Chapter 1
Characterization of Distributed Systems

THE AMERICAN
UNIVERSITY IN CAIRO

Outline

- What is a distributed system?
- Examples of distributed systems
- Resource sharing
- Challenges

What is a distributed system?

- At a high-level of abstraction, a distributed system is one that is composed of multiple connected computers where a user requests services *by name*.
- In distributed systems, components interact solely via *message passing*.

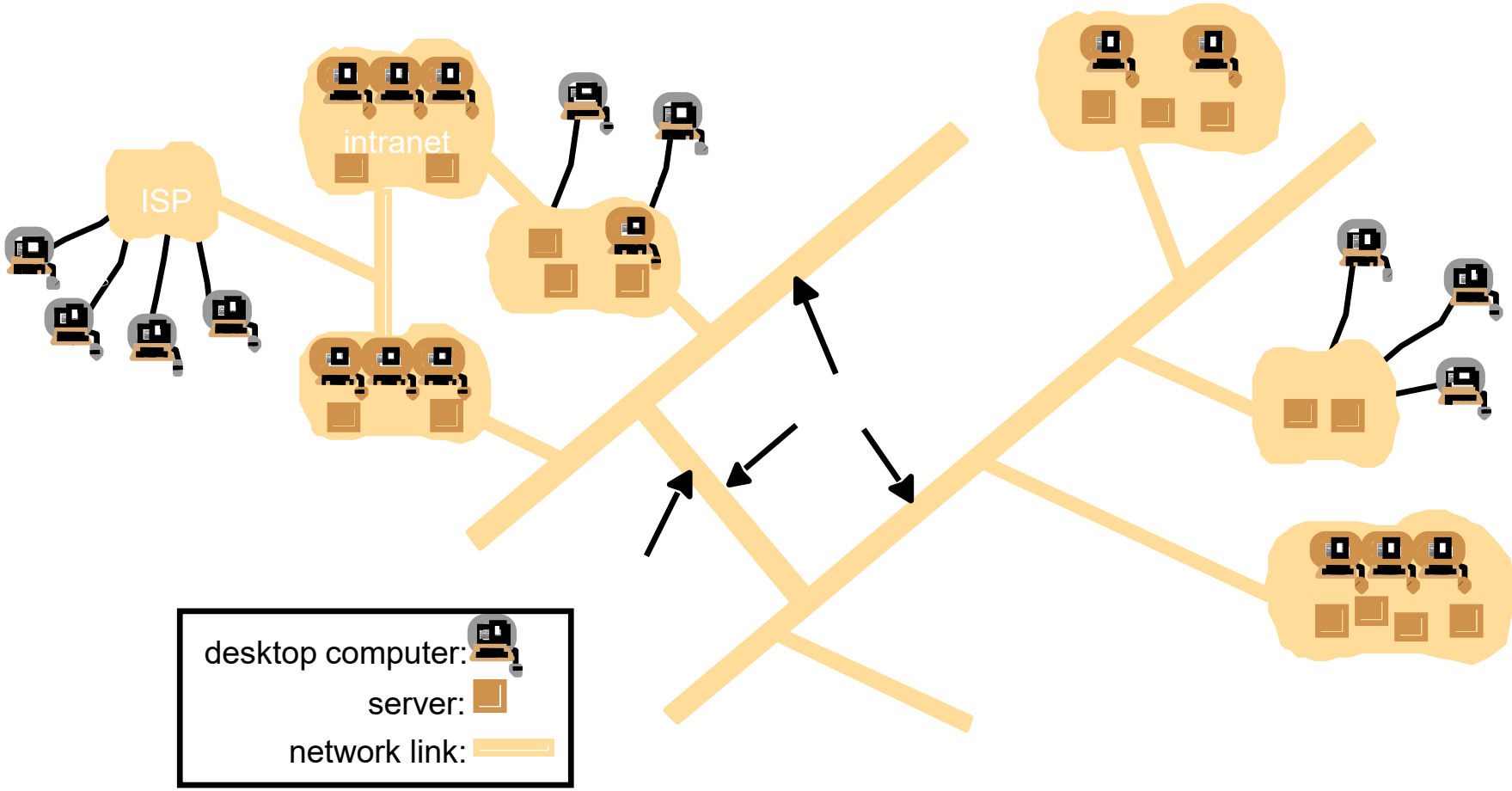
What is a distributed system?

- Based on such layout, we observe the following significant consequences:
  - Concurrency is inherent in distributed systems,
  - There is no global clock,
  - Distributed systems fail in new ways.
- We build distributed systems for one main reason: *sharing resources*, all kinds of resources!

