DISTRIBUTED SYSTEMS

Introduction to Distributed Systems

Learning Outcomes

- By the end of this chapter, the learner should be able to:
  - Describe the design issues of distributed system.
  - Discuss advantages and limitations of Distributed Computing Systems.
  - Describe organization of Distributed Computing Systems.
  - Describe the concept of Distributed Systems,

Introducti on

- Networks of computers are everywhere.
- The Internet is one, as are the many networks of which it is composed.

Introducti on

- Mobile phone networks, corporate networks, factory networks, campus networks, home networks, in-car networks –all of these, both separately and in combination, share the essential characteristics that make them relevant subjects for study under the heading distributed systems.

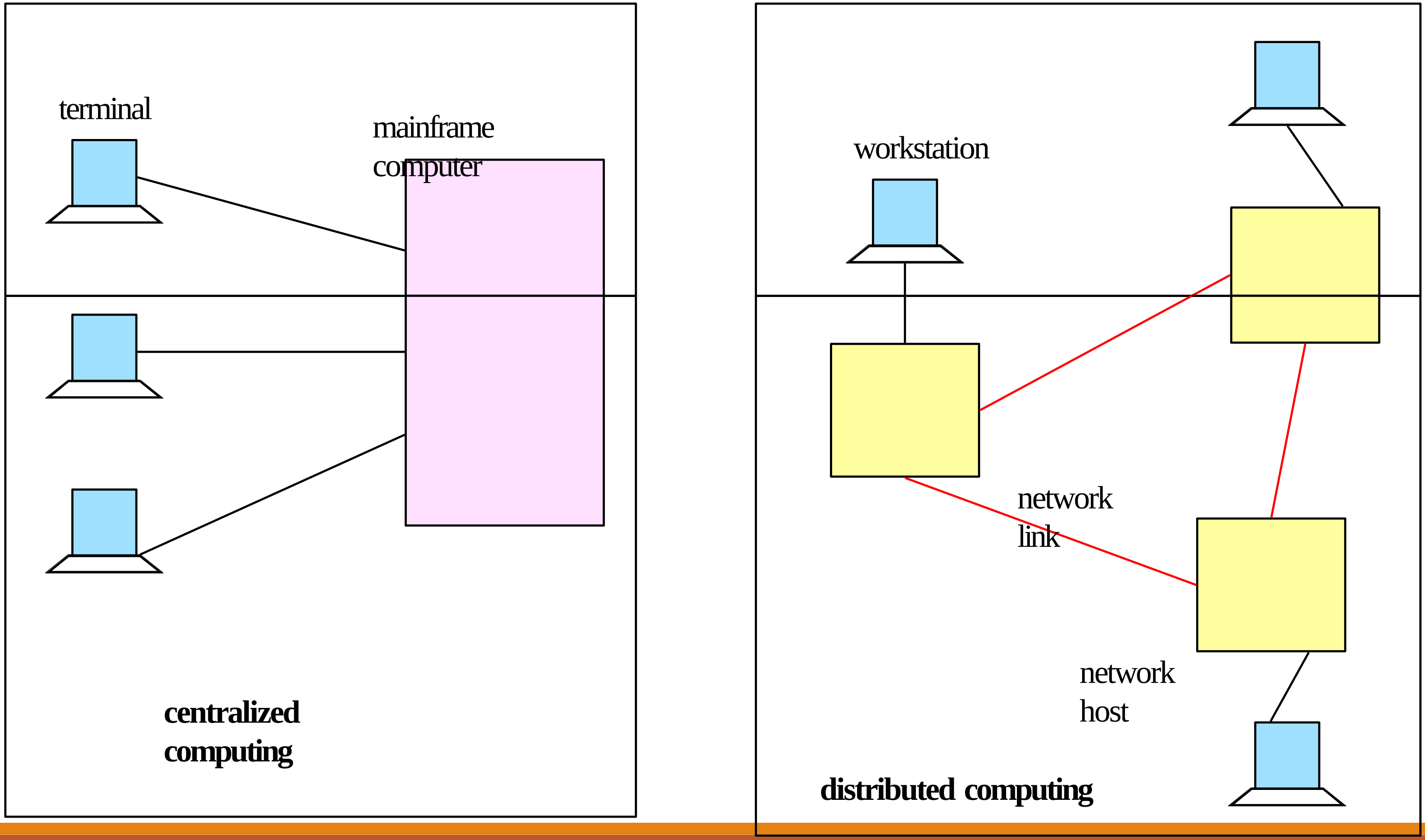
Centralized Systems

- Early computing was performed on a single processor.
- Uni-processor computing can be called *centralized computing*.
- Centralized systems have non-autonomous components

Centralized Systems

- Centralized systems are often built using homogeneous technology.
- Multiple users share the resources of a centralized system at all times.
- Centralized systems have a single point of control and of failure.

