Introduction to Distributed Systems

Introduction to Distributed Systems

- ✓ **Key concepts**
  - **What is Distributed System?**
  - **Fundamental issues in Distributed Systems**
  - **Distributed System models and Architectures**
  - **Design goals of Distributed Systems**
  - **Differences between client/server and D.S. environments**

What is distributed system?

- Consist of several computers that do not share a memory or a clock. {Excludes tightly coupled Systems.}
- Communicate with each other by message passing over a communication network.
- Each computer has its own memory & run its own operating systems.
- The design of Distributed Operating System is much more difficult due to the lack of both shared memory and a physical clock, and unpredictable communication delays.

What is distributed system?

- Definition
  - Distributed System consists of a collection of autonomous computers linked by a computer network and equipped with distributed system software.
  - Distributed System software enables computers to co-ordinate their activities & to share the resources of the system – hardware, software and data.
  - Users of a Distributed System should perceive a single, integrated computing facility even through it may be implemented by many computers in different locations.
  - Entails the dispersion of hardware, software & data to multiple computers

What is distributed system?

- ✓ DS definitions
  - A collection of autonomous computers linked by a computer network and supported by software that enables the collections to operate as an <u>integrated facility.</u>
  - Example: ATM, WWW

- ✓ What is distributed?
  - Processing logic
  - Functions
  - Data
  - Control

Characteristics of Distributed Systems

- ✓ Multiple nodes working independently with each others. {NODE → CPU & MEMORY}
- ✓ The nodes do not have common shared memory.
- ✓ There exists an interconnection network
- ✓ Distribution transparency is provided
- ✓ There is no single point of failure. { fault tolerance}

Advantages of distributed systems over centralized systems

- **Economics:** a collection of microprocessors offer a better price/performance than mainframes. Low price/performance ratio: cost effective way to increase computing power.
- **Speed:** a distributed system may have more total computing power than a mainframe. Ex. 10,000 CPU chips, each running at 50 MIPS. Not possible to build 500,000 MIPS single processor since it would require 0.002 nsec instruction cycle. Enhanced performance through load distributing.
- **Inherent distribution:** Some applications are inherently distributed. Ex. a supermarket chain.

Advantages of distributed systems over centralized systems

- **Reliability**: **If one machine crashes, the system as a whole can still survive. Higher availability and improved reliability**.
- **Incremental growth: Computing power can be added in small increments. Modular expandability**
- **Another deriving force: the existence of large number of personal computers, the need for people to collaborate and share information.**

Advantages of distributed systems over independent PCs

- **Data sharing:**
  - allow many users to access to a common data base
- **Resource Sharing:**
  - expensive peripherals like color printers
- **Communication:**
  - enhance human-to-human communication
  - e.g., email, chat
- **Flexibility:**
  - spread the workload over the available machines

Disadvantages of Distributed Systems

- **Software:**
  - difficult to develop software for distributed systems
- **Network:**
  - saturation, loss```y transmissions
- **Security:**
  - easy access also applies to secret data

Significant Consequences of Distributed Systems

- ✓ **Concurrency**
  - **The capacity of the system to handle shared resources can be increased by adding more resources (for example Computers) to the network.**
- ✓ **No global clock**
  - **There is no single global notion of the correct time. The only communication is by sending messages through a network.**
- ✓ **Independent failures**
  - **In DS, faults in the network result in the isolation of the computers that are connected to it.**
  - **The programs may not be able to detect whether the network has failed or has become unusually slow.**

Distributed Systems Challenges

- ✓ Heterogeneity
- ✓ Openness
- ✓ Security
- ✓ Scalability
- ✓ Failure handling
- ✓ Concurrency
- ✓ Transparency

Heterogeneity

"Hello, Bob? It's your father again. I have another question about my new computer. Can I tape a movie from cable TV then fax it from my VCR to my CD-ROM then E-mail it to my brother's cellular phone so he can make a copy on his neighbor's camcorder?"

Heterogeneity

- ✓ Different networks, hardware, operating systems, programming languages, developers.
- ✓ Need to set up protocols to solve these heterogeneities.
- ✓ Middleware: a software layer that provides a programming abstraction as well as masking the heterogeneity.
- ✓ Mobile code: code that can be sent from one computer to another and run at the destination.

Openness

- ✓ The openness of DS 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.
- ✓ Open systems are characterized by the fact that their key interfaces are published.
- ✓ Open DS are based on the provision of a uniform communication mechanism and published interfaces for access to shared resources.
- ✓ Open DS can be constrcted from heterogeneous hardware and software.

