Characterization of Distributed Systems

Introduction

A distributed system is the one in which hardware or software components located at networked computers communicate and coordinate their actions only by passing messages.

Characteristics of Distributed Systems

Concurrency

In a network of computers, concurrent programs execution is the norm. Concurrency refers to the system’s ability to handle multiple access and use of shared resources. This is important because if there is no measure implemented, then data can get corrupted or lost by two nodes making different changes to the same resource causing an error in the program execution. One way to counteract these errors is to implement a locking mechanism making a node unable to access a resource whilst it is being used by another node.

Openness

The openness of a distributed system is defined as the difficulty involved to extend or improve an existing system. This characteristic allows us to reuse a distributed system for multiple functions or to process varying sets of data.

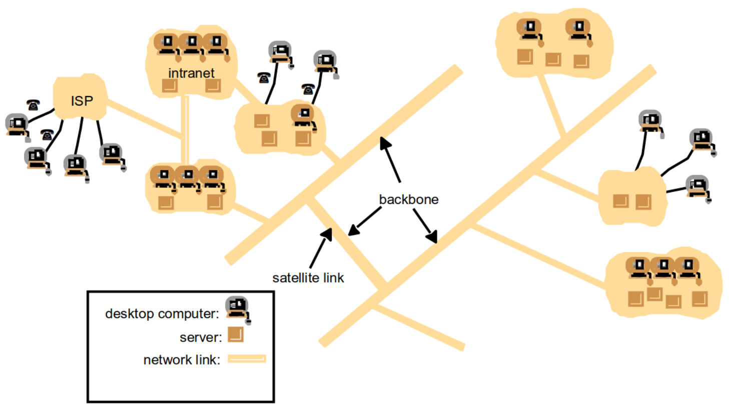
Fault Tolerance

Due to a distributed system having many computers comprised of different aged hardware, it is very likely for a part to fail in such a way that a node can no longer operate. Fault Tolerance is the ability for the system to handle such failures, this is achieved by using recovery and redundancy. Recovery is where a component will act in a predictable, controlled way if it relies on a component. Redundancy is where crucial systems and processes will have a backup that takes over if a system fails.

Q1 Examples of Distributed Systems

Internet

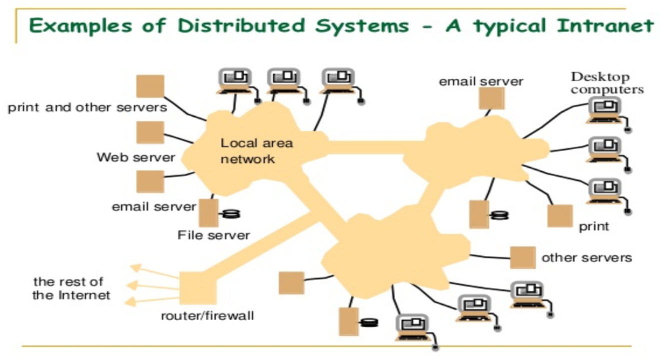
Figure . A typical portion of the Internet



The Internet is a very large distributed system. It enables users, wherever they are, to make use of services like www, email, file transfer, etc. The set of services is open-ended i.e. it can be extended by the addition of server computers and new types of services. The modern Internet is a vast interconnected collection of computer networks of many different types, with a range of types increasing all the time.

Intranets

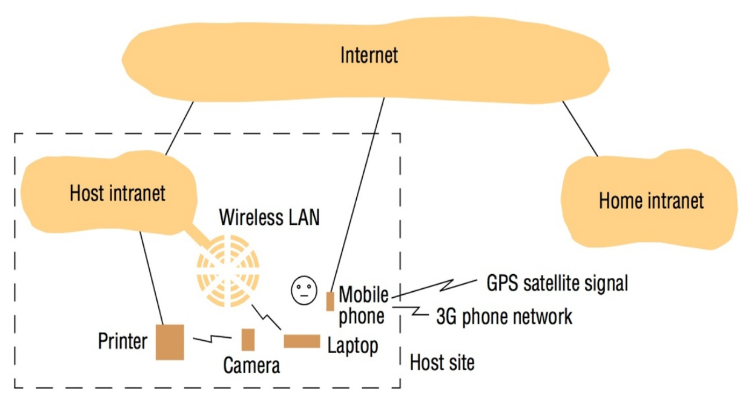
A typical Intranet



- An Intranet is a portion of the internet that is separately administered and has a boundary that can be configured to enforce local security policies.
- It may be composed of several LANs linked by backbone connections.
- The network configuration of a particular intranet is the responsibility of the organization that administers it.
- An intranet is connected to the Internet via a router, which allows the users to use the services available on the Internet.
- A firewall is used to protect the intranet by preventing unauthorized messages from leaving or entering.
- Some organizations do not wish to connect their internal networks to the Internet at all.
- E.g. police and other security and law enforcement agencies are likely to have at least some internal networks that are isolated from the outside world.
- These organizations can be connected to the Internet to avail themselves of the services by dispensing with the firewall.
- The main issues arising in the design of components for use in intranets are:
  - File services are needed to enable users to share data
  - Firewalls should ensure legitimate access to services
  - The cost of installation and support should be minimum