Centralized Systems



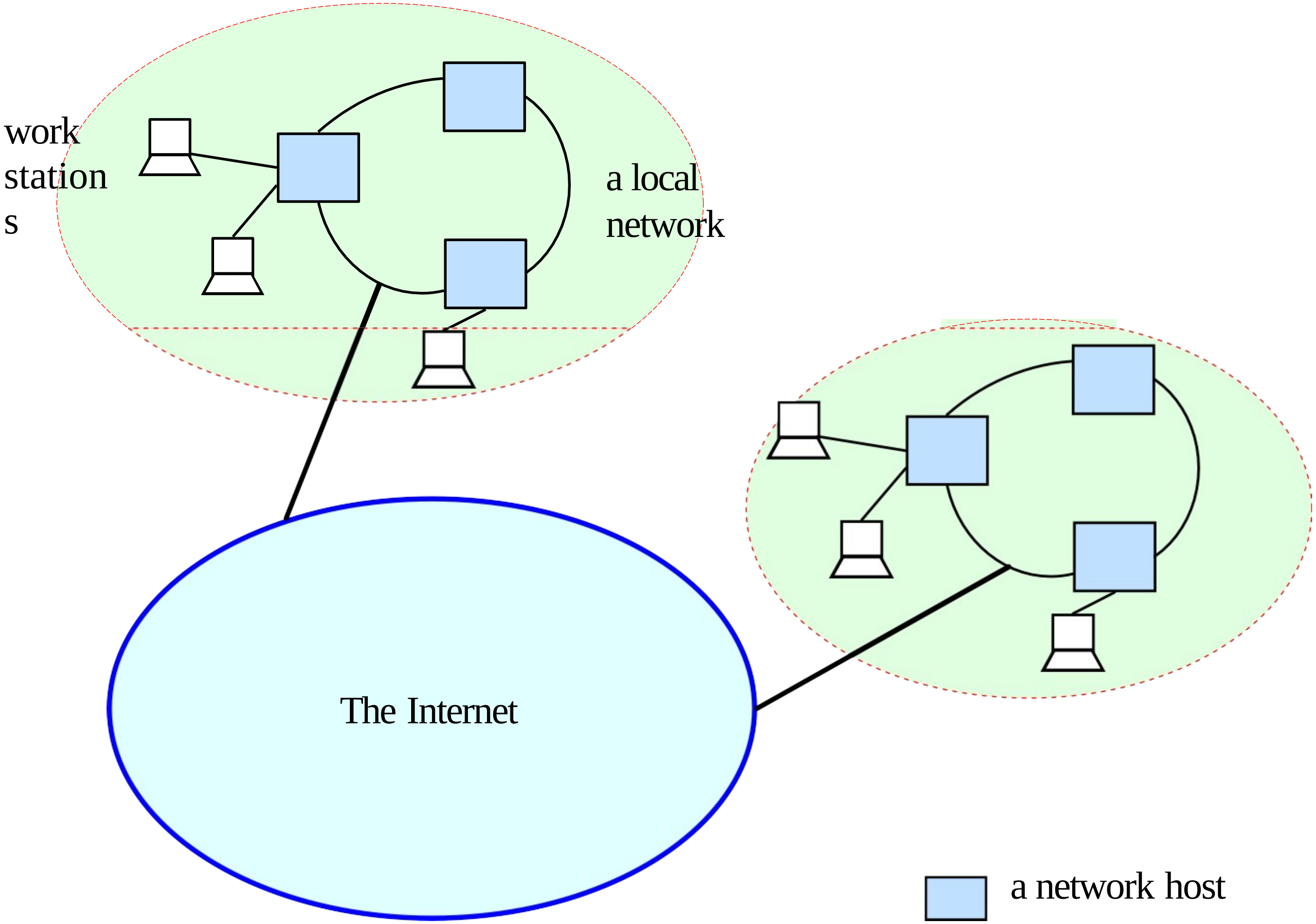
Distributed Systems

- A distributed system consists of a number of components, which are themselves computer systems.
- These components are connected by some communication medium, usually a sophisticated network.

Distributed Systems

- Applications execute by using a number of processes in different component systems.
- These processes communicate and interact to achieve productive work within the application.

