KTU LECTURE NOTES

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

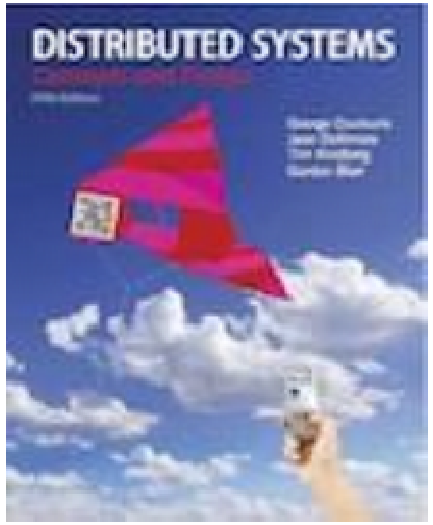
DOWNLOAD KTU ASSIST APP FROM PLAYSTORE

Distributed Computing

Module 1

Prepared by Amal Ganesh, Assistant Professor, Vidya Academy of Science and Technology, Thrissur

Distributed Computing – Text Book



From **Coulouris, Dollimore, Kindberg and Blair**

Distributed Systems:
Concepts and Design

Edition 5, © Addison-Wesley 2012

Course Plan			
Module	**Contents**	**Hours**	**End Sem. Exam Marks**
I	Evolution of Distributed Computing -Issues in designing a distributed system- Challenges- Minicomputer model – Workstation model - Workstation-Server model– Processor - pool model - Trends in distributed systems	7	15%
II	System models: Physical models - Architectural models - Fundamental models	6	15%
FIRST INTERNAL EXAM			
III	Interprocess communication: characteristics – group communication - Multicast Communication –Remote Procedure call - Network virtualization. Case study : Skype	7	15%
IV	Distributed file system: File service architecture - Network file system- Andrew file system- Name Service	7	15%
SECOND INTERNAL EXAM			
V	Transactional concurrency control:- Transactions, Nested transactions-Locks-Optimistic concurrency control	7	20%
VI	Distributed mutual exclusion – central server algorithm – ring based algorithm- Maekawa's voting algorithm – Election: Ring -based election algorithm – Bully algorithm	7	20%
END SEMESTER EXAM			

1 Characterization of distributed systems

1.1 Introduction

What is a Distributed System?

A distributed system is one in which components located at networked computers communicate and coordinate their actions only by passing messages

A distributed system consists of a **collection of** autonomous **computers linked by** a computer **network** and equipped with **distributed system software**. This software enables computers to **coordinate** their activities and to **share the resources of the system hardware, software, and data.**

How to characterize a distributed system?

- Concurrency of components
  - In a network of computers, concurrent program execution is the 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.
  - The capacity of the system to handle shared resources can be increased by adding more resources
- Lack of global clock
  - When programs need to cooperate they coordinate their actions by exchanging messages.
  - Close coordination often depends on a shared idea of the time at which the programs' actions occur.
  - But it turns out that there are limits to the accuracy with which the computers in a network can synchronize their clocks
  - There is no single global notion of the correct time.

How to characterize a distributed system?

- Independent failures of components
  - All computer systems can fail, and it is the responsibility of system designers to plan for the consequences of possible failures.
  - Distributed systems can fail in new ways. Faults in the network result in the isolation of the computers that are connected to it, but that doesn't mean that they stop running.
  - In fact, the programs on them may not be able to detect whether the network has failed or has become unusually slow.

Evolution of Distributed Computing System

- Distributed systems (to be exact, distributed computer systems) has come a long way from where it was started.
- At the very beginning, one computer could only do one particular task at a time.
- If we need multiple tasks to be done in parallel, we need to have multiple computers running in parallel.
- But running them parallel was not enough for building a truly distributed system since it requires a mechanism to communicate between different computers (or programs running on these computers).
- This requirement of exchanging (sharing) data across multiple computers triggered the idea of the message-oriented communication where two computers share data across using a message which wraps the data.
- There were few other mechanisms like file sharing, database sharing also came into the picture.

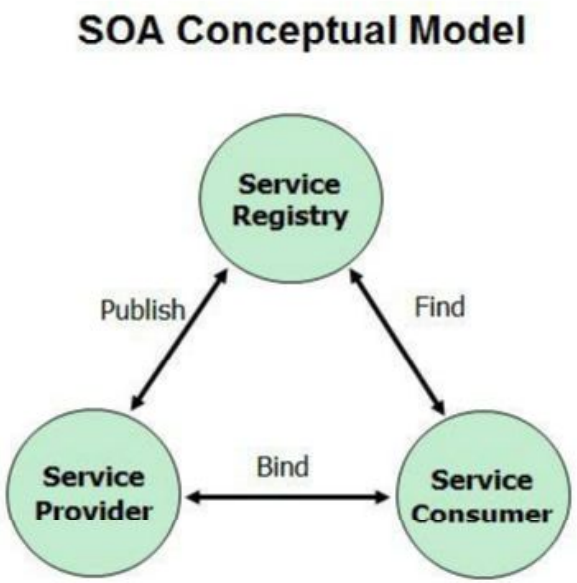
Evolution of Distributed Computing System

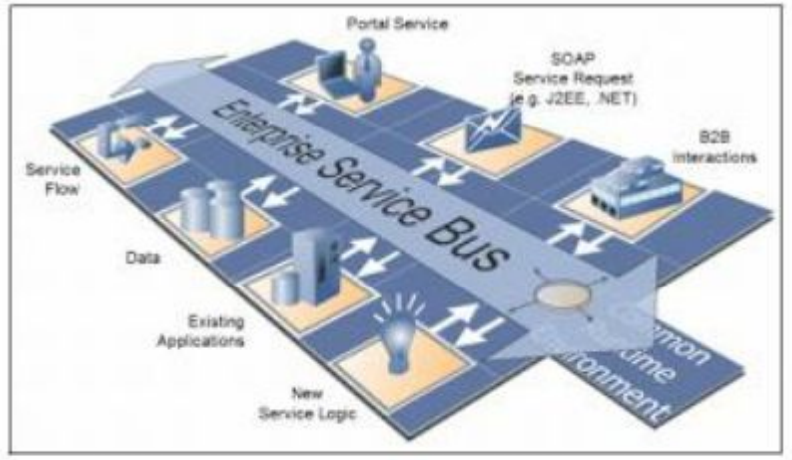
- Then came the era of multitasking operating systems and personal computers.
- With Windows, Unix, Linux operating systems, it was possible to run multiple tasks on the same computer.
- This allowed distributed systems developers to build and run an entire distributed system within one or few computers which are connected over messaging.
- This lead to the Service Oriented Architecture (SOA) where each distributed system could be built with integrating a set of services which are running on either one computer or multiple computers.
- Service interfaces were properly defined through a WSDL (for SOAP) or WADL (for REST) and the service consumers used those interfaces for their client-side implementations.