Security

- 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.
- Two new security challenges:
  - Denial of service attacks (DoS).
  - Security of mobile code.

Scalability

- A system is described as scalable if it remains effective when there is a significant increase in the number of resources and the number of users.
- Challenges:
  - Controlling the cost of resources.
  - Controlling the performance loss.
  - Preventing software resources from running out
  - Avoiding preformance bottlenecks.

Failure handling

- ✓ When faults occur in hardware or software, programs may produce incorrect results or they may stop before they have completed the intended computation.
- ✓ Techniques for dealing with failures:
  - Detecting failures
  - Masking failures
  - Tolerating failures
  - Recovering form failures
  - Redundancy

Concurrency

- ✓ There is a possibility that several clients will attempt to access a shared resource at the same time.
- ✓ Any object that represents a shared resource in a distributed system must be responsible for ensuring that it operates correctly in a concurrent environment.

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.
- Eight forms of transparency:
  - Access transparency
  - Location transparency
  - Concurrency transparency
  - Replication transparency
  - Failure transparency
  - Mobility transparency
  - Performance transparency
  - Scaling transparency

Issues in designing distributed O.S.

- Transparency
  - Kinds of transparency
    - Access Transparency: local and remote resources are accessed in the same way.
    - Location Transparency: The location of objects is not known to the users.
      - Name transparency
      - User mobility
    - Replication (&Fragmentation) Transparency: enables multiple instances of resources to be used to increase reliability and performance without the knowledge of the replicas by the user or application programmer. Objects are accessed without knowledge about any possible fragmentations.

Issues in designing distributed O.S. (contd…)

- Concurrency transparency: Objects are accessed in the same way as in a single – user system.
- Error (Failure) Transparency: Applications terminate in a defined way even in the event of hardware or software failures.
- Migration(mobility) transparency: Objects can migrate without affecting applications.
- Performance transparency: the system automatically reconfigures to improve performance.
- Scaling transparency: The System can be extended in size without changes to its structures or the applications.

Issues in designing distributed O.S. (contd…)

- **Reliability**
  - **Methods for dealing with faults, and to detect and remove the faults:**
    - **Fault avoidance**
    - **Fault tolerance**
      - **Redundancy techniques**
      - **Distributed control**
    - **Fault detection and recovery**
      - **Atomic transactions**
      - **Stateless servers**
      - **Acknowledgments**

Issues in designing distributed O.S. (contd…)

- **Flexibility**
  - Design of a distributed operating system should be flexible due to following reasons
    - Ease of modification
    - Ease of enhancement
- **Performance**
  - Design principles
    - Batch if possible
    - Cache whenever possible

Issues in designing distributed O.S. (contd…)

  - Minimize copying of data
  - Minimize network traffic
  - Take advantage of fine-grain parallelism for multiprocessing
- ✓ Scalability
  - Avoid centralized entities
  - Avoid centralized algorithms
  - Perform most operations on client workstations
- ✓ Heterogeneity
- ✓ Security
- ✓ Emulation of existing OS