Selected application domains and associated networked applications

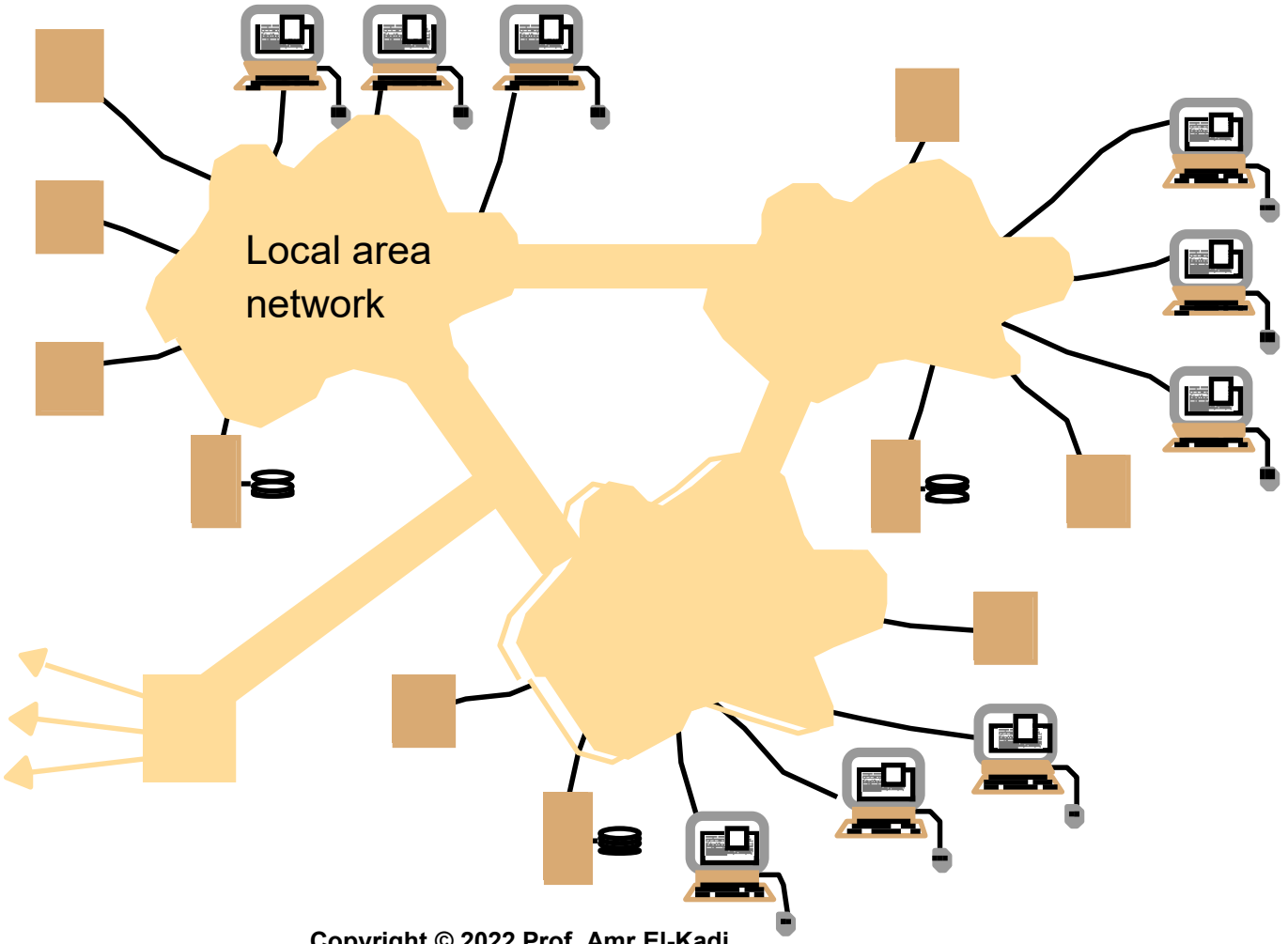
Finance and commerce	eCommerce e.g. Amazon and eBay, PayPal, online banking and trading
The information society	Web information and search engines, ebooks, Wikipedia; social networking: Facebook and MySpace.
Creative industries and entertainment	online gaming, music and film in the home, user-generated 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: The Internet



- Few issues to observe:
  - The set of services is open-ended,
  - Backbones are links with high transmission capacities realized through different technologies,
  - The ability of the Internet today to handle multimedia data is currently very limited due to the lack of facilities to reserve network bandwidth for individual streams of data.