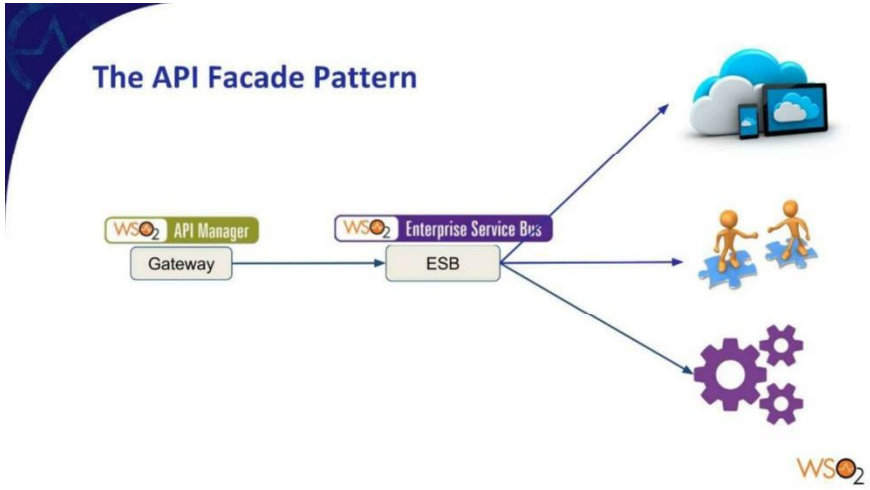
Evolution of Distributed Computing System

- With the reduction of price for computing power and storage, organizations all over the world started using distributed systems and SOA based enterprise IT systems.
- Once the number of services or systems increased, the point to point connection of these services was no longer scalable and maintainable.
- This leads to the concept of centralized “Service Bus” which interconnects all different systems via a hub type architecture.
- This component is called the ESB (Enterprise Service Bus) and it acts as a language translator or a middleman between a set of people who talk different languages but wanted to communicate with each other.
- The main advantage of this model was that each system can build server side and client side implementations without worrying about the protocols of the connecting systems.

Evolution of Distributed Computing System







Evolution of Distributed Computing System

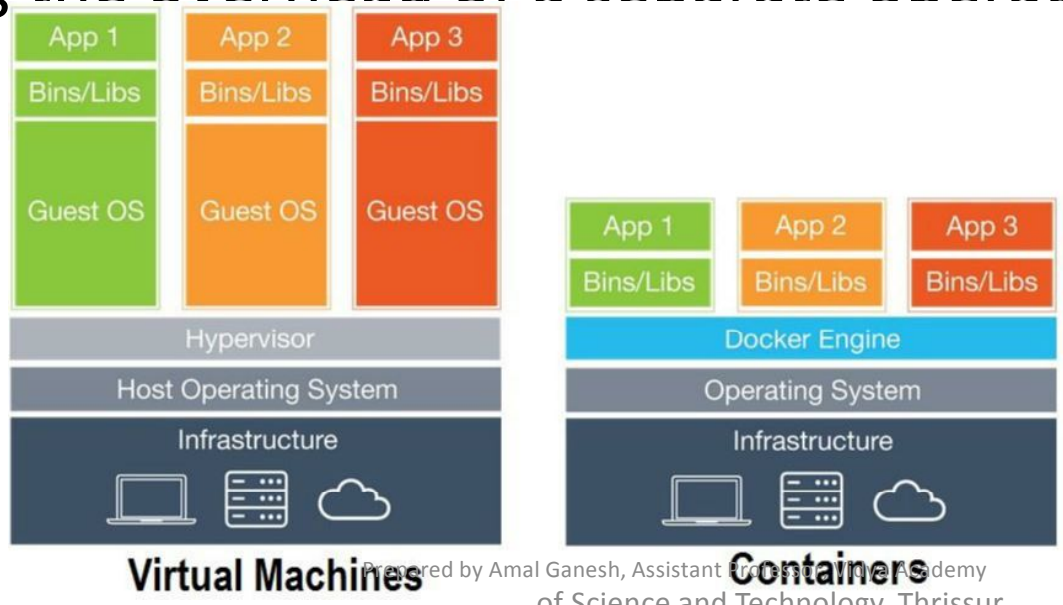
- This model was working fine and works fine even today.
- With the popularity of world wide web and the simplicity of the model, lead to the evolution of Application Programming Interface (API) based communication.
- The features like security (authentication and authorization), caching, throttling and monitoring type capabilities were needed to implement on top of the standard API implementations.
- Instead of implementing these capabilities at each and every API separately, there came the requirement to have a common component to apply these features on top of the API.
- This requirement leads the API management platform evolution and today it has become one of the core features of any distributed system.

Evolution of Distributed Computing System

- Then came the big bang moment of distributed systems where Internet-based companies like Facebook, Google, Amazon, Netflix, LinkedIn, Twitter became so large that they wanted to build distributed systems which span across multiple geographies and multiple data centres.
- Engineers started thinking about the concept of a single computer and single program. Instead of considering one computer as a one computer, they think about a way to create multiple virtual computers within the same machine.
- This leads to the idea of virtual machines where same computer can act as multiple computers and run them all in parallel.
- Even though this was a good enough idea, it was not the best option when it comes to resource utilization of the host computer.
- Running multiple operating systems required additional resources which was not required when running in the same operating system.

Evolution of Distributed Computing System

- This lead to the idea of containers where running multiple programs and their required dependencies on separate runtime using the same host operating system kernel (Linux).
- This concept was available with the Linux operating system for some time, it became more popular and improved a lot with the introduction of container-based application deployment. Containers can act as same as virtual machines without having the overhead of a separate operating system.