Distributed Systems



Distributed System

- A distributed system consists of a collection of autonomous computers, connected through a network and distribution middleware, which enables computers to coordinate their activities and to share the resources of the system, so that users perceive the system as a single, integrated computing facility.

Distributed Systems

- This simple definition covers the entire range of systems in which networked computers can usefully be deployed.
- Computers that are connected by a network may be spatially separated by any distance.

Distributed Systems

- These computer networks may be on separate continents, in the same building or in the same room.
- Distributed computing is a method of processing in which different parts of a computer program are run simultaneously on two or more computers that are communicating with each other over a network.

Distributed Computing

- Distributed computing is a type of segmented or parallel computing, but parallel computing is most commonly used to refer to processing in which different parts of a program run simultaneously on two or more processors that are part of the same computer.

Distributed Computing

- While bothtypes of processing require that a program be segmented—divided into sections that can run simultaneously, distributed computi ng also requires the program that the division of take into account the different environments on which the different sections of the program will be running.

Distributed Computing

- For example, comput are likely two have ers to and different hardware syste different
- Distributed computing is a result natural of using networks to computers to enable communicate efficiently.

Distributed Computing

- Computer networking or fragmented computing refers to two or more computers interacting with each other, but not, typically, sharing the processing of a single program.

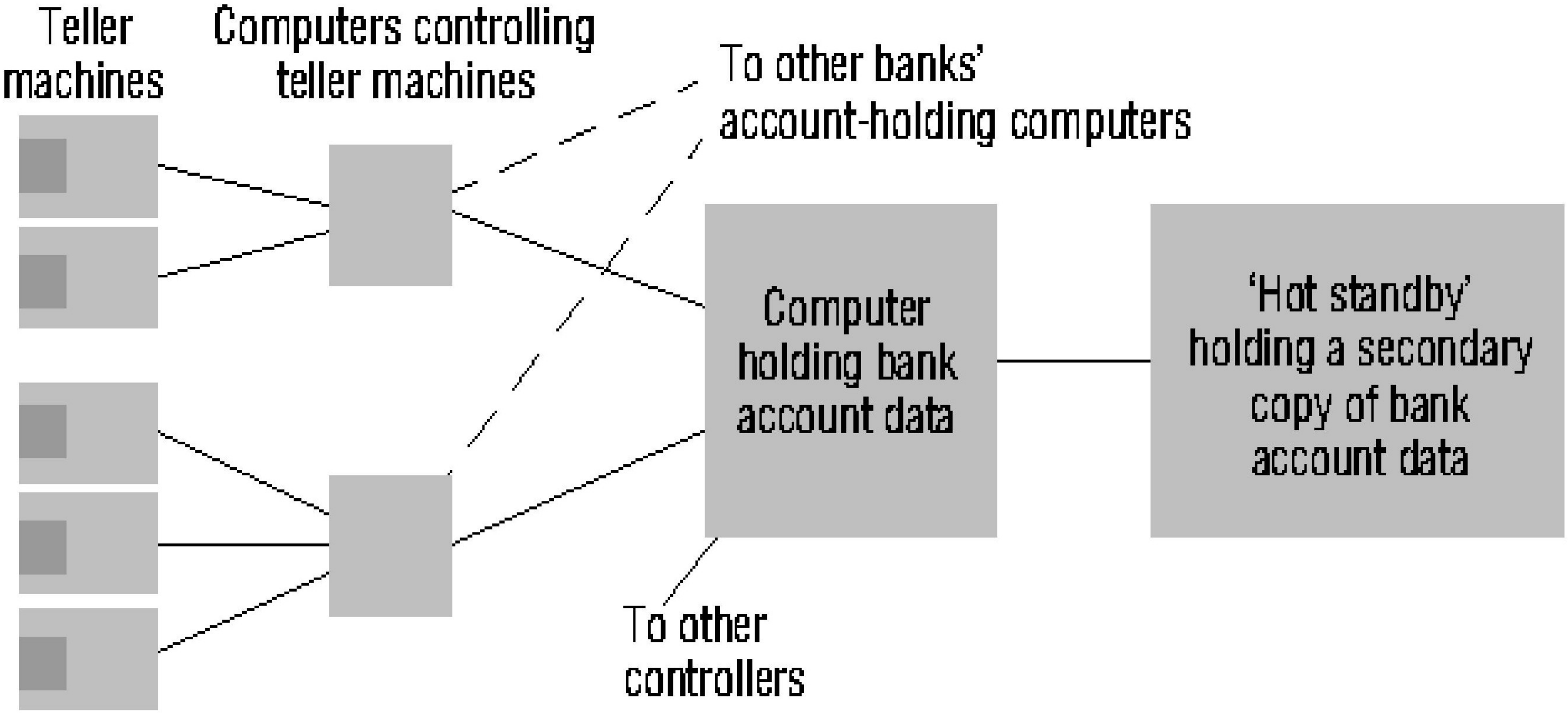
Distributed Computing

- The World Wide Web is an example of a network, but not an example of distributed computing.

