Al Sham Private University
Faculty of Informatics Engineering

جامعة الشام الخاصة
كلية الهندسة المعلوماتية

نظم معلومات موزعة
Distributed Information Systems

Lecture 2: Challenges and Properties of Distributed Systems

اعداد: أ. غاندي هسام

Heterogeneity

- 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:
  - networks;
  - computer hardware;
  - operating systems;
  - programming languages;
  - implementations by different developers.
- their differences are masked by the fact that all of the computers attached to them use the Internet protocols to communicate with one another.

- Data types such as integers may be represented in different ways on different sorts of hardware – for example, there are two alternatives for the byte ordering of integers.
- These differences in representation must be dealt with if messages are to be exchanged between programs running on different hardware.
- For example, the calls for exchanging messages in UNIX are different from the calls in Windows.
- Different programming languages use different representations for characters and data structures such as arrays and records.
- Programs written by different developers cannot communicate with one another unless they use common standards

❑ 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.
- The Common Object Request Broker (CORBA) is an example.
- Some middleware, such as Java Remote Method Invocation (RMI) supports only a single programming language.
- 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

- In addition to solving the problems of heterogeneity, middleware provides a uniform computational model for use by the programmers of servers and distributed applications.
- Possible models include remote object invocation, remote event notification, remote SQL access and distributed transaction processing.
- For example, CORBA provides remote object invocation, which allows an object in a program running on one computer to invoke a method of an object in a program running on another computer.
- Its implementation hides the fact that messages are passed over a network in order to send the invocation request and its reply.

❑ 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.
- The ***virtual machine*** approach provides a way of making code executable on a variety of host computers.
- Today, the most commonly used form of mobile code is the inclusion ***Javascript*** programs in some web pages loaded into client browsers.

Openness

- The openness of a computer system is the characteristic that determines whether the system can be extended and reimplemented in various ways.
- 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.
- Openness cannot be achieved unless the specification and documentation of the key software interfaces of the components of a system are made available to software developers.
- the publication of interfaces is only the starting point for adding and extending services in a distributed system.

- Systems that are designed to support resource sharing in this way are termed ***open distributed systems*** to emphasize the fact that they are extensible.
- To summarize:
  - 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.

Security

- Security for information resources has three components: ***confidentiality*** (protection against disclosure to unauthorized individuals), ***integrity*** (protection against alteration or corruption), and ***availability*** (protection against interference with the means to access the resources).
- the challenge is to send sensitive information in a message over a network in a secure manner.
- it also involves knowing for sure the identity of the user or other agent on whose behalf a message was sent.
- identify a remote user or other agent correctly.

- The following two security challenges have not yet been fully met:

❑ *Denial of service attacks*

A user may wish to disrupt a service for some reason. This can be achieved by bombarding the service with such a large number of pointless requests that the serious users are unable to use it.

There have been several denial of service attacks on well-known web services.

❑ *Security of mobile code*:

Consider someone who receives an executable program as an electronic mail attachment: the possible effects of running the program are unpredictable; for example, it may seem to display an interesting picture but in reality it may access local resources, or perhaps be part of a denial of service attack.

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. Being possible to add server computers to avoid the performance bottleneck that would arise if a single file server (for example) had t handle all file access requests.

- *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 – for example, the table with the correspondence between the domain names of computers and their Internet addresses.

Algorithms that use hierarchic structures scale better than those that use linear structures.

- *Avoiding performance bottlenecks*:

Algorithms should be decentralized to avoid having performance bottlenecks. caching and replication may be used to improve the performance of resources that are very heavily used.

Failure handling

- 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.
- The following techniques for dealing with failures:

❖ *Detecting failures*:

Some failures can be detected. For example, checksums can be used to detect corrupted data in a message or a file.

The challenge is to manage in the presence of failures that cannot be detected but may be suspected.

❖ *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*:

Their clients can be designed to tolerate failures, which generally involves the users tolerating them as well. For example, when a web browser cannot contact a web server, it does not make the user wait for ever while it keeps on trying – it informs the user about the problem, leaving them free to try again later.

