiple resources, data replication, and so on.
- Unlike the connectivity and resource layer, which consist of a relatively small, standard collection of protocols,
  - the collective layer may consist of many different protocols for many different purposes.
- Finally, the *application layer* consists of the applications that operate within a virtual organization.
- Typically the collective, connectivity, and resource layer form the heart of a ***grid middleware layer***.

18

Distributed computing as a utility

- **Concept of Distributed computing as a utility?**
  - With the increasing maturity of distributed systems infrastructure, the view of distributed resources as a utility (or commodity) is emerging.
  - It represents the analogy between distributed resources and other utilities such as water or electricity.
  - With this model, resources (physical/logical) are provided by appropriate service suppliers and effectively *rented* rather than owned by the end user.
- *Physical resources* such as storage and processing can be made available to networked computers.

19

Distributed computing as a utility

- At one end of the spectrum, a user may
  - opt for a remote *storage facility* for files (e.g., for multimedia data such as photographs, music or video) and/or for backups.
  - rent one or more *computational nodes* to perform distributed computation.
- At the other end of the spectrum, users can access
  - sophisticated *data centres* (networked facilities offering access to repositories of often large volumes of data) or
  - indeed *computational infrastructure* using the sort of services now provided by companies such as Amazon and Google.

20

Distributed computing as a utility

- ***Software resources***
  - Distributed calendar services where people use diary agents to negotiate dates with each other.
  - Amazon web services
  - Google Apps. bundles a range of business services
- Operating system virtualization is a key enabling technology for this approach.
- The goal of system virtualization is
  - to provide multiple virtual machines (virtual hardware images) over the underlying physical machine architecture,
  - with each virtual machine running a separate operating system instance.
- It implies that users may actually be provided with services by a virtual rather than a physical node.
- This offers greater flexibility to the service supplier in terms of resource management.

21

Distributed computing as a utility

- The term *cloud computing* is used to capture the vision of computing as a utility.

➢ **Cloud**: a set of Internet-based *application*, *storage* and *computing* services to support most users' needs, thus enabling them to largely or totally dispense with local data storage and application software (see Fig 4).

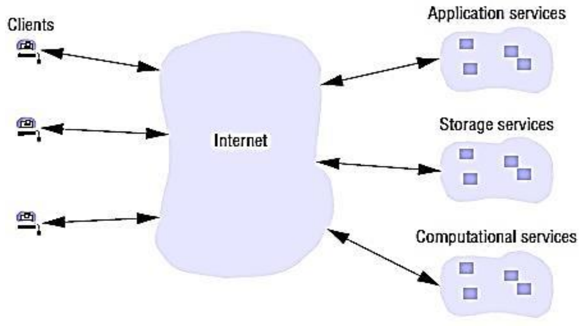
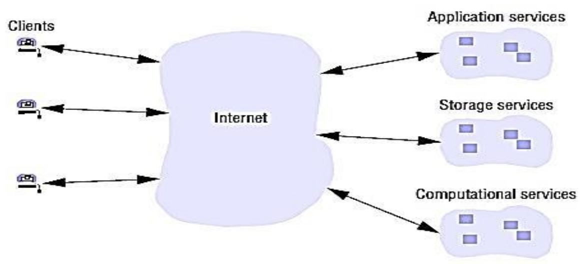


Fig 4: Cloud Computing

22

Distributed computing as a utility

- The *services* often charged on per-usage basis rather than purchased.
- Cloud computing reduces requirements on users' devices, allowing very simple desktop or portable devices to access a potentially wide range of resources and services.
- Clouds are generally implemented on cluster computers to provide the necessary scale and performance required by such services.



23

Mobile and Ubiquitous Computing

- ➢ *Mobile computing* is the performance of computing tasks while the user is moving, or visiting places other than their usual environment.
- Broadly speaking, mobile computing is concerned with exploiting the connectedness of many small portable devices.
- The devices include laptop, netbook or tablet computer,
  - each with several forms of wireless connectivity including cellular networks, WiFi and Bluetooth.
- Smart phones have computing functionality by virtue of running operating systems such as Google's *Android*, Apple's *iOS* or *Windows Phone 7* from Microsoft.
- For example, a user can read a barcode via a smart phone's camera to obtain price-comparison information.
- Smart phones also often have built-in GPS units for navigation and other location-specific purposes.