Mobile and Ubiquitous Computing

Figure: Portable and handheld devices in a distributed system



- Integration of portable computing devices like Laptops, smartphones, handheld devices, pagers, digital cameras, smartwatches, devices embedded in appliances like refrigerators, washing machines, cars, etc. with the distributed systems became possible because of the technological advances in device miniaturization and wireless networking.
- These devices can be connected to each other conveniently in different places, making mobile computing possible.
- In mobile computing, users who are away from the home intranet, are still allowed to access resources via the devices they carry.
- Ubiquitous computing is the harnessing of many small, cheap computational devices that are present in users' physical environments, including home, office, and others.
- The term ubiquitous is intended to suggest that small computing devices will eventually become so pervasive in everyday objects that they are scarcely noticed.
- The presence of computers everywhere is useful only when they can communicate with one another.
- E.g. it would be convenient for users to control their washing machine or their entertainment system using a “Universal remote control” device at home.
- The mobile user can get benefit from computers that are everywhere.
- Ubiquitous computing could benefit users while they remain in a single environment such as the home, office, or hospital.
- The figure shows a user who is visiting a host organization. The user's home intranet and the host intranet at the site that the user is visiting. Both intranets are connected to the rest of the Internet.

Q2 Challenges

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: Local network, the Internet, wireless network, satellite links, etc.

- Hardware devices: Computers, tablets, smartphones, embedded devices, etc.
- Operating systems: Windows, Linux, Mac, Unix, etc.
- Programming languages: Java, C/C++, Python, PHP, etc.
- Different roles like software developers, designers, system managers, etc.

Different programming languages use different representations for characters and data structures such as arrays and records. These differences must be addressed if programs written in different languages are to be able to communicate with one another. Programs written by different developers cannot communicate with one another unless they use common standards, for example, for network communication and the representation of primitive data items and data structures in messages. For this to happen, standards need to be agreed upon and adopted – as have the Internet protocols.

Openness

- The openness of a computer system is the characteristic that determines whether the system can be extended and re-implemented in various ways.
- The openness of distributed systems is determined primarily by the degree to which new resource-sharing devices can be added.
- 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. If the well-defined interfaces for a system are published, it is easier for developers to add new features or replace sub-systems in the future.

Security

- Many of the information in distributed systems have a high intrinsic value to their users.
- Their security is therefore of considerable importance.
- Security for information resources has three components:
  - **Confidentiality:** protection against disclosure to unauthorized individuals
  - **Integrity:** protection against alteration or corruption of data
  - **Availability:** protection against interference with the means to access the resources

Scalability

- 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 number of computers and servers on the Internet has increased dramatically.
- The design of scalable distributed systems presents the following challenges:
    - Controlling the cost of physical resources. Ex: File server expanding.
    - Controlling the performance loss. Ex: DNS nametable algorithms.
    - Preventing software & hardware resources from running out. Ex: IP address, CPU & Memory burst.
    - Avoiding performance bottlenecks. Ex: caching and replication, unexpected program failure, no proper error handling mechanism, etc.

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, i.e. some components fail while others continue to function.
- The following are techniques for dealing with failures:
    - **Detecting failures:** checksum and investigating servers under the crash.
    - **Masking failures:** Message re-transmission, writing data on multiple areas of the disk.
    - **Tolerating failures:** Web browser trying to re-connect to the API.
    - **Recovery from failure:** Rollback or backup state.
    - **Redundancy:** Routing (load-balancers), replication of database on multiple locations.

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.
- For example, a data structure that records bids for an auction may be accessed very frequently, multiple times when it gets close to the deadline 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.

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. In other words, distributed systems designers must hide the complexity of the systems as much as they can. Some terms of transparency in distributed systems are:

- **Access transparency:** Hide differences in data representation and how a resource is accessed.
- **Location transparency:** Enables resources to be accessed without knowledge of their physical or network location (for example, which building or IP address).
- **Concurrency transparency:** Hide that a resource may be shared and accessed by multiple users at the same time.
- **Replication transparency:** Hide that resources may be copied in several places.
- **Failure transparency:** Enables the concealment of faults, allowing users and application programs to complete their tasks despite the failure of hardware or software components.
- **Mobility transparency** allows the movement of resources and clients within a system without affecting the operation of users or programs.
- **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.

3. Define Architecture Model. Mention its goal & explain the following with example:
a) Mobile Code b) Mobile Agent
c) Proxy Server & Cace