Week 1

Tuesday, 11 February 2025 9:31 PM

Definition and Examples (1)

- Tanebaum and Van Steen (2007):
    - A distributed system (DS) is a collection of independent computers that appears to its users as a single coherent system.

- This definition addresses that:
    - Machines are autonomous;
    - Users think they are dealing with a single system - transparency.

- Issues with this definition:
    - How does the system work? - Concurrency
    - What is the purpose of using a Distributed system? - communication to...

Definition and Examples (2)

- Coulouris et al (2001):
    - A distributed system is a system in which hardware and software components located at networked computers communicate and coordinate their actions only by passing messages.

- This definition addresses that:
    - A network is for passing messages - communications
    - The system coordinates all the actions - concurrency.

- Issues with this definition:
    - How does the system appear to its user? - transparency

Definition and Examples (3)

- Both previous definitions only focus on computers.

- An alternative combination of both:
    - A distributed system is a collection to independent devices that appears to its users as a single coherent system, in which networked components communicate and coordinate their actions only by passing messages.
    - This definition covers the four important issues of any distributed system: networked communications, concurrency, transparency, and independent failure.

Es c

Example 1: A typical portion of Internet - Hardware

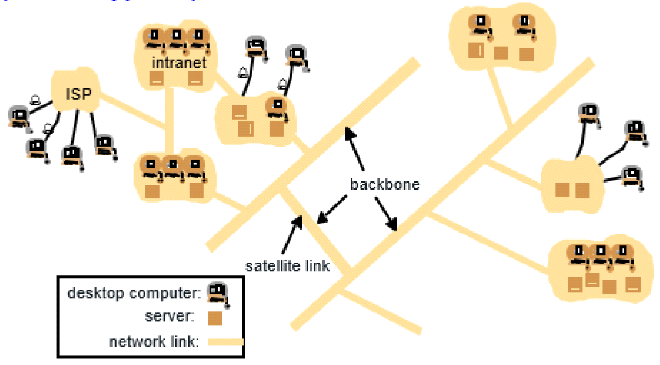


Figure 1.3 A typical portion of the Internet

Coulouris, Dollimore, Kindberg & Blair, Distributed Systems: Concepts and Design,5e, (c) 2012 Pearson.

Example 2: Portable and handheld devices in a distributed system - Hardware

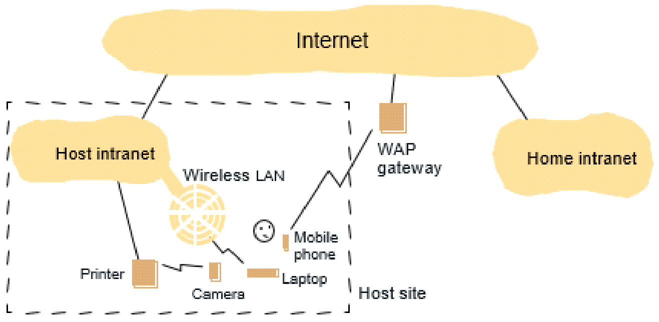


Figure 1.4 Portable and handheld devices in a distributed system

Coulouris, Dollimore, Kindberg & Blair, Distributed Systems: Concepts and Design,5e, (c) 2012 Pearson. All rights reserved. 0-13-214301-1

Example 3: Distributed services – Software system

- Electronic mail – SMTP (Simple Mail Transport Protocols)
- News – NNTP (Network News Transport Protocols)
- File Transfer – FTP (File Transfer Protocols)
- The World Wide Web – HTML/XML + HTTP (HyperText Transfer Protocols)
- Electronic payment
- Distributed transaction processing
- Distributed real-time processing

Network Connection perspective

- Centralized, decentralized, distributed

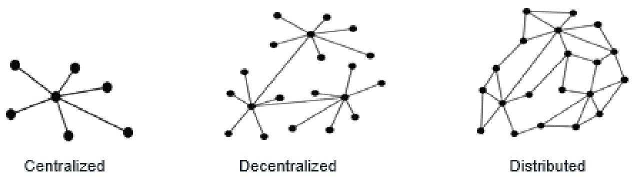


Figure 1.1: The organization of a (a) centralized, (b) decentralized, and (c) distributed system

(Van Steen & Tanenbaum, Distributed Systems: 4e, (c) 2023).

- When does a centralised system become distributed?
  - Adding 1 link between two nodes in a decentralised system?
  - Adding 2 links between two other nodes?
  - In generall adding k > 0 links?