Distributed system models and architectures

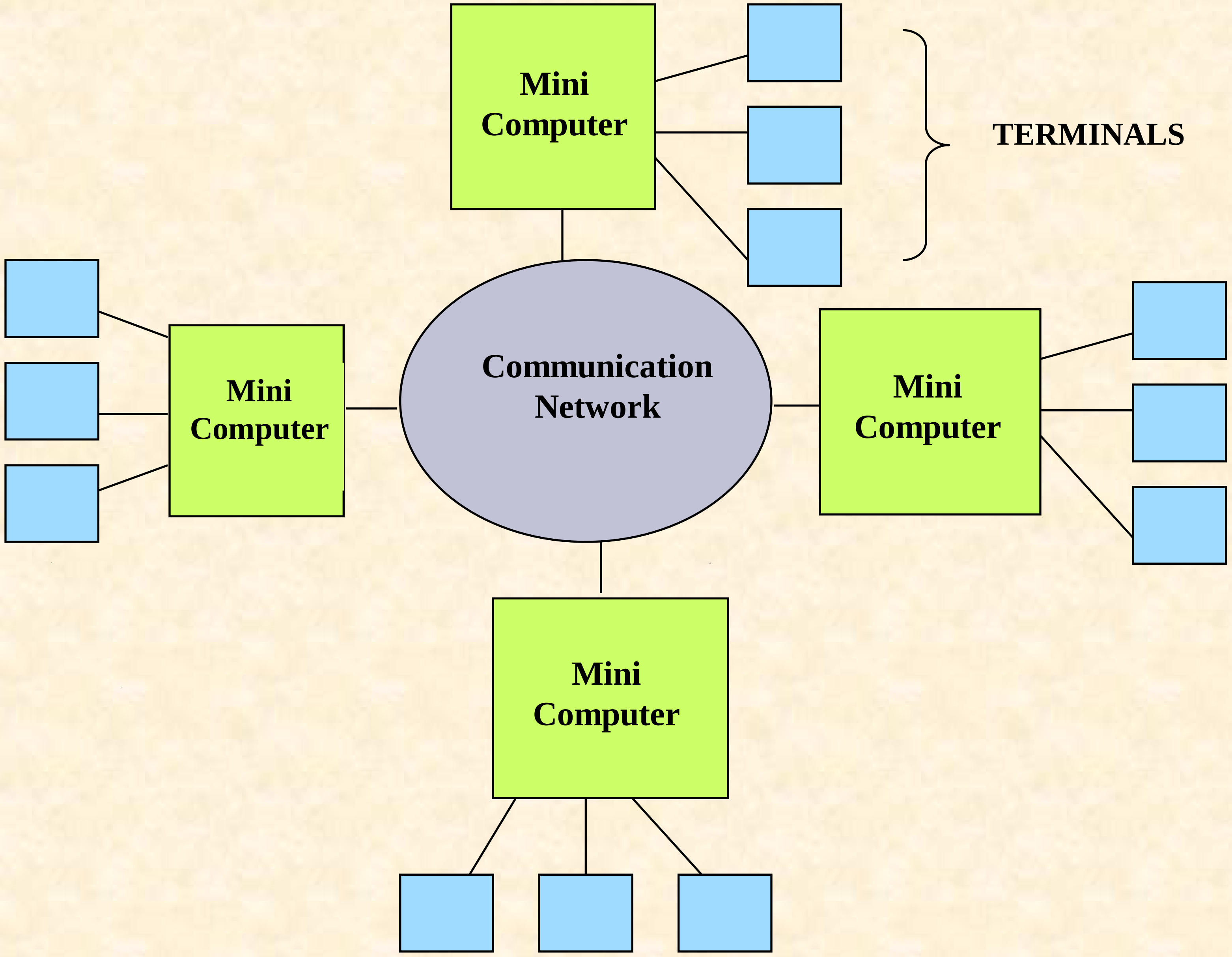
- ✓ **Models are classified into five categories**

1. **MINICOMPUTER MODEL**
2. **WORKSTATION MODEL**
3. **WORKSTATION-SERVER MODEL**
4. **PROCESSOR-POOL MODEL**
5. **HYBRID MODEL**

Minicomputer model

- Simple extension of centralized time sharing system
- May be used when resource sharing with remote users is desired
- Consists of few minicomputers interconnected
- Each minicomputer has multiple users simultaneously logged on to it through several interactive terminals connected to the minicomputer
- Each user logged on to one specific minicomputer, with remote access to other minicomputers and remote shared resources
- Example: ARPAnet