Examples of distributed systems: An Intranet



- This is a portion of the Internet with clearly defined boundaries in which local security polices can be enforced.
- What is a *firewall* and how is it implemented?

- The main issues related to the deployment of components in an intranet are:
  - Users need to share data, so a distributed file service is needed.
  - Firewalls may be too restrictive, sometimes fine-grained security mechanisms are needed.
- The cost of software deployment and maintenance is a crucial issue; proper system architecture should be employed.

Mobile and ubiquitous computing

- Increasingly becoming the norm.
- Examples include:
  - Laptop computers.
  - Handheld devices, including Personal Digital Assistants (PDAs), mobile phones, and digital cameras.
  - Wearable devices, such as smart watches.
  - Embedded devices, such as hi-fi systems, microwave ovens, and vending machines.

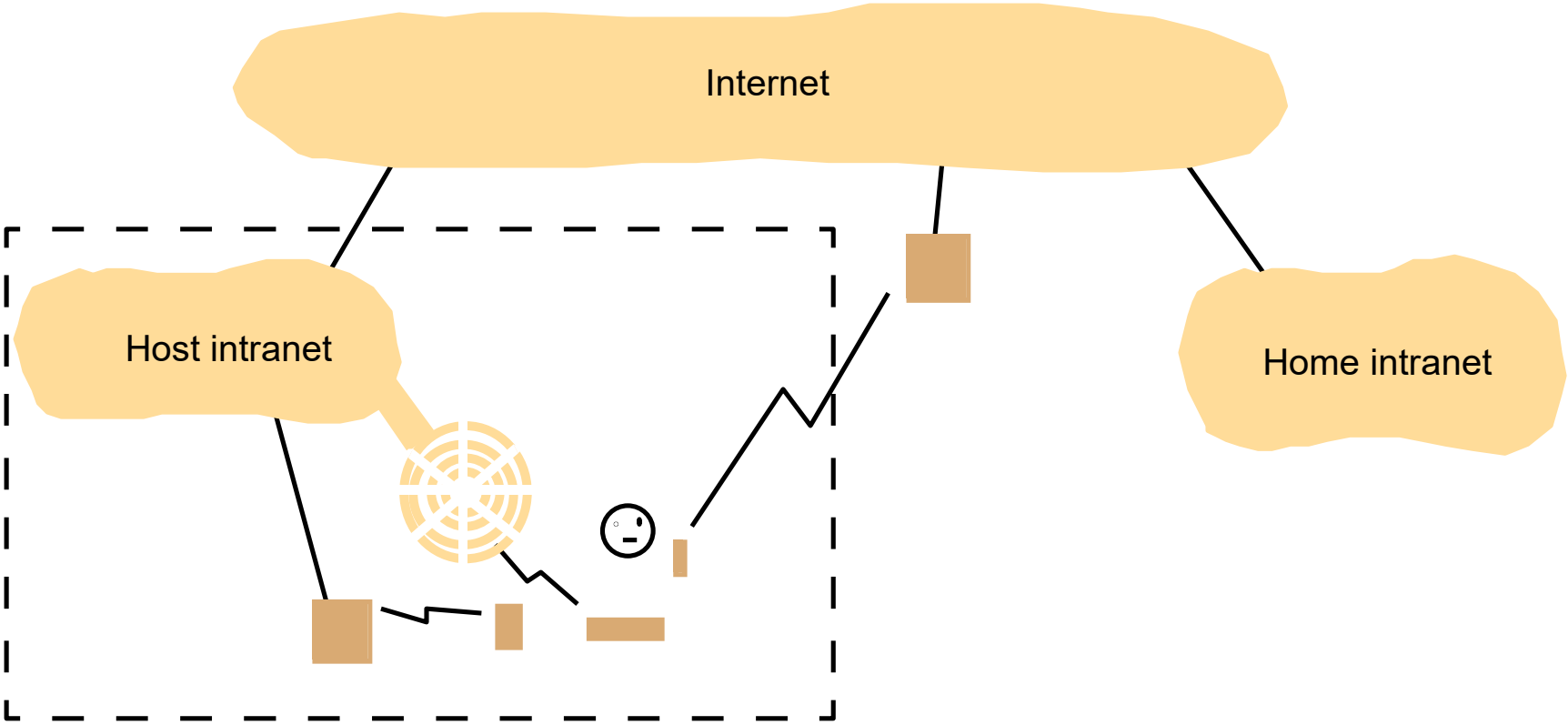
- *Mobile computing* (also called *nomadic computing*) is defined as the ability to perform computing while roaming.
- Users are able to utilize resources such as printers at conveniently nearby places as they move around. This is known as *location-aware computing*.