24

Ubiquitous Computing

- Ubiquitous computing is about exploiting the increasing integration of computing devices with our everyday physical world.
- **Mark Weiser** coined the term *ubiquitous computing* in 1988.
- Also sometimes known as *pervasive computing*.
- Weiser's vision of ubiquitous computing was characterized by two main shifts in computing paradigm.
- First shift: *one person, many computers*
- In ubiquitous computing, computers multiply in form and function, not just in number, to suit different tasks.
- e.g., suppose all the inert display and writing surfaces in a room – whiteboards, books, papers, etc. were replaced by tens or hundreds of individual computers with electronic displays.

Ubiquitous Computing

- Books already appear in an electronic form.
- Now suppose we embed computing functionality in all the writing implements.
- For example, pens and markers become able to store what the user has written and drawn, and to collect and move multimedia content between the many computers lying around.
- The second shift that Weiser predicted: *computers would 'disappear'*.
- Small computing devices will eventually become so pervasive in everyday objects that they are scarcely noticed.
- We don't consider washing machines, refrigerators or vehicles as 'computing devices', even though embedded microprocessors control them (about 100 microprocessors, in the case of some cars).

Ubiquitous Computing

- **Connectivity in ubiquitous computing**
  - The presence of computers everywhere only becomes useful when they can communicate with one another.
  - For example, it may be convenient for users to control their washing machine or their entertainment system from their phone or a 'universal remote control' device in the home.
  - Equally, the washing machine could notify the user via a smart badge or phone when the washing is done.

27

Ubiquitous Computing

- Ubiquitous and mobile computing ***overlap***, since the mobile user can in principle benefit from computers that are everywhere.
- But they are distinct, in general.
- Ubiquitous computing could benefit users while they remain in a single environment such as the home or a hospital.
- Similarly, mobile computing has advantages even if it involves only conventional, discrete computers and devices such as laptops and printers.

28

Ubiquitous Computing

- Fig 5 shows a user visiting a host organization.

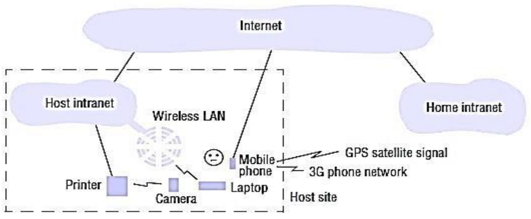


Fig 5: Portable and handheld devices in a distributed system

- The figure also shows the user's home intranet.
- Both intranets are connected to the rest of the Internet.
- The user has access to three forms of wireless connection.
- Their laptop has a means of connecting to the host's wireless LAN.
- This network provides coverage of a few hundred metres (a floor of a building, say).
- It connects to the rest of host intranet via a gateway or access point.

29

Ubiquitous Computing

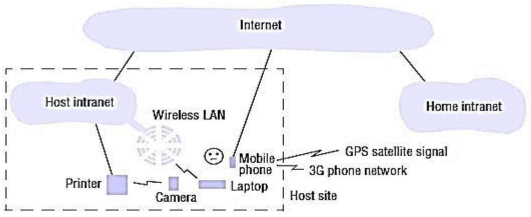
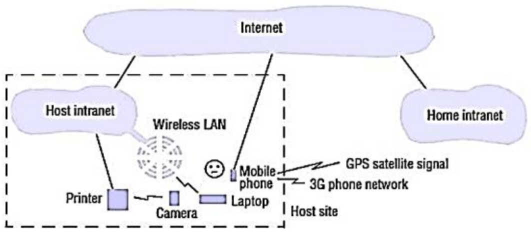


Fig 5 : Portable and handheld devices in a distributed system

- The user also has a mobile (cellular) telephone, which is connected to the Internet.
- The phone gives access to the Web and other Internet services, and may also provide location information via built-in GPS functionality.
- Finally, the user carries a digital camera, which can communicate over a personal area wireless network (with range up to about 10m) with a device such as a printer.

30

Ubiquitous Computing



- With a suitable system infrastructure, the user can perform some simple tasks in the host site using the devices they carry.
- While journeying to the host site, the user can fetch the latest stock prices from a web server using the mobile phone and can also use the built-in GPS and route finding software to get directions to the site location.
- During the meeting with their hosts, the user can show them a recent photograph by sending it from the digital camera directly to a suitably enabled (local) printer or projector in the meeting room (a process called *service discovery* or *location service*).

31

Ubiquitous Computing

- This requires only the wireless link between the camera and printer or projector.
- And they can send a document from their laptop to the same printer, utilizing the wireless LAN and wired Ethernet links to the printer.
- This scenario demonstrates the need to support *spontaneous interoperation,* whereby associations between devices are routinely created and destroyed – for example by locating and using the host's devices, such as printers.
- The main challenge of such situations is to make interoperation fast and convenient (i.e., spontaneous) even though the user is visiting an environment for the first time.

32