Challenges/Issues in design of distributed System

1. Heterogenity

- The Internet enables users to access services and run applications over a heterogeneous collection of computers and networks. Heterogeneity (that is, variety and difference) applies to all of the following:
- Hardware devices: computers, tablets, mobile phones, embedded devices, etc.
- Operating System: Ms Windows, Linux, Mac, Unix, etc.
- Network: Local network, the Internet, wireless network, satellite links, etc.
- Programming languages: Java, C/C++, Python, PHP, etc.
- Different roles of software developers, designers, system managers

and data structures such as arrays and records.

- Different programming languages use different representations for characters
- These differences must be addressed if programs written in different languages are to be able to communicate with one another.
- Programs written by different developers cannot communicate with one another unless they use common standards,
- for example, for network communication and the representation of primitive data items and data structures in messages. For this to happen, standards need to be agreed and adopted – as have the Internet protocols.

Challenges/Issues in design of distributed System

- **Middleware :**
  - The term 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.
  - Most middleware is implemented over the Internet protocols, which themselves mask the differences of the underlying networks, but all middleware deals with the differences in operating systems and hardware
- **Heterogeneity and mobile code :**
  - The term mobile code is used to refer to program code that can be transferred from one computer to another and run at the destination – Java applets are an example.
  - Code suitable for running on one computer is not necessarily suitable for running on another because executable programs are normally specific both to the instruction set and to the host operating system.

Challenges/Issues in design of distributed System

2. Transparency

- Transparency is defined as the concealment from the user and the application programmer of the separation of components in a distributed system, so that the system is perceived as a whole rather than as a collection of independent components.
- In other words, distributed systems designers must hide the complexity of the systems as much as they can.

- Some terms of transparency in distributed systems are:
  - **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 may be copied in several places
  - **Concurrency** Hide that a resource may be shared by several competitive users
  - **Failure** Hide the failure and recovery of a resource
  - **Persistence** Hide whether a (software) resource is in memory or a disk

Challenges/Issues in design of distributed System

3. Openness

- The openness of a computer system is the characteristic that determines whether the system can be extended and re-implemented in various ways.
- The openness of distributed systems 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.
- In other words, distributed systems designers must hide the complexity of the systems as much as they can.

- Open systems are characterized by the fact that their key interfaces are published.
- Open distributed systems are based on the provision of a uniform communication mechanism and published interfaces for access to shared resources.
- Open distributed systems can be constructed from heterogeneous hardware and software, possibly from different vendors. But the conformance of each component to the published standard must be carefully tested and verified if the system is to work correctly.

Challenges/Issues in design of distributed System

4. Security

- Many of the information resources that are made available and maintained in distributed systems have a high intrinsic value to their users.
- Their security is therefore of considerable importance.
- Security for information resources has three components:
  - Confidentiality (protection against disclosure to unauthorized individuals),
  - Integrity (protection against alteration or corruption),
  - Availability (protection against interference with the means to access the resources).
- Examples:
  - Denial of service attacks
  - Security of mobile code

Challenges/Issues in design of distributed System

5. Scalability

- Distributed systems operate effectively and efficiently at many different scales, ranging from a small intranet to the Internet.
- A 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.

- The design of scalable distributed systems presents the following challenges:
  - *Controlling the cost of physical resources*: As the demand for a resource grows, it should be possible to extend the system, at reasonable cost, to meet it.
  - *Controlling the performance loss*: Consider the management of a set of data whose size is proportional to the number of users or resources in the system
  - *Preventing software resources running out*: IPv4 to IPv6

Challenges/Issues in design of distributed System

6. Failure handling

- Computer systems sometimes fail.
- When faults occur in hardware or software, programs may produce incorrect results or may stop before they have completed the intended computation.
- Failures in a distributed system are partial – that is, some components fail while others continue to function.
- Therefore the handling of failures is particularly difficult.

- The following techniques for dealing with failures:
  - *Detecting failures*: Example: checksum
  - *Masking failures*: Some failures that have been detected can be hidden or made less severe. Two examples of hiding failures:
    1. Messages can be retransmitted when they fail to arrive.
    2. File data can be written to a pair of disks so that if one is corrupted, the other may still be correct.
  - *Tolerating failures*:
  - *Recovery from failures*: Recovery involves the design of software so that the state of permanent data can be recovered or 'rolled back' after a server has crashed.
- *Redundancy*: Services can be made to tolerate failures by the use of redundant components. For eg:- In the Domain Name System, every name table is replicated in at least two different servers.

Challenges/Issues in design of distributed System

7. Concurrency

- Both services and applications provide resources that can be shared by clients in a distributed system.
- There is therefore a possibility that several clients will attempt to access a shared resource at the same time.
- For example, a data structure that records bids for an auction may be accessed very frequently when it gets close to the deadline time.
- For an object to be safe in a concurrent environment, its operations must be synchronized in such a way that its data remains consistent.
- This can be achieved by standard techniques such as semaphores, which are used in most operating systems.

Challenges/Issues in design of distributed System

8. Quality of Service

- Once users are provided with the functionality that they require of a service, such as the file service in a distributed system, we can go on to ask about the quality of the service provided.
- The main non-functional properties of systems that affect the quality of the service experienced by clients and users are reliability, security and performance.
- Adaptability to meet changing system configurations and resource availability has been recognized as a further important aspect of service quality.

9. Performance

- Always the hidden data in the background is the issue of performance.
- Building a transparent, flexible, reliable distributed system, more important lies in its performance.
- In particular, when running a particular application on a distributed system, it should not be appreciably worse than running the same application on a single processor. Unfortunately, achieving this is easier said than done.

Examples of distributed systems

Finance and commerce	eCommerce e.g. Amazon and eBay, PayPal, online banking and trading
The information society	Web information and search engines, ebooks, Wikipedia; social networking: Facebook and MySpace
Creative industries and entertainment	online gaming, music and film in the home, user-generated content, e.g. YouTube, Flickr
Healthcare	health informatics, on online patient records, monitoring patients
Education	e-learning, virtual learning environments; distance learning
Transport and logistics	GPS in route finding systems, map services: Google Maps, Google Earth
Science	The Grid as an enabling technology for collaboration between scientists
Environmental management	sensor technology to monitor earthquakes, floods or tsunamis

Examples of distributed systems

1.2.1 Web search

An example: Google

Highlights of this infrastructure:

- physical infrastructure
- distributed file system
- structured distributed storage system
- lock service
- programming model