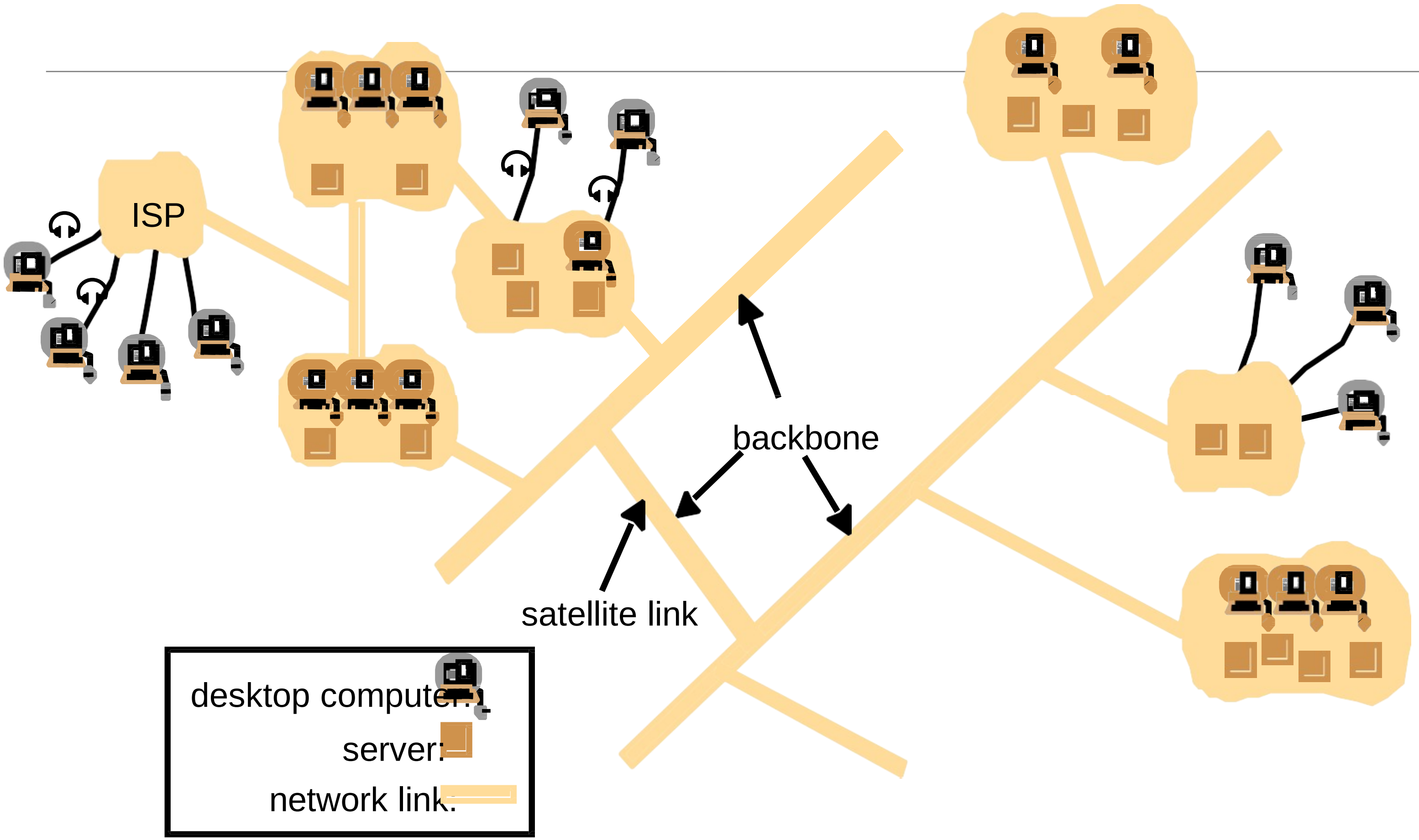
Examples of Distributed Systems

- Automatic Teller Machine network.
- Internet.
- Mobile and Ubiquitous Computing.

