Distributed Internet Applications – DIA

Introduction
(Concepts and Definitions)
Uni-processor computing can be called *centralized computing*.

A *distributed system* is a collection of independent computers, interconnected via a network, capable of collaborating on a task.

*Distributed computing* is computing performed in a distributed system.
Distributed Systems

Internet
Examples of Distributed Systems

- Network of workstations: a group of networked personal workstations connected to one or more server machines.
- The Internet
- An intranet: a network of computers and workstations within an organization, segregated from the Internet via a firewall (a protective device)
Computers in a Distributed System

- **Workstations**: computers used by end-users to perform computing.
- **Server machines**: computers which provide resource and services.
- **Personal Assistance Devices**: handheld computers connected to the system via a wireless communication link.
A *host* is a computer that executes components that form part of a distributed system.
**What Is a Distributed System? (2)**

*Middleware* is a layer between network operating systems and applications that aims to resolve heterogeneity and distribution.

![Diagram]

- Component #1
- Component #n
- Middleware
- Network Operating System
- Hardware
- Host # i
A distributed system consists of components on networked hosts that interact via middleware so that they appear as an integrated facility.

A certain aspect of distribution is transparent if it is not visible to users, application programmers or administrators.

Transparency is a very important property of a distributed system.
A working Model of a Distributed System

Middleware

Network Operating System

Hardware

Host #2

Component #1
Component #n

Middleware

Network Operating System

Hardware

Host #1

Component #1
Component #n

Middleware

Network Operating System

Hardware

Host #n-1

Component #1
Component #n

Middleware

Network Operating System

Hardware

Host #n
Centralized Versus Distributed Systems

Centralized Systems

- Centralized systems have *non-autonomous* components.
- 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*.
- Centralized system have a single *point of failure*. 
Centralized versus Distributed Systems

**Distributed Systems**

- Distributed systems have *autonomous* components.
- Distributed systems may be built using heterogeneous technology.
- Distributed system components may be used exclusively.
- Distributed systems are executed in concurrent processes.
- Distributed system have multiple *points of failure*.
Centralized versus Distributed Systems

Computing

Centralized computing

- terminal
- mainframe computer

Distributed computing

- workstation
- network host
- network link
Distributed System Requirements

The need to distribute a system is often derived from non-functional requirements.

The non-functional requirements that typically lead to the adoption of distributed system architecture are:

- Scalability
- Openness
- Heterogeneity
- Resource Sharing
- Fault-Tolerance
Scalability denotes the ability to accommodate a growing load in the future.

Distributed system architectures achieve scalability through employing more than one host.

Distributed systems can be scalable because additional computers can be added in order to host additional components.
Openness means that the system can be easily extended and modified (integration of new components).

The integration of new components means that they have to be able to communicate with some of the components that already exist in the system.

Openness and distribution are related.

Distributed system components achieve openness by communicating using well-defined interfaces.
Heterogeneity of components can arise in the programming languages, operating systems, hardware platforms and network protocols.

The integration of heterogeneous components implies the construction of distributed systems.

Components have to communicate via the network and heterogeneity has to be resolved during that communication.

Component heterogeneity can be accommodated by distributed systems.
Resource Access and Sharing

- **Resources** – hardware, software and data – need to be shared by more than one user.
- Security of resource access needs to be considered.
- Distributed objects provide a sophisticated model of resource sharing.
Fault-Tolerance

Operations that continue even in the presence of faults are referred to as *fault-tolerant*.

Distributed systems achieve fault-tolerance by means of *replication*.
Transparency in distributed systems has several different dimensions.
Transparency in Distributed Systems

**Access transparency** means that the interfaces for local and remote communication are the same.

**Location transparency** means that service requesters do not need to know physical component locations.

**Migration transparency** means that a component can be relocated without users or clients noticing it.

Migration transparency depends on access and location transparency.
A **replica** is a component copy that remains synchronized with its original.

**Replication transparency** means that users and programmers do not know whether a replica or a master provides a service.

Replication transparency depends on access and location transparency.

**Concurrency transparency** means that users and programmers are unaware that components request services concurrently.
Transparency in Distributed Systems

 Scalability transparency means that users and programmers do not know whether a replica or a master provides a service.

 Scalability transparency depends on replication and migration transparency.

 Performance transparency means that users and programmers are unaware how good system performance is maintained.

 Performance transparency depends on replication and migration transparency.

 Replication of components provides the basis for a technique called load balancing, which achieves performance transparency.
Failure transparency means that users and programmers are unaware of how distributed systems conceal failure.

Failure transparency depends on concurrency and replication transparency.