1.2.2 Massively multiplayer online games (MMOGs)

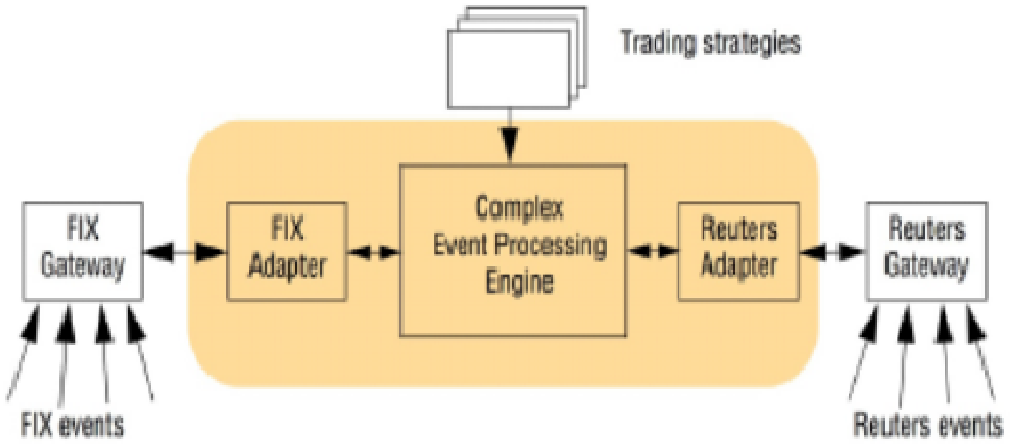
Examples

- EVE online – *client-server architecture*!
- EverQuest – more distributed architecture
- Research on completely decentralized approaches based on *peer-to-peer (P2P) technology*

Examples of distributed systems

1.2.3 Financlial trading

- *distributed even-based systems*




```
WHEN
        MSFT price moves outside 2% of MSFT Moving
              Average
FOLLOWED-BY (
        MyBasket moves up by 0.5%
        AND (
                HPQ–s price moves up by 5%
                OR
                MSFT–s price moves down by 2%
        )
)
ALL WITHIN
        any 2 minute time period
THEN
        BUY MSFT
        SELL HPQ
```

- Reuters market data events

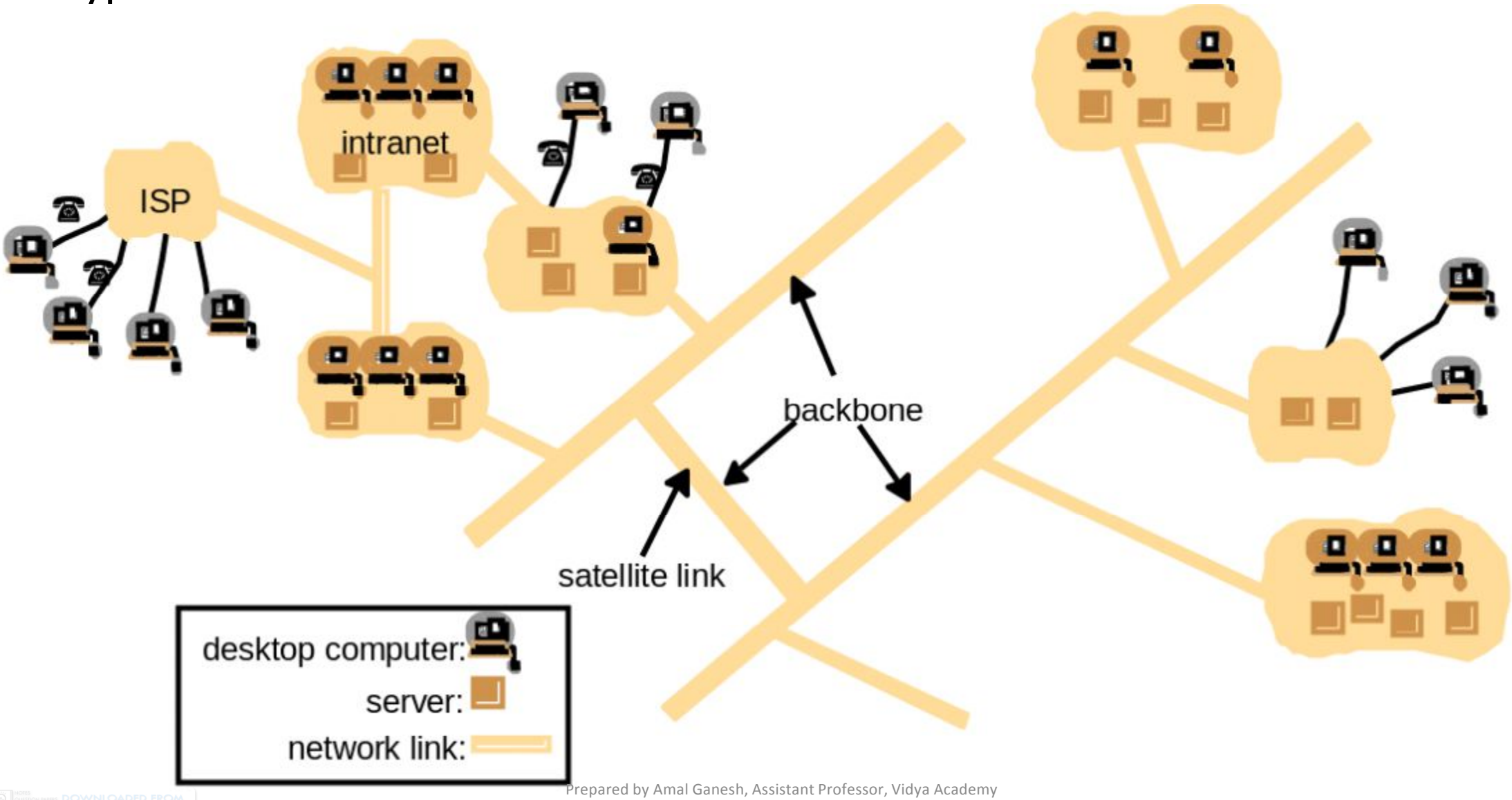
- FIX events (events following the specific format of the Financial Information eXchange protocol)

Trends in distributed System

- Distributed systems are undergoing a period of significant change and this can be traced back to a number of influential trends:
  - the emergence of pervasive networking technology;
  - the emergence of ubiquitous computing coupled with the desire to support user mobility in distributed systems;
  - the increasing demand for multimedia services;
  - the view of distributed systems as a utility.
- **Pervasive networking and the modern Internet:**
  - The modern Internet is a vast interconnected collection of computer networks of many different types, with the range of types increasing all the time and now including, for example, a wide range of wireless communication technologies such as WiFi, WiMAX, Bluetooth and third-generation mobile phone networks.
  - The net result is that networking has become a pervasive resource and devices can be connected (if desired) at any time and in any place.
  - The Internet is also a very large distributed system. It enables users, wherever they are, to make use of services such as the World Wide Web, email and file transfer.