- *Ubiquitous computing* is possible through the utilization of many small and cheap computing devices that are present in the users' environments, for example smart cards.
- Computational behavior will be limited by their physical function.

- Ubiquitous and mobile computing overlap since mobile users expect to benefit from computers that are everywhere.
- Yet they are distinct
  - Ubiquitous computing would benefit a user while he/she remains in a single environment.
  - Similarly, a mobile computing user can still perform computing tasks even if it involves only more sophisticated devices such as laptops and printers.

- Both mobile and ubiquitous computing raise significant system issues including: discovery of resources, automatic reconfiguration of mobile devices, coping with limited connectivity while on the move, and security.

Portable and handheld devices in a distributed system



Resource sharing

- Resource sharing varies widely in scope and in how close users collaborate together.
- Examples of the two extremes include: *search engines*, and *Computer-Supported Cooperative Working (CSCW)*.
- Shared resources are accessed by *services* via the *operations* they export.

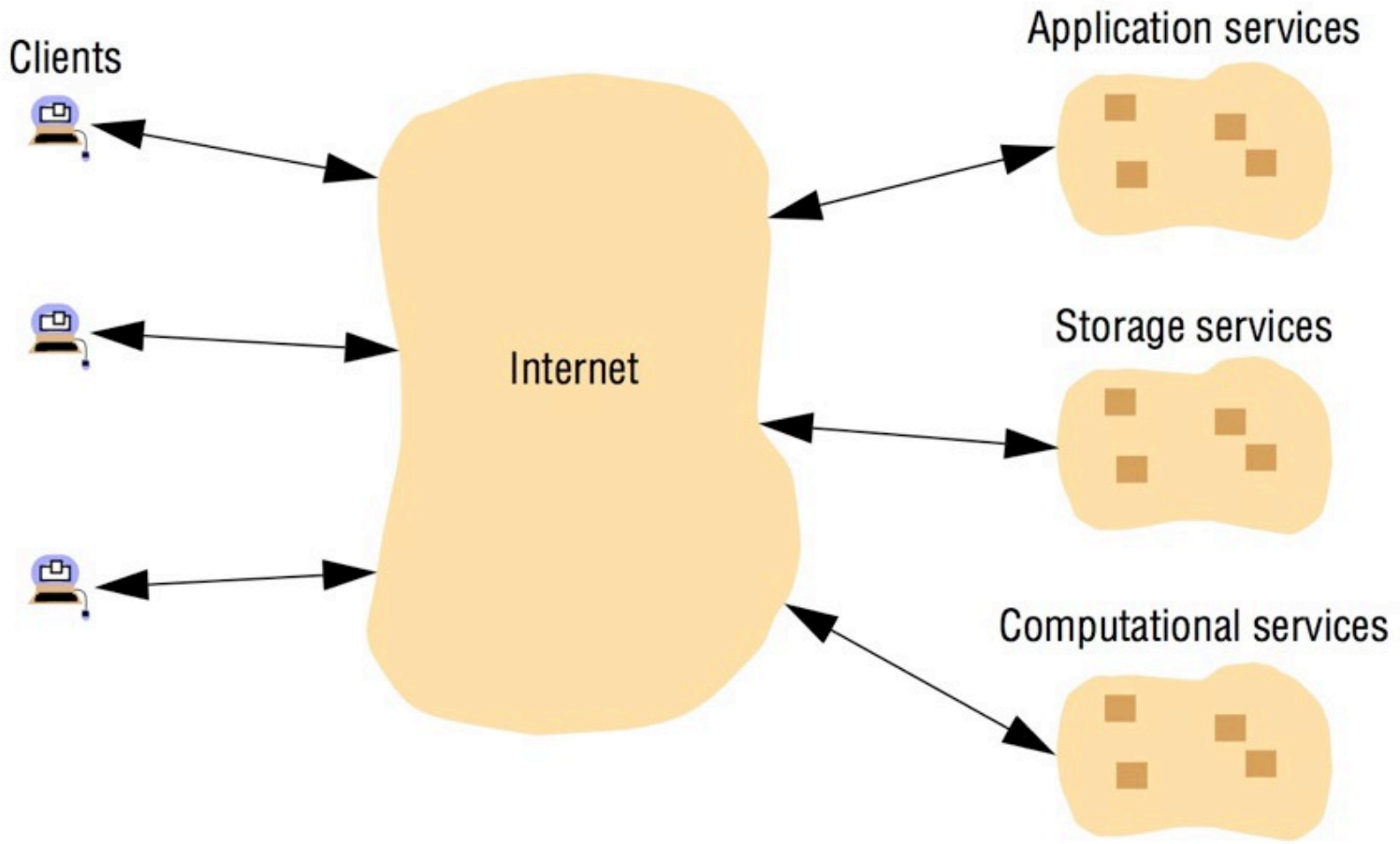
- Getting a service in a distributed system is achieved via a *remote invocation* mechanism.
- In the simple case, we have two active entities (*client* and *server*).
- Not all distributed systems are constructed entirely using the client-server model.