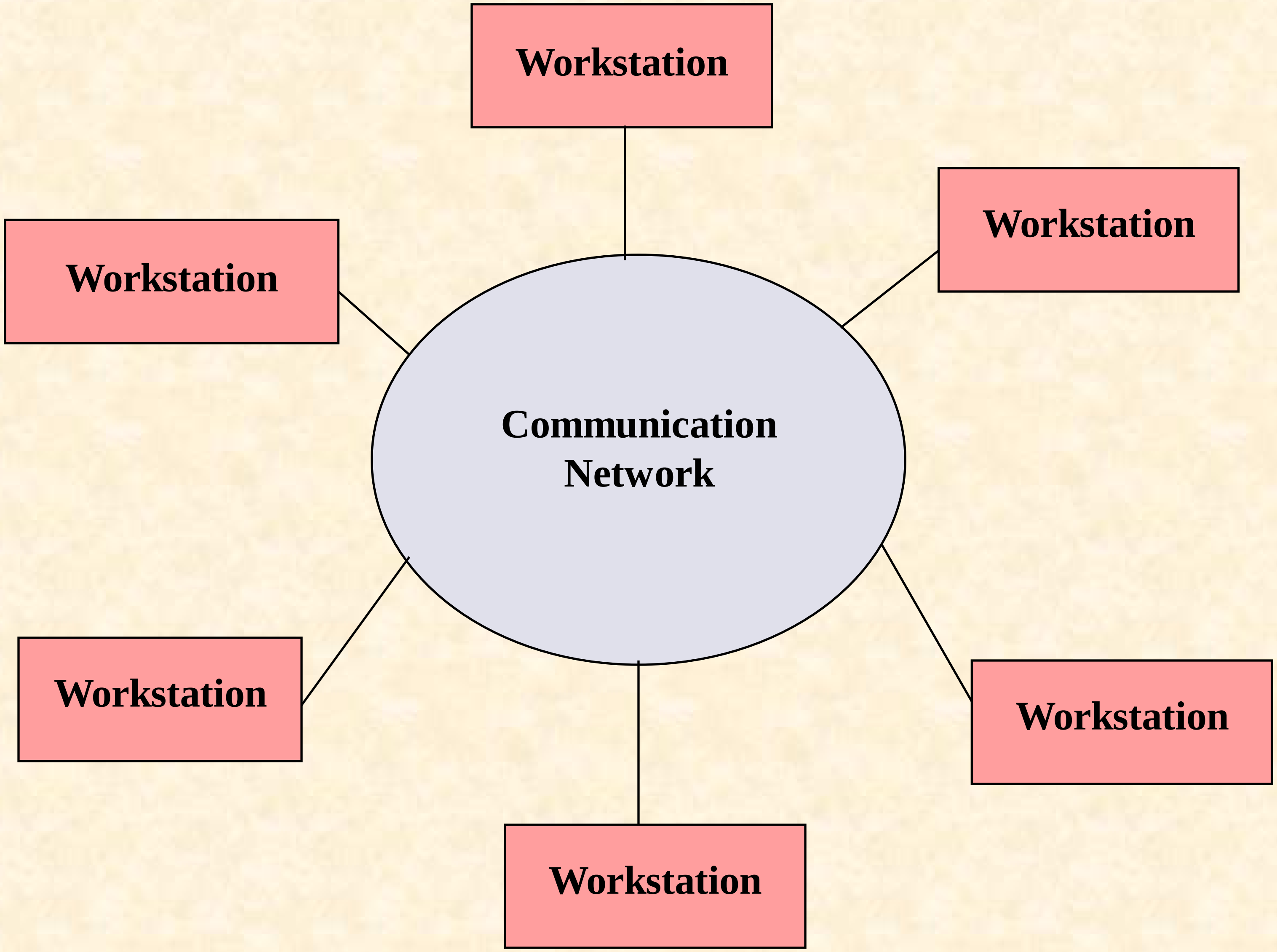
Minicomputer model



Workstation model

- Several workstations are interconnected by a communication network
- User logs onto one of the workstations (“home”) and submits jobs for execution.
- Each workstation is serving as a single user computer
- If workstation does not have sufficient power for execution
    - It transfers one or more processes from user’s workstation to some other workstation that is idle and gets work done