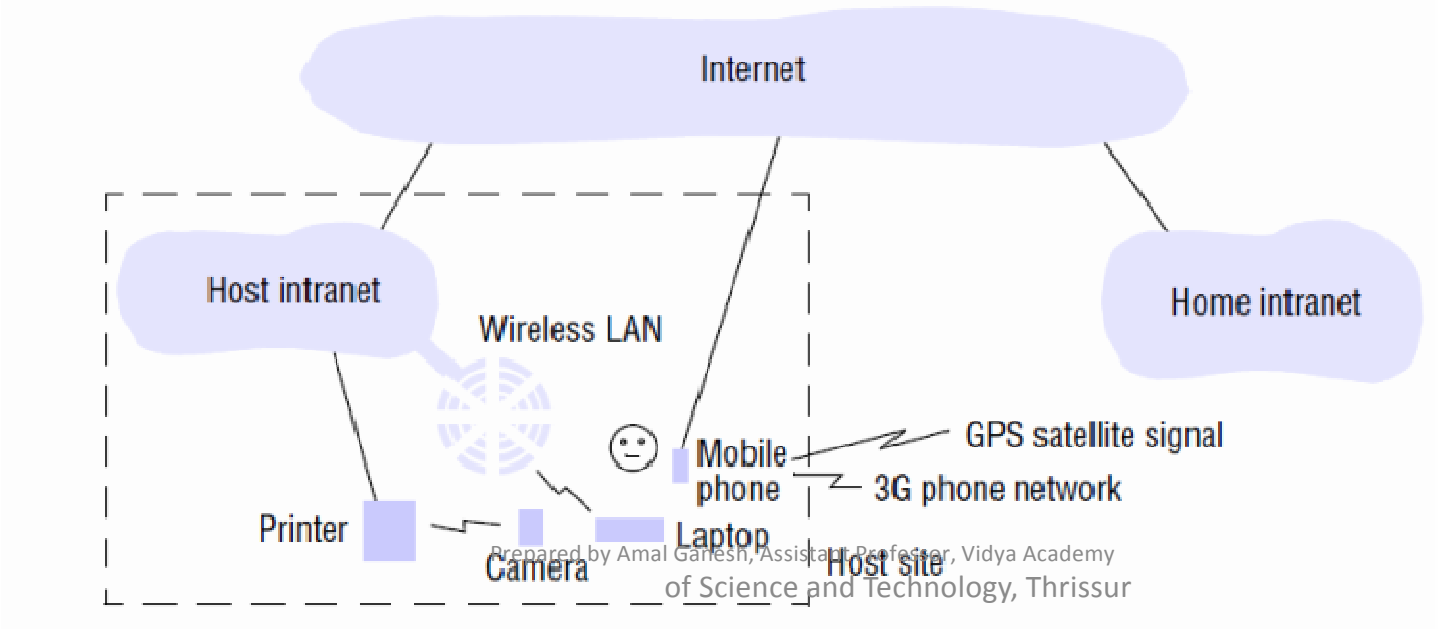
Trends in distributed System

- A Typical Portion of Internet:



Trends in distributed System

- Mobile and ubiquitous computing:
  - Technological advances in device miniaturization and wireless networking have led increasingly to the integration of small and portable computing devices into distributed systems. These devices include:
    - laptop computers
    - handheld devices (mobile phones, smart phones, tablets, GPS-enabled devices, PDAs, video and digital cameras)
    - wearable devices (smart watches, glasses, etc.)
    - devices embedded in appliances (washing machines, refrigerators, cars, etc.)



Trends in distributed System

- Mobile Computing
  - The portability of many of these devices, together with their ability to connect conveniently to networks in different places, makes *mobile computing* possible.
  - Mobile computing is the performance of computing tasks while the user is on the move, or visiting places other than their usual environment.
  - In mobile computing, users who are away from their 'home' intranet (the intranet at work, or their residence) are still provided with access to resources via the devices they carry with them.
  - They can continue to access the Internet; they can continue to access resources in their home intranet; and there is increasing provision for users to utilize resources such as printers or even sales points that are conveniently nearby as they move around.
  - The latter is also known as *location-aware* or *context-aware computing*.
  - Mobility introduces a number of challenges for distributed systems, including the need to deal with variable connectivity and indeed disconnection, and the need to maintain operation in the face of device mobility.

Trends in distributed System

- Ubiquitous computing
  - *Ubiquitous computing* is the harnessing of many small, cheap computational devices that are present in users' physical environments, including the home, office and even natural settings.
  - The term 'ubiquitous' is intended to suggest that small computing devices will eventually become so pervasive in everyday objects that they are scarcely noticed.
  - That is, their computational behaviour will be transparently and intimately tied up with their physical function.
  - Example – IOT, Home automation