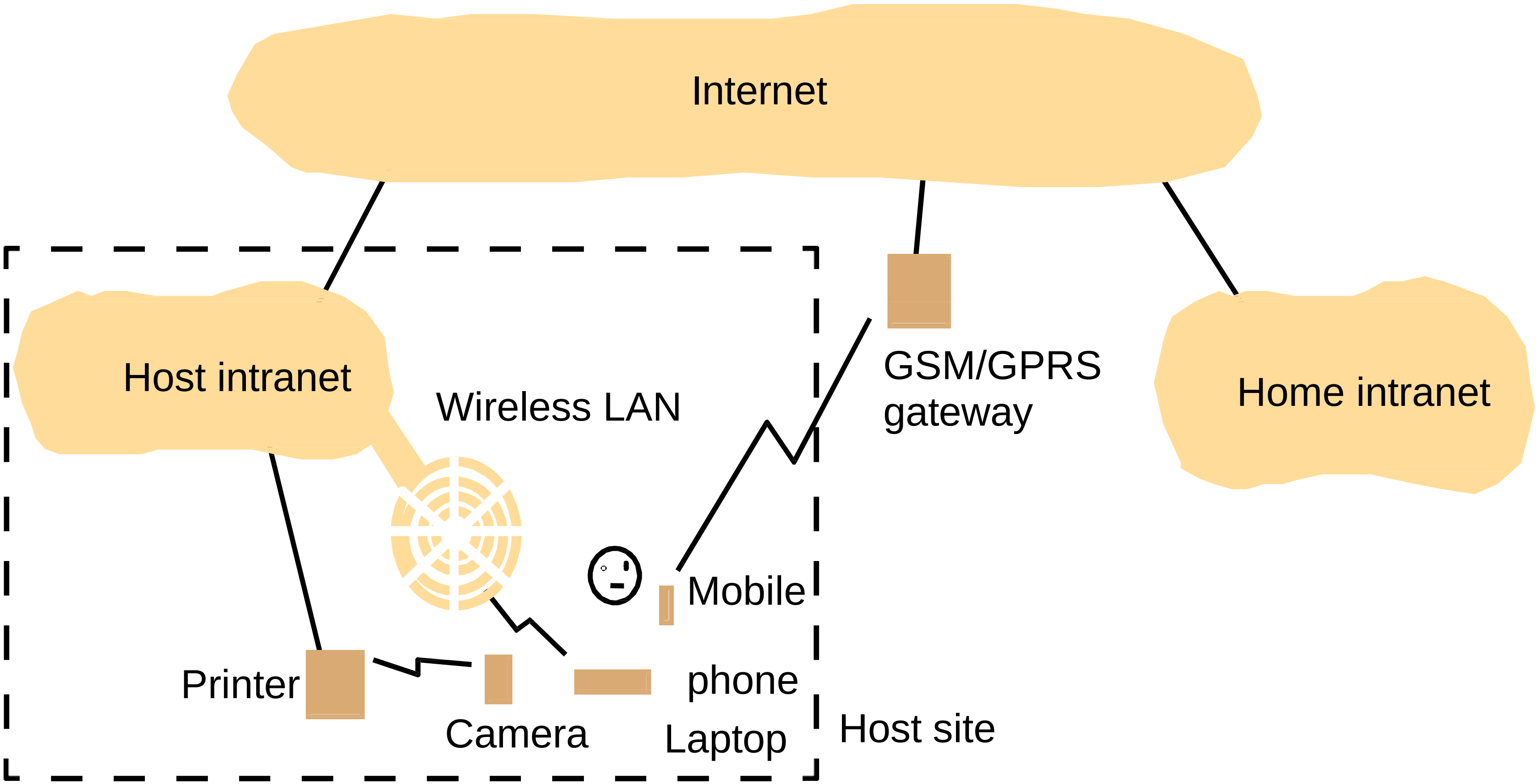
Automatic Teller Machine Network


Teller machines
Computers controlling teller machines
To other banks' account-holding computers
Computer holding bank account data
'Hot standby' holding a secondary copy of bank account data
To other controllers

Intern
et


ISP
backbone
satellite link
desktop computer:
server:
network link:

Mobile and Ubiquitous Computing



Why are Systems Distributed?

- Distributed systems have been around since the early 1970’s and have been made possible by advances in computer and communications technology.

Why are Systems Distributed?