- Alternative approach:
  - Two views on realizing distributed systems
    - Integrative view: a connection existing networked computer systems into a larger system.
    - Expansive view: an existing networked computer systems is extended with additional computers.

- Two definitions
  - ★ A **decentralized system** is a networked computer system in which processes and resources are necessarily spread across multiple computers .
  - ★ A **distributed system** is a networked computer system in which processes and resources are sufficiently spread across multiple computers .

Networked systems to distributed systems

- Distributed systems are complex: take perspectives
  - Architecture: common organisations
  - Process: what kind of processes, and their relationships
  - Communication: facilities for exchanging data.
  - Coordination/synchronization: application-independent algorithms

    - Naming: how do you identify resources?
    - Consistency and replication: performance requires data, which needs to be the same.
    - Fault tolerance: keep running in the presence of partial failures
    - Security: ensures authorised access to resources .

Design Goals, Benefits and Challenges

1.2.1 Design Goals (1)

- Overall design goals
    - Support sharing of resources
    - Distribution transparency
    - Openness
    - Scalability

1.2.1 Design Goals (2)

- Sharing of resources
    - Canonical examples
        - Cloud based shared storage and files.
        - Peer-to-peer assisted multimedia streaming
        - Shared mail services (think of outsourced mail systems)
        - Shared web hosting (think of content distribution networks)

1.2.1 Design Goals (3)

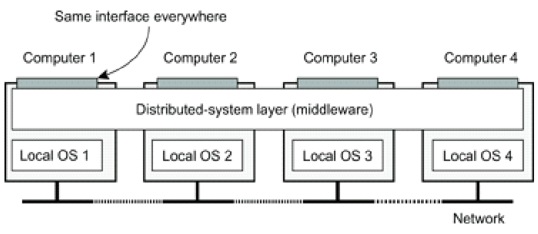
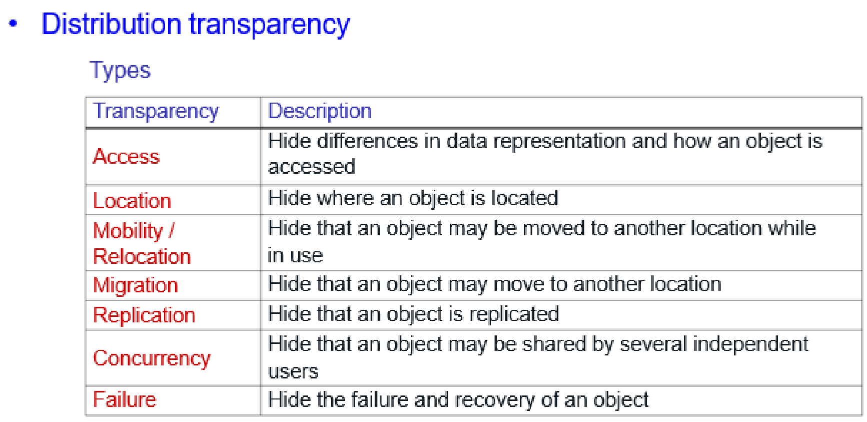
- Distribution transparency
- 

Figure 1.2: Realizing distribution transparency through a middleware layer

(Van Steen & Tanenbaum, Distributed Systems: 4e, (c) 2023).

- What is transparency
    - The phenomenon by which a distributed system attempts to hide the fact that its processes and resources are physically distributed across multiple computers, possibly separated by large distances.
- Observation
    - Distribution transparency is handled through many different techniques in a layer between applications and operating systems: a middleware layer

1.2.1 Design Goals (4)



- Distribution transparency

Types

Transparency	Description
Access	Hide differences in data representation and how an object is accessed
Location	Hide where an object is located
Mobility / Relocation	Hide that an object may be moved to another location while in use
Migration	Hide that an object may move to another location
Replication	Hide that an object is replicated
Concurrency	Hide that an object may be shared by several independent users
Failure	Hide the failure and recovery of an object

- ***Access transparency***: enables local and remote resources to be accessed using identical operations.

- ***Location transparency***: enables resources to be accessed without knowledge of their location.

    Access + Location = Network transparency

- ***Mobility/relocation 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.

- ***Concurrency transparency***: enables several processes to operate concurrently using shared resources without interference between them.

- ***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.