Workstation model



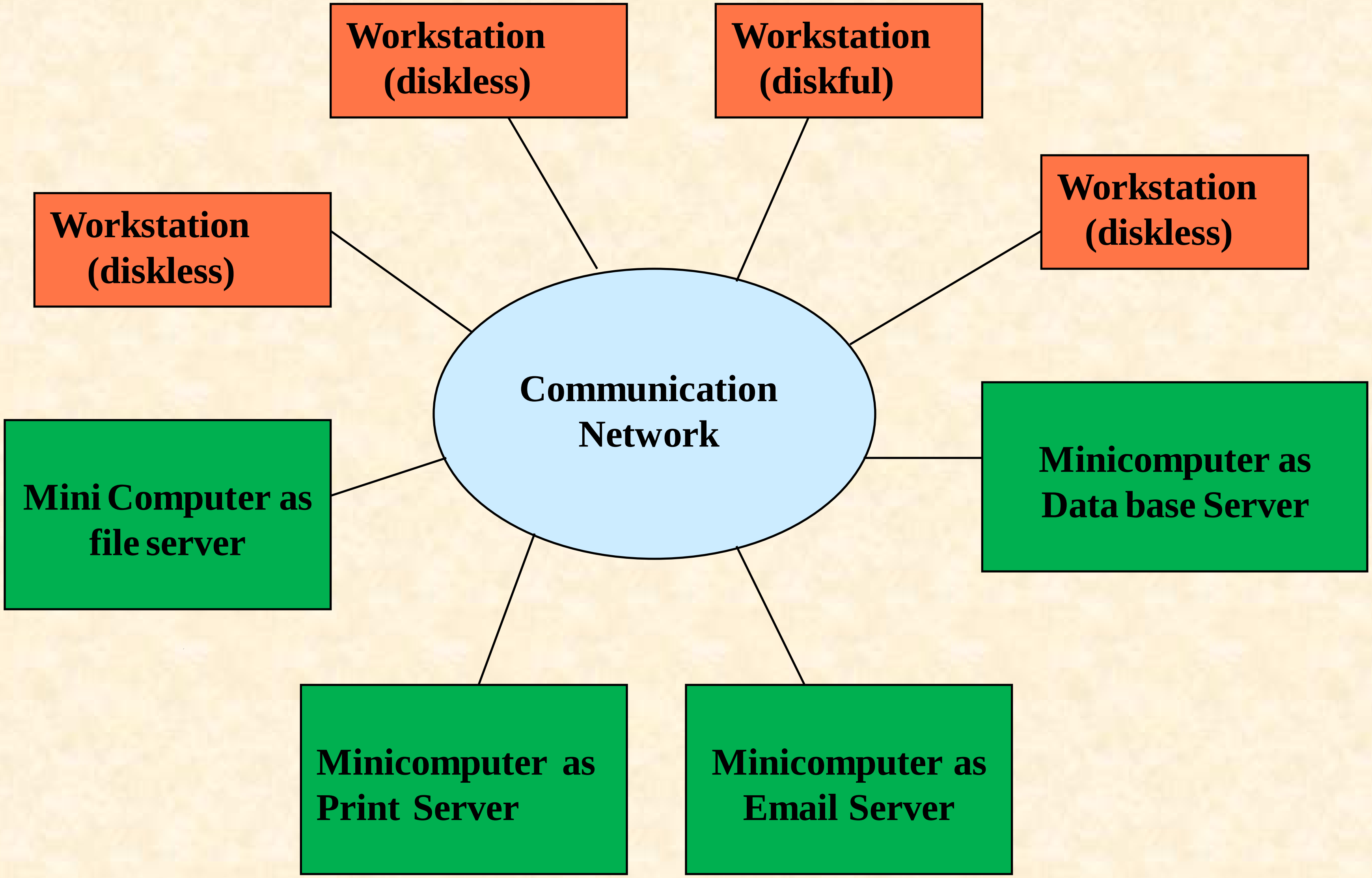
Workstation model(Contd…)

- Issues in Workstation model are
  - How does the system find an idle workstation?
  - How is a process transferred from one workstation to get it executed on another workstation?
  - What happens to a remote process if a user logs on a workstation that has idle until now and was being used to execute a process of another workstation?

Workstation-Server model

- Is a network of several personal workstations, each with its own disk and local file system and few minicomputers
- User logs onto a workstation called his/her home workstation
- Normal computing activities required by the user’s processes are performed at users’s workstation
- Requests for services provided by special servers are sent to servers providing that service

Workstation-server model


Workstation (diskless)
Workstation (diskful)
Workstation (diskless)
Workstation (diskless)
Communication Network
Mini Computer as file server
Minicomputer as Data base Server
Minicomputer as Print Server
Minicomputer as Email Server