Trends in distributed System

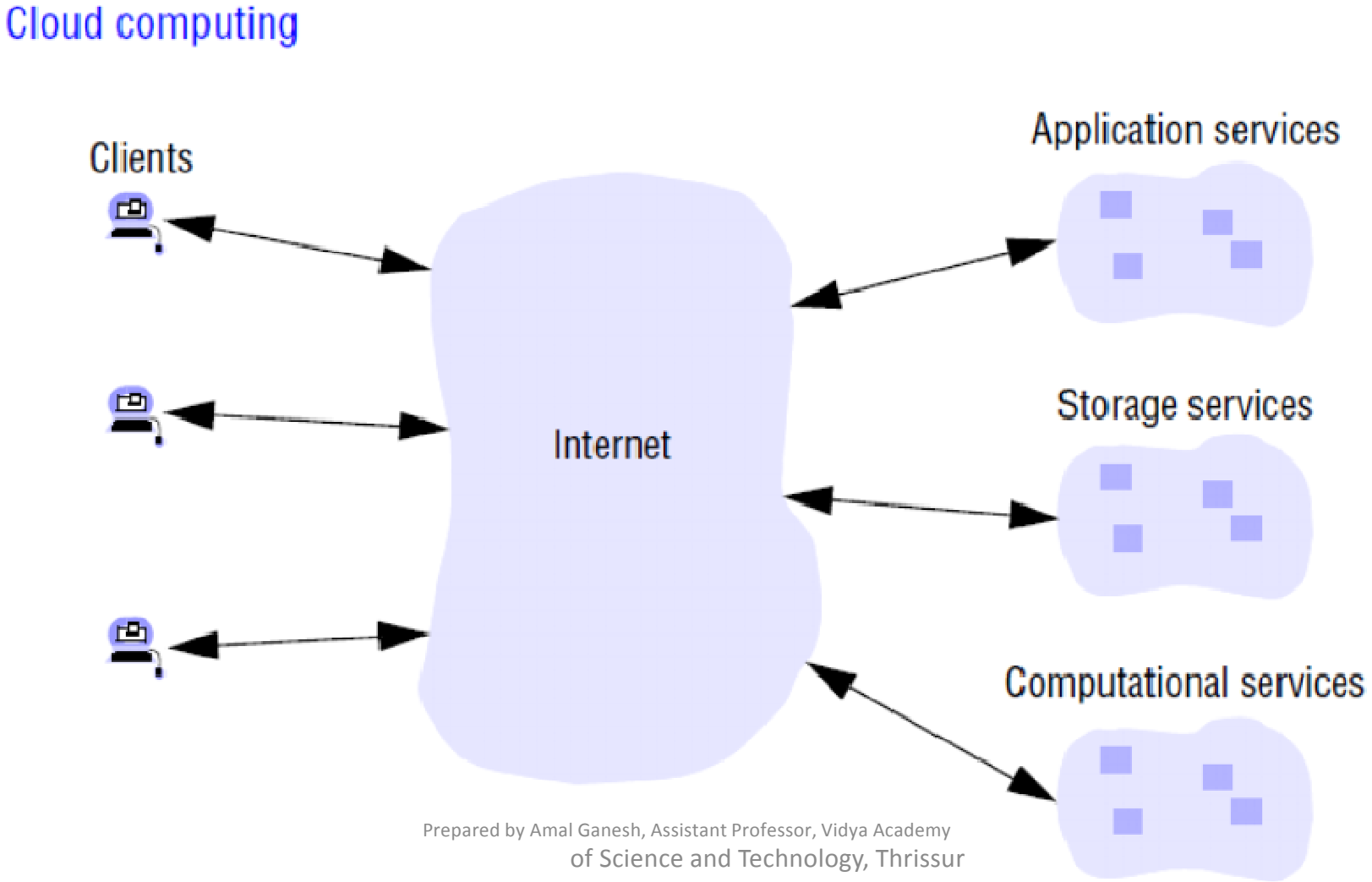
- Distributed multimedia systems
  - Another important trend is the requirement to support multimedia services in distributed systems.
  - Multimedia support can usefully be defined as the ability to support a range of media types in an integrated manner.
  - One can expect a distributed system to support the storage, transmission and presentation of what are often referred to as discrete media types, such as pictures or text messages.
  - A distributed multimedia system should be able to perform the same functions for continuous media types such as audio and video; that is, it should be able to store and locate audio or video files, to transmit them across the network (possibly in real time as the streams emerge from a video camera), to support the presentation of the media types to the user and optionally also to share the media types across a group of users
  - Examples: access to live or pre-recorded television broadcasts, access to film libraries offering video-on-demand services, access to music libraries, the provision of audio and video conferencing facilities and integrated telephony features including IP telephony or related technologies such as Skype, Webcasting

Trends in distributed System

- Distributed computing as a utility
  - With the increasing maturity of distributed systems infrastructure, a number of companies are promoting the view of distributed resources as a commodity or utility.
  - With this model, resources are provided by appropriate service suppliers and effectively rented rather than owned by the end user.
  - This model applies to both physical resources and more logical services:
  - Physical resources such as storage and processing can be made available to networked computers, removing the need to own such resources on their own.
  - Software services can also be made available across the global Internet using this approach. Example: Google Apps.
- Cloud Computing
  - The term *cloud computing* is used to capture this vision of computing as a utility.
  - A cloud is defined as a set of Internet-based application, storage and computing services sufficient to support most users' needs, thus enabling them to largely or totally dispense with local data storage and application software

Trends in distributed System

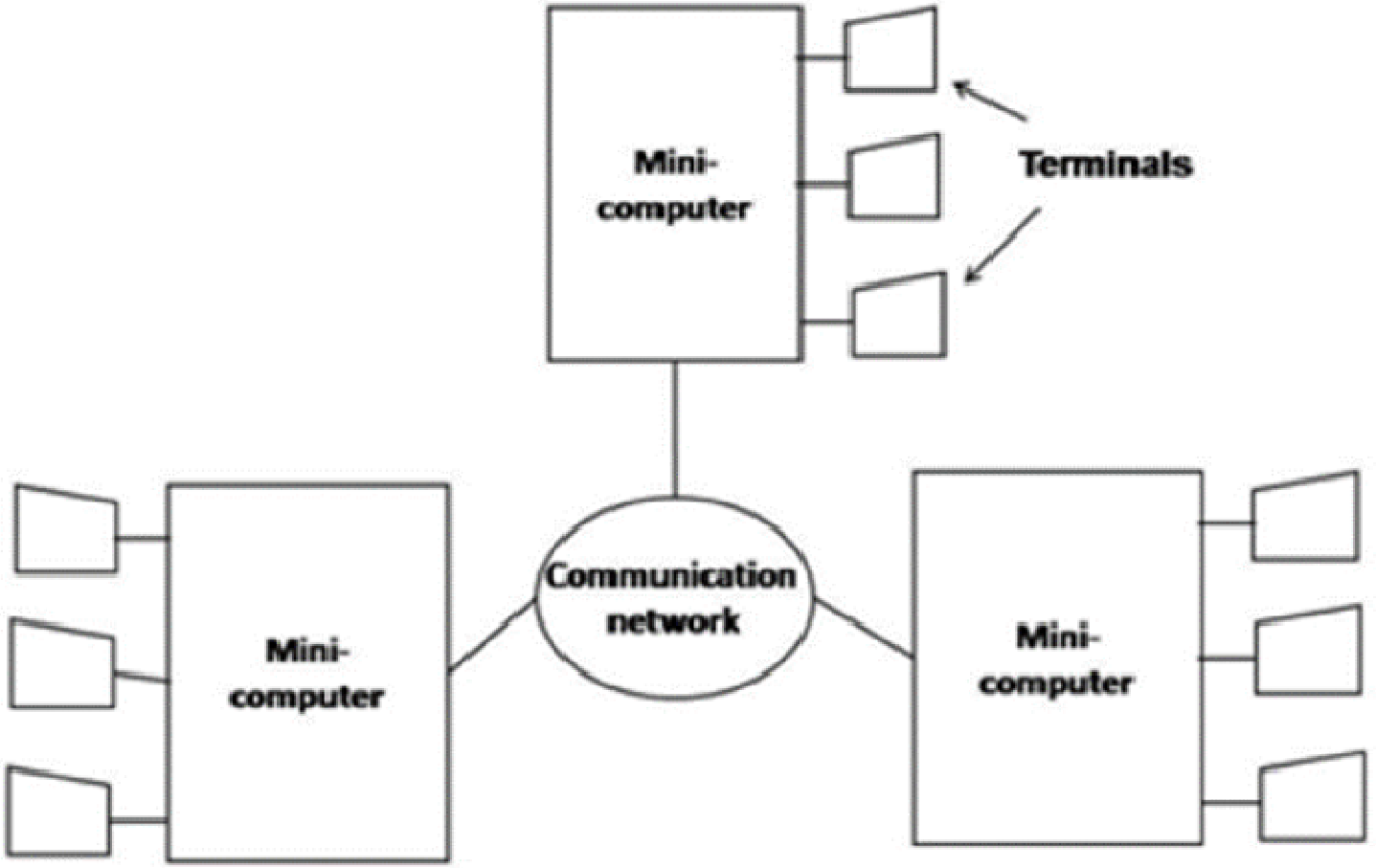
- Cloud computing also promotes a view of everything as a service, from physical or virtual infrastructure through to software, often paid for on a per-usage basis rather than purchased.