- Systems are distributed for either or both of two main reasons:
  - Organizations today are expanding far beyond their traditional geographic boundaries in search for new business, new customers, new markets and improved financial and organizational viability. Consequently, an organization and its information systems may be inherently distributed and in connecting its systems into a seamless whole, a distributed system appears.
  - An organization may take inherently centralized information processing systems and distribute them to achieve higher reliability, availability, safety or performance, or all of the above.

Distributed Systems

- Multiple autonomous processing elements
- Information exchange over a network
- Processes interact via non-shared local memory
- Transparency.

autonomous processing elements

- A distributed system is composed of several independent components each with processing ability.
- There is no master-slave relationship between processing elements.

exchange over a network

- Thus, it excludestraditional centralized mainframe based systems.
- The network connects autonomous processing elements.

Processes interact via non-shared local memory.

- Multiple processor computer systems can be classified into those that share memory (multiprocessor computers) and those without shared memory (multi computers).
- A hybrid configuration involves separate computers with a distributed shared memory.

Transparency

- A distributed system is designed to conceal from the users the fact that they are operating over a wide spread geographical area and provide the illusion of a single desktop environment.

Transparency

- It should allow every part of the system to be viewed the same way regardless of the system size and provide services the same way to every part of the system.

Transparency

- Some aspects of transparency include:
  - Global names.
  - Global access.
  - Global security.
  - Global management.

Global names

- The same name works everywhere.
- Machines, users, files, control groups and services have full names that mean the same thing regardless of where in the system the name is used.

Global access

- The same functions are usable everywhere with reasonable performance.
- A program can run anywhere and get the same results.
- All the services and objects required by a program to run are available to the program regardless of where in the system it is executing.

Global security

- Same user authentication and control access work everywhere e.g. same mechanism to let the same person next door and someone at another site read ones files.
- Authentication to any computer in the system.

Global management

- The same person can administrate system components anywhere.
- System management tools perform the same actions e.g. configuration of workstations.

Components of a Distributed System.

- A distributed Systemhas three major components:
  - Data.
  - Processing.
  - Presentation.

Dat
a

- This is concerned with the structures and functions for information retention and manipulation.

Processing

- This is concerned with processing data objects.

Presentation

- Presentation is processing directly concerned with making data visible to users and handling user interaction.

Advantages of Distributed Systems.

- Fault tolerance.
- Flexibility.
- Extensibility.
- Upgrade.
- Local autonomy.
- Reliability and Availability.
- Localized security breaches.
- Improved performance.

Fault-tolerant

- It can be designed so that if one component of the system fails then the others will continue to work.
- Such a system will provide useful work in the face of quite a large number of failures in individual component systems.

More flexible

- A distributed system can be made up from a number of different components.
- Some of these components may bespecialized for a specific task while others may be general purpose.
- Components can be added, upgraded, moved and removed without impacting upon other components.

Easier to extend

- More processing, storage or other power can be obtained by increasing the number of components.

Easier to upgrade

- When a single large computer system becomes obsolete all of it has to be replaced in a costly and disruptive operation.
- A distributed system may be upgraded in increments by replacing individual components without a major disruption, or a large cash injection.

Local autonomy

- By allowing domains of control to be defined where decisions are made relating to purchasing, ownership, operating priorities, IS development and management, etc.
- Each domain decides where resources under its control are located

Reliability and Availability

- In a centralized system, a component failure can mean that the whole system is down, stopping all users from getting services.
- In a distributed system, multiple components of the same type can be configured to fail independently.

Reliability and Availability

- This aspect of replication of components improves the fault tolerance in distributed systems, consequently, the reliability and availability of the system is enhanced.

Improved Performance

- Large centralized systems can be slow performers due to the sheer volume of data and transactions being handled.
- A service that is partitioned over many server computers each supporting a smaller set of and users access to local data and resources results in faster access.

Improved performance

- Another performance advantage is the support for parallel access to distributed data across the organization.

Security breaches are localized

- In distributed systems with multiple security control domains, a security breach in one domain does not compromise the whole system.
- Each security domain has varying degree of security authentication, access control and auditing.

Disadvantages of Distributed Systems

- It’s more difficult to manage and secure.
- Lack of skilled support and development Staff.
- They are significantly more complex.
- They introduce problems of synchronization between processes.
- They introduce problems of maintaining consistency of data.

It’s more difficult to manage and secure.

- Centralized systems are inherently easier to secure and easier to manage because control is done from a single point. Distributed systems require more complex procedures for security, administration, maintenance and user support due to greater levels of co-ordination and control required.

Lack of skilled support and development Staff.