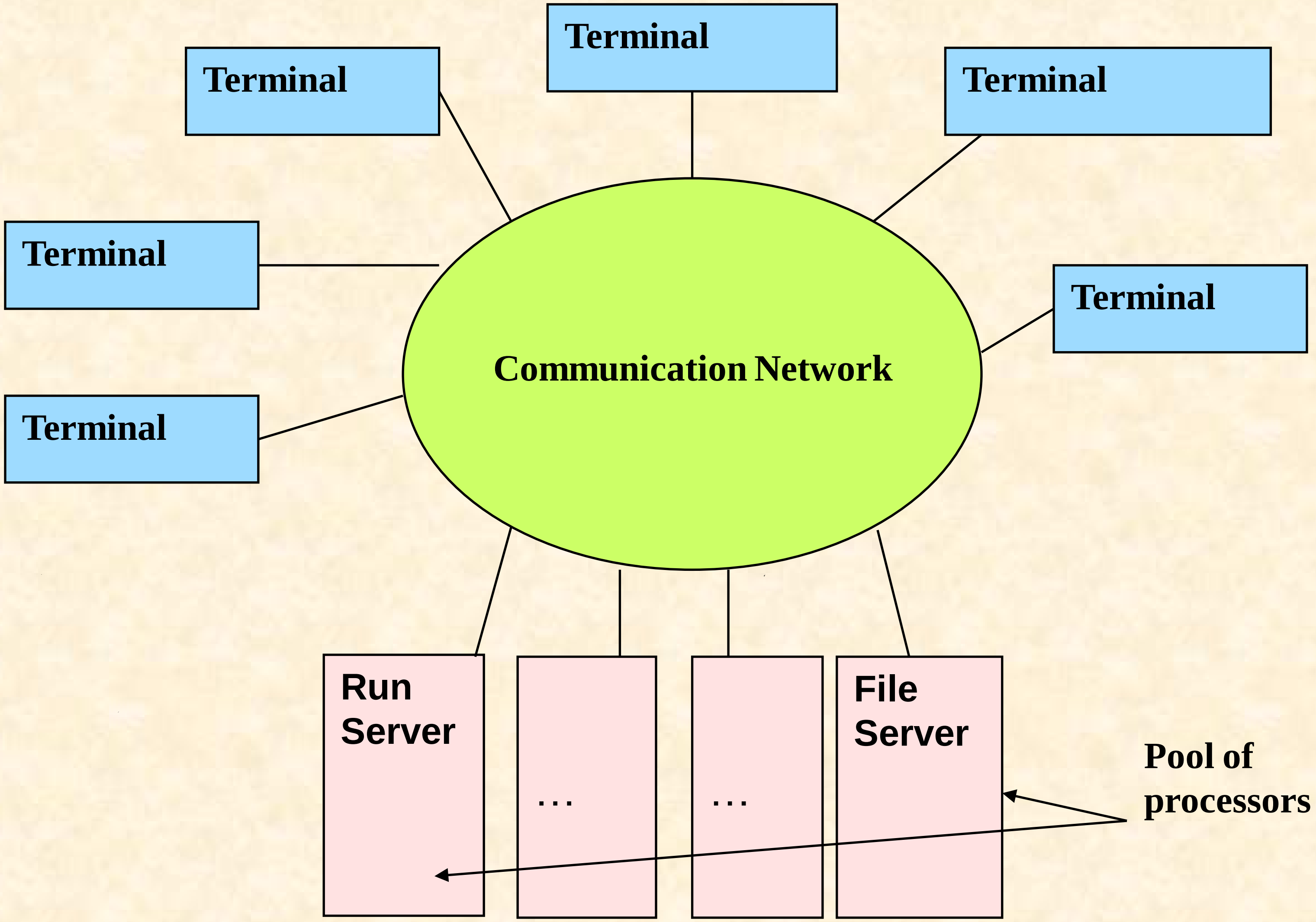
Workstation-Server vs Workstation model

- ✓ It is much cheaper – use of few minicomputers with large fast disks than large number of diskful workstations with small, slow disk
- ✓ Backup and hardware maintenance are easier to perform with a few large disks than many small disks scattered around
- ✓ Since all files are managed by the file servers, users have the flexibility to use any workstation and access the files
- ✓ Process migration facility is not needed here
- ✓ Model does not utilize the processing capability of idle workstations
- ✓ A user has guaranteed response time because workstations are not used for executing remote processes

Processor-pool model

- Processors are pooled together to be shared by the users as needed
- Each processor in the pool has its own memory to load and run a system program or an application program of distributed computing system
- No concept of Home computer
- Compared to workstation server model, this model allows better utilization of the available processing power of the distributed computing system

Processor-pool model



Hybrid Model

- ✓ Combines the advantages of both workstation-server and processor pool model
- ✓ Efficient execution of computation-intensive jobs
- ✓ Gives guaranteed response to interactive jobs by allowing them to be processed locally
- ✓ More expensive to build

Why are distributed systems gaining popularity?

- ✓ Inherently distributed applications
- ✓ Information sharing among distributed users
- ✓ Resource sharing
- ✓ Better price-performance ratio
- ✓ Shorter response times and higher throughput
- ✓ Higher reliability
- ✓ Extensibility and incremental growth
- ✓ Better flexibility in meeting user’s needs

End of the chapter **Introduction to Distributed Systems**