Distributed Computing System Models

- The various models that are used for building distributed computing systems can be classified into 5 categories:

1. Minicomputer Model:

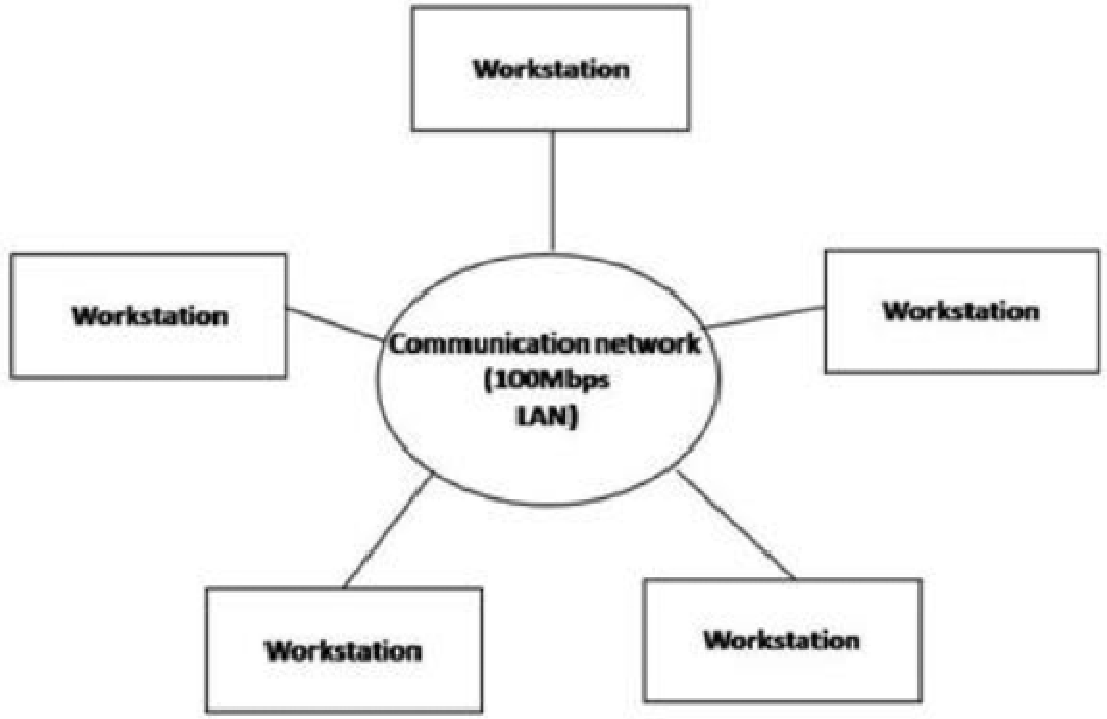


Distributed Computing System Models

- The minicomputer model is a simple extension of the centralized time-sharing system.
- A distributed computing system based on this model consists of a few minicomputers interconnected by a communication network were each minicomputer usually has multiple users simultaneously logged on to it.
- Several interactive terminals are connected to each minicomputer.
- Each user logged on to one specific minicomputer has remote access to other minicomputers.
- The network allows a user to access remote resources that are available on some machine other than the one on to which the user is currently logged.
- The minicomputer model may be used when resource sharing with remote users is desired.
- The early ARPA net is an example of a distributed computing system based on the minicomputer model.