The World Wide Web

- It started as an environment providing a community of physicists working for the European Center for Nuclear Research (CERN) with a medium for exchanging documents via the Internet.
- Its key feature is the structure it provides in *hypertext*.
- The Web is an *open system* for two main reasons:
  1. It is based on standardized communication and documents' structures.
  2. It is open with respect to the type of resources that it can manipulate via the use of "helper" applications and "plug-ins".

- It is based on three major standard components:
  - The HyperText Markup Language (HTML).
  - Uniform Resource Locators (URLs).
  - A client-server model, with HyperText Transfer Protocol (HTTP).

Cloud computing



Challenges

- The design and development of distributed systems is **not** quite simple.
- Most of the issues we discuss here as challenges have been resolved, still designers have to be highly aware of them and take them into account.

1. Heterogeneity

- It applies to the following:
  - Networks
  - Computer hardware
  - Operating systems
  - Programming languages
  - Applications' interfaces

Middleware

- What is it?
- Most middleware deal with differences in operating systems and hardware (e.g., mobile code)
- They also provide uniformity for application developers and end-users

2. Openness

- In general, what is openness of a computer system?
- In distributed systems, this primarily targets how easy it is to add new resources to the system and make them available to clients.
- Openness cannot be achieved unless the key interfaces are published.

3. Security

- This has three components: confidentiality, integrity, and availability.
- Several security challenges are resolved through the use of encryption techniques.
- Examples of security challenges that have not been fully met include *denial of service attacks*, and *security of mobile code*.

4. Scalability

- What makes a system scalable?
- The Internet is a good example of a scalable system

Growth of the Internet (computers and web servers)

Date	*Computers*	*Web servers*	*Percentage*
1993, July	1,776,000	130	0.008
1995, July	6,642,000	23,500	0.4
1997, July	19,540,000	1,203,096	6
1999, July	56,218,000	6,598,697	12
2001, July	125,888,197	31,299,592	25
2003, July	~200,000,000	42,298,371	21
2005, July	353,284,187	67,571,581	19

- Designing scalable distributed systems present the following challenges:
  - Controlling the cost of physical resources
  - Controlling the performance loss
  - Preventing software resources running out
  - Avoiding performance bottlenecks
- Ideally, no change should be needed to the system as a result of a scale increase (or decrease)

5. Failure handling

- Failures in distributed systems are not the same as in other systems, they are *partial*.
- The following techniques exist for handling failures:
  - ***Detecting Failures*: Some failures could be detected, but the real challenge is to continue operation when failures happen and cannot be detected, yet they are suspected.**
  - ***Masking Failures*: When you detect failures, they can be masked or made less severe.**
  - ***Tolerating Failures*: Inform clients of failure presence and let them tolerate!**
  - ***Recovering from Failures*: Snapshots and rolling back.**
  - ***Redundancy*: Replicate resources. The challenge here is keeping replicas synchronized without major performance degradation.**

6. Concurrency

- Proper inter-process communication and synchronization mechanisms have to be utilized to ensure data consistency.

7. Transparency

- This is ***the*** key issue in a distributed system. The system should appear to end-users and application developers as a whole (one system) rather than a collection of interconnected components.

- Eight forms of transparency were identified:
  - *Access transparency*. Identical operations should be used to access local and remote resources.
  - *Location transparency.* Resources should be accessed without knowledge of their physical location.
  - *Concurrency transparency*. Ensures consistency of resources in the presence of concurrent access.
  - *Replication transparency.* End-users and applications developers should not be aware of the existence of multiple replicas of the resources.

- *Failure transparency*. End-users and applications developers should continue their tasks even with the presence of faults in hardware or software components.
- *Mobility transparency*. Both resources and clients should be allowed to move within the system without affecting the operation of the system.
- *Performance transparency*. The system should be reconfigured dynamically to match load variances.
- *Scaling transparency*. Allows the system and applications to scale dynamically with no change to the system structure or algorithms.