- Since the equipment and software in a DS can be sourced from different vendors, unlike in traditional systems where everything is sourced from the same vendor, its difficult to find personnel with a wide range of skills to offer comprehensive support.

Distribution Transparency

- A transparency is some aspect of the distributed system that is concealed from the user, programmer or system developer.

Distribution Transparency

- A major consideration when designing distributed systems is the extent to which the distribution of components of the system should be made transparent to the application designers and users because there are trade- offs to be made.

Distribution Transparency

- Users demand a simple user interface to this complex world, which in turn, demands functions for concealing complexity. Most users desire a single system image where all resources are perceived to be centralized at the desktop computer.

Distribution Transparency

- The term distribution transparency is used to describe the visibility of distributed components within a distributed system.
- There are two major transparency choices:
  - Partial or selective Distribution Transparency

Full Distribution Transparency

- Complexities introduced by distribution are completely concealed.
- It simplifies application development and improves usability.
- In addition, it accommodates the incorporation of existing systems based

Full Distribution Transparency

- For example, a database application designed to execute on a single host machine is able to make use of distributed resources without knowledge of whether the resource is local or remote and with no change to the application.

Partial or selective Distribution Transparency.

- The application designer may need to reveal that some DS components are distributed and therefore requires a development environment that gives some freedom to take account of this.

Partial or selective Distribution Transparency

- Full distribution does not allow applications the opportunity to exploit decentralization at the level of application process.

Partial or selective Distribution Transparency

- It’s the designer who chooses the extent to which users are made aware of distributed components by selecting a level of distribution transparency appropriate to the application.

Types of Transparency

- Access transparency
- Location Transparency
  .
- Concurrency Transparency .
- Replication Transparency .
- Fault Transparency .
- Migration Transparency.
- Performance Transparency.

Access transparency

- Hiding the use of communications to access remote resources like program files, data, printers, etc. so that the user is under the illusion that all resources are local.

Access transparency

- Remote resources are accessed using exactly the same mechanism for accessing local resources.
- From a programmer’s point of view, the access method to a remote object may be identical to access a local object of the same class.

Access transparency

- This transparency has two parts:
  - Keeping a syntactical or mechanical consistency between distributed and non-distributed access
  - Keeping the same semantics. Because the semantics of remote access are more complex, particularly failure modes, this means the local access should be a subset.

Location Transparency

- The details of the topology of the system should be of no concern to the user.
- The location of an object in the system may not be visible to the user or programmer.

Location Transparency

- This differs from access transparency in that both the naming and access methods may be the same.
- Names may give no hint as to location.

Concurrency Transparency

- Users and Applications should be able to access shared data or objects without interference between each other.
- This requires very complex mechanisms in a distributed system, since there exists true concurrency rather than the simulated concurrency of a central system.

Concurrency Transparency

- For example, a distributed printing service must provide the same atomic access per file as a central system so that printout is not randomly interleaved.

Replication Transparency

- Hiding differences between replicated and non-replicated resources.
- If the system provides replication (for availability or performance reasons) it should not concern the user.

Fault Transparency

- If software or hardware failures occur, these should be hidden from the user.
- This can be difficult to provide in a distributed system, since partial failure of the communications subsystem is possible, and this may not be reported

Fault Transparency

- As far as possible, fault transparency will be provided by mechanisms that relate to access transparency.
- However, when the faults are inherent in the distributed nature of the system, then access transparency may not be maintained.

Fault Transparency

- The mechanisms that allow a system to hide faults may result in changes to access mechanisms (e.g. access to reliable objects may be different from access to simple objects).

Fault Transparency

- In a software system, especially a networked one, it is often hard to tell the difference between a failed and a slow running process or processor.
- This distinction is hidden or made visible here.

Migration Transparency

- If objects (processes or data) migrate (to provide better performance, or reliability, or to hide differences between hosts), this should be hidden from the user.
- This means that resources can be relocated dynamically without the user being aware of reconfigurations.

Performance Transparency

- Minimizing performance overheads in using remote resources, so that the response time and through put are comparable with cases when all resources are local.
- The configuration of the system should not be apparent to the user in terms of performance

Performance Transparency

- This may require complex resource management mechanisms.
- It may not be possible at all in cases where resources are only accessible via low performance networks.

Scaling Transparency

- A system should be able to grow without affecting application algorithms. Graceful growth and evolution is an important requirement for most enterprises.

Scaling Transparency

- A system should also be capable of scaling down to small environments where required, and be space and/or time efficient as required.

Transparency

- If all of the above transparency functions are built into the IT infrastructure (usually the software components) then full distribution transparency can be achieved.