Distributed Computing System Models

2. Workstation Model

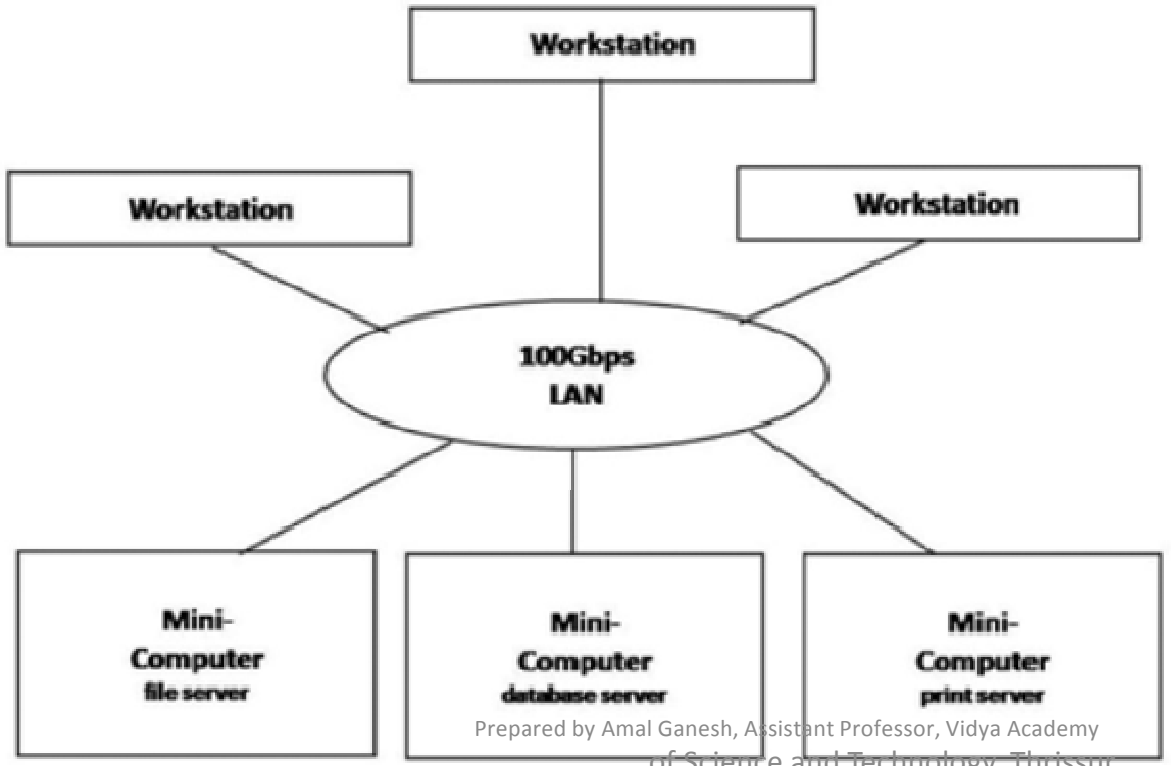


- A distributed computing system based on the workstation model consists of several workstations interconnected by a communication network.
- An organization may have several workstations located throughout an infrastructure were each workstation is equipped with its own disk & serves as a single-user computer.
- In such an environment,at any one time a significant proportion of the workstations are idle which results in the waste of large amounts of CPU time.

Distributed Computing System Models

- Therefore,the idea of the workstation model is to interconnect all these workstations by a high-speed LAN so that idle workstations may be used to process jobs of users who are logged onto other workstations & do not have sufficient processing power at their own workstations to get their jobs processed efficiently.
- Example:Sprite system & Xerox PARC

3. Workstation–Server Model

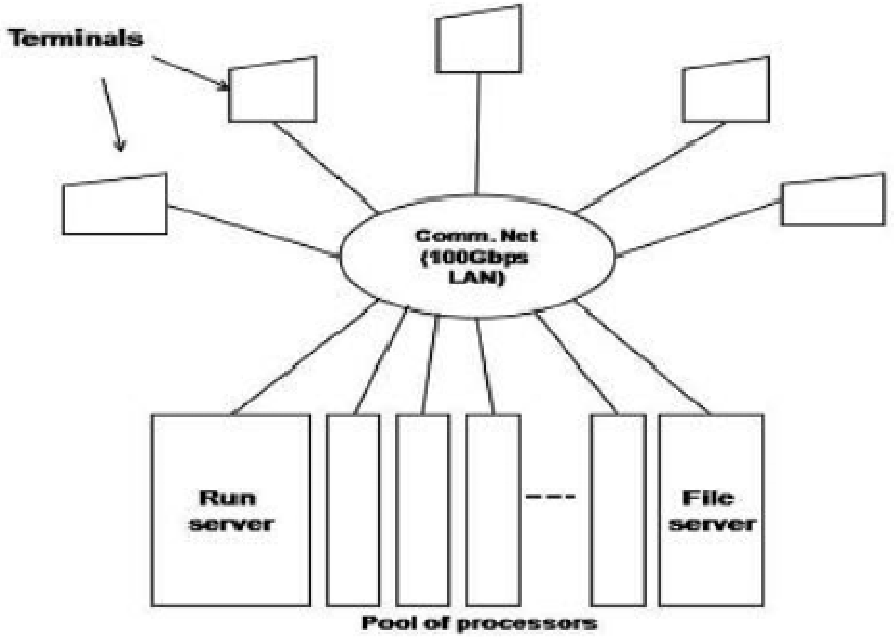


Distributed Computing System Models

- The workstation model is a network of personal workstations having its own disk & a local file system.
- A workstation with its own local disk is usually called a diskful workstation & a workstation without a local disk is called a diskless workstation. Diskless workstations have become more popular in network environments than diskful workstations, making the workstation-server model more popular than the workstation model for building distributed computing systems.
- A distributed computing system based on the workstation-server model consists of a few minicomputers & several workstations interconnected by a communication network.
- In this model, a user logs onto a workstation called his or her home workstation. Normal computation activities required by the user's processes are performed at the user's home workstation, but requests for services provided by special servers are sent to a server providing that type of service that performs the user's requested activity & returns the result of request processing to the user's workstation.
- Therefore, in this model, the user's processes need not migrated to the server machines for getting the work done by those machines.
- Example: The V-System.

Distributed Computing System Models

4. Processor–Pool Model:



- The processor-pool model is based on the observation that most of the time a user does not need any computing power but once in a while the user may need a very large amount of computing power for a short time.
- Therefore, unlike the workstation-server model in which a processor is allocated to each user, in processor-pool model the processors are pooled together to be shared by the users as needed.
- The pool of processors consists of a large number of microcomputers & minicomputers attached to the network.

Distributed Computing System Models

- Each processor in the pool has its own memory to load & run a system program or an application program of the distributed computing system.
- In this model no home machine is present & the user does not log onto any machine.
- This model has better utilization of processing power & greater flexibility.
- Example: Amoeba & the Cambridge Distributed Computing System.

5. Hybrid Model:

- The workstation-server model has a large number of computer users only performing simple interactive tasks &-executing small programs.
- In a working environment that has groups of users who often perform jobs needing massive computation, the processor-pool model is more attractive & suitable.
- To combine Advantages of workstation-server & processor-pool models, a hybrid model can be used to build a distributed system.
- The processors in the pool can be allocated dynamically for computations that are too large or require several computers for execution.
- The hybrid model gives guaranteed response to interactive jobs allowing them to be more processed in local workstations of the users.

END