❖ *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. Consider the following examples:

1. There should always be at least two different routes between any two routers in the Internet.

2. A database may be replicated in several servers to ensure that the data remains accessible after the failure of any single server

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.
- The process that manages a shared resource could take one client request at a time. But that approach limits throughput. Therefore services and applications generally allow multiple client requests to be processed concurrently.
- To make this more concrete, suppose that each resource is encapsulated as an object and that invocations are executed in concurrent threads.

Transparency

- Transparency is defined as the hiding from the user and the application programmer of the separation of components in a distributed system, so that the system is seen as a whole rather than as a collection of independent components.
- *Access transparency* enables local and remote resources to be accessed using identical operations.
- *Location transparency* enables resources to be accessed without knowledge of their physical or network location (for example, which building or IP address).
- *Concurrency transparency* enables several processes to operate concurrently using shared resources without interference between them.
- *Mobility transparency* allows the movement of resources and clients within a system without affecting the operation of users or programs.

- *Replication transparency* enables multiple instances of resources to be used to increase reliability and performance without knowledge of the replicas by users or application programmers.
- *Failure transparency* enables the concealment of faults, allowing users and application programs to complete their tasks despite the failure of hardware or software components.
- *Performance transparency* allows the system to be reconfigured to improve performance as loads vary.
- *Scaling transparency* allows the system and applications to expand in scale without change to the system structure or the application algorithms.

- The two most important transparencies are access and location transparency; their presence or absence most strongly affects the utilization of distributed resources. They are sometimes referred to together as *network transparency*.
- As an illustration of access transparency, consider a graphical user interface with folders, which is the same whether the files inside the folder are local or remote.
- Another example is an API for files that uses the same operations to access both local and remote files
- As an example of a lack of access transparency, consider a distributed system that does not allow you to access files on a remote computer unless you make use of the ftp program to do so.

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 **nonfunctional** properties of systems that affect the quality of the service experienced by clients and users are *reliability*, *security* and *performance*.
- Its achievement depends upon the availability of the necessary computing and network resources at the appropriate times.

Case study: *The World Wide Web*

- **WWW** is an evolving system for publishing and accessing resources and services across the Internet.
- The Web is an open system: it can be extended and implemented in new ways without disturbing its existing functionality
- First, its operation is based on communication standards and document or content standards that are freely published and widely implemented.
- Second, the Web is open with respect to the types of resource that can be published and shared on it.
- The Web has moved beyond these simple data resources to encompass services, such as electronic purchasing of goods. It has evolved without changing its basic architecture.

- The Web is based on three main standard technological components:
  - the HyperText Markup Language (HTML), a language for specifying the contents and layout of pages as they are displayed by web browsers;
  - Uniform Resource Locators (URLs), also known as Uniform Resource Identifiers (URIs), which identify documents and other resources stored as part of the Web;
  - a client-server system architecture, with standard rules for interaction (the HyperText Transfer Protocol – HTTP) by which browsers and other clients fetch documents and other resources from web servers.
- A program that web servers run to generate content for their clients is referred to as a Common Gateway Interface (**CGI**) program. A CGI program may have any application-specific functionality, as long as it can parse the arguments that the client provides to it and produce content of the required type (usually HTML text). The program will often consult or update a database in processing the request.

- *Downloaded code*: A CGI program runs at the server. Sometimes the designers of web services require some service-related code to run inside the browser, at the user's computer. In particular, code written in Javascript. It often downloaded with a web page containing a form, in order to provide better-quality interaction with the user
- A *Javascript* enhanced page can give the user immediate feedback on invalid entries, instead of forcing the user to check the values at the server, which would take much longer.
- Javascript can also be used to update parts of a web page's contents without fetching an entirely new version of the page and re-rendering it.
- An alternative to a Javascript program is an **applet**: an application written in the Java language [2002], which the browser automatically downloads and runs when it fetches a corresponding web page.

End of Lecture 2