Transparency

- The degree of distribution transparency is a measure of the distributed nature of the IT infrastructure.

Issues in Design of Distributed systems

- There are key designs issues that people building distributed systems must deal with, with a goal to ensure that they are attained.

Issues in Design of Distributed systems

- These design issues are:
  - Transparency
  - Fault tolerance
  - Openness
  - Concurrency
  - Scalability
  - Performance

Transparency

- This is one of the key design issues in distributed systems.
- To design a system that achieves a single system image is very challenging.
- The concept of transparency has been proposed to describe distributed systems that can be made to behave like their non-distributed counterparts.

Transparency

- It is described 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 a collection of independent components.”

Transparency

- Transparency there involves hiding all the distribution from human users and application programs

Fault Tolerance

- Since failures are inevitable, a computer system can be made more reliable by making it fault tolerant.
- A fault tolerant system is one designed to fulfill its specified purposes despite the occurrence of component failures (machine and network).

Fault Tolerance

- Fault tolerant systems are designed to mask component failures i.e. attempt to prevent the failure of a system in spite of the failure of some of its components.

Fault Tolerance

- Fault tolerance can be achieved through hardware and software.
  - **Software** – Replication of programs e.g. replication of applications
    - Backward error recovery e.g. rollbacks and save points in database systems
  - **Hardware** – redundant hardware components e.g. multiple processors

Fault Tolerance

- Redundancy is defined as those parts of the system that are not needed for the correct functioning of the system if no fault tolerance is supported.
- Meaning that the system will continue to work correctly without redundancy if no failure occurs.

Fault Tolerance

- Redundancy can be exhibited in both hardware and software components.
- Fault tolerance improves system availability and reliability.
- Fault tolerance masks faults, which means that it makes faults not visible to the users externally.

Fault Tolerance

- The system still continues to function according to the specification.
- Fault tolerance can also be achieved by fault prevention i.e. trying to detect any fault in hardware and software before connecting to the system. For example program debugging

Fault Tolerance

- Although fault tolerance improves the system availability and reliability, it brings some overheads in terms of:
  - Cost - increased system costs
  - Software development – recovery mechanisms and testing
  - Performance – makes system slower in updates of replicas
  - Consistency – maintaining data consistency

Concurrency

- Concurrencyarises in a system whenseveral processes run in parallel.
- If these processes are not controlled then inconsistencies may arise in the system.

Concurrency

- This is an issue in of distributed systems because designers have to do this carefully and keenly to control the problem of inconsistency and conflicts.

Concurrency

- The end result is to achieve a serial access illusion.
- Concurrency control is important to achieve proper resource sharing and co-operation of processes.

Openness

- This is the ability of the system to accommodate different technology (hardware and software components) without changing the underlying structure of the system.

Openness

- For example, the ability to accommodate a 64- bit processor where a 32-bit processor was being used without changing the underlying system structure or the ability to accommodate a machine running a MAC O/S in a

Openness

- A distributed system should be open with respect to software vendors, developers, hardware components etc. this calls for well-defined interface so that the user just issues commands or requests without regard to the way the function is implemented so that you can have multiple independently developed copies using the same interface to provide services to the users.

Scalability

- Each component of a distributed system has a finite capacity.
- Designing for scalability involves calculating the capacity of each of these elements and the extent to which the capacity can be increased.

Performan ce

- There are two common measures of performance for distributed systems:
  - Response time – defined as the average elapsed time from the moment the user is ready to transmit and the entire response is received. The response received depends on the nature of the user interaction.
  - Throughput – the number of requests handled per unit time.

Performance

- Satisfactory performance as perceived by the users is dependent on the nature of the task being performed.
- The components in the path between and application

all performance characteristics wil
have that l
determi performance characteristics will
ne that
determi overall performance as the
ne perceived by
user.

Performance

- Servers will queue user requests until the necessary resource becomes available.
- In order to calculate performance in terms of response time the utilization of each component needs to be established and the effects of queuing calculated

Performance

- An alternate approach to improving CPU performance is to implement multi-processor or multi-computer configurations where each CPU handles a portion of the total workload.

Performance

- Effectively, the total workload is handled by a service that is partitioned across multiple CPU server group.
- When workload is partitioned across multiple servers, resolution protocol is required to find the server that is able to satisfy the client request.

Performance

- Performance improvements can be made in distributed systems environment by migrating much of the processing on to a user's client workstation.

- This reduces the processing on the server per client request, which, leads to faster, and more predictable response time

END