1.2.1 Design Goals (5)

- Degree of transparency
  - Aiming at full distribution transparency may be too much.
    - There are communication latencies that cannot be hidden.
    - Completely hide failures of networks and nodes is (theoretically and practically impossible.
      - You cannot distinguish a slow computer from a failing one.
      - You can never be sure that a server actually performed an operation before a crash.
  - Full transparency will cost performance, exposing distribution of the system
    - Keeping replicas exactly up to date with master takes time
    - Immediately flushing write operations to disk for fault tolerance.

1.2.1 Design Goals (6)

- Degree of Transparency
  - Exposing distribution may be good
    - Making use of location based services (finding your nearby friends)
    - When dealing with users in different time zones
    - When it makes it easier for a user to understand what's going on (when e.g a server does not respond for a long time, report it as failing).

1.2.1 Benefits (1)

1) Shareability: the ability the comprising systems to use each other's resources.
   - The sharing of resources (hardware , software and information) is a main motivation for constructing distributed systems.
   - Resources may be managed by servers and accessed by clients, or they may be encapsulated as objects and accessed by other client objects
2) Expandability: the ability that permits new systems to be added as members of the overall system.
   - This not only brings more sharing resources into the system, but also allows more users to share both existing and new resources.
3) Local Autonomy:
   - means that a distributed system is responsible for managing its resources.
   - It can apply local policies, settings and access controls to its resources and services.
4) Improved Performance:
   - a distributed system allows resources to be distributed in different machines, making the processing more efficient.
5) Improved reliability and availability:
   - in a distributed system resources are spread or replicated across multiple computers. A disruption might cause only minimum impact of the system.
6) Potential Cost Reduction.
   - Obvious but not always.

1.2.3 Challenges (1)

1) Heterogeneity:

- A distributed system should allow users to access services and run application over a heterogenous collection of computers and networks.

- Heterogeneity applies to
  - Networks
  - Computer hardware
  - Operating systems
  - Programming languages
  - Application implementations by different developers .

2) Challenges (2)

- Openness
  - A distributed system should allow the system to be extended and re-implemented in various ways.
    - It is measured primarily by the degree to which new resource-sharing services can be added without disruption or duplication of existing services and be made available for use by a variety of client programs.

  - Open distributed system
    - A system that offers components that can easily be used by, or integrated into other systems, an open distributed system itself will often consist of components that originate from elsewhere.
    - Be able to interact with services from other open systems, irrespective of the underlying environment.

3) Security:

- A distributed system should not allow its users to share resources more easily, but also protect these resources against unauhtorized actives with them.

- Security for data resources has three components (CIA):
  - Confidentiality (against disclosure to unauthorized individuals)
  - Integrity 9against alteration or corruption)
  - Availability (against interference with the means to access the resources)

4) Scalability:
- Scalability in a distributed system is the characteristics where a system remains effective when there is a significant increase in the number of resources and the number of users.

- Observation
  - Many developers of modern distributed system easily use the adjective "Scalable" without making clear why their system scale.
  - At least three components
    - Number of users or processes (size scalability)
    - Maximum distance between nodes (geographical scalability)
    - Number of administrative domains (administrative scalability)

5) Failure handling:
- Failures in a distributed system should be partial, I.e some components fail while other continue to function. Distributed systems provided a high degree of availability in the face of hardware faults.
- The availability of a system is a measure of the proportion of time that is available for use.

- Techniques for handling failures include:
  - Detecting failures
  - Masking failures
  - Tolerating failures
  - Recovery from failures
  - Redundancy

6) Concurrency Control:
- The ability to process multiple tasks at the same time.

7) Transparency:
- The concealment of the separation of components in a distributed system so that the system is able to present itself to users and applications as if it were only a single computer system.
- Measured by different type of transparencies
- Achieving may cost performance (e.g impossible to completely hide failures, and keeping data consistency among nodes will take time and waste network bandwidth, etc)

General System Model
- Distributed systems are often organised as layered structure

-

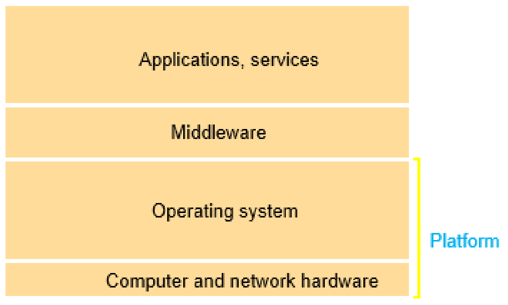


Figure 2.7 Software and hardware service layers in distributed systems.

- System Model
  - Platform
    - This lowest layer provides services to the layers above them, which are implemented independently in each computer. Each local OS, which forms part of the underlying network OS, should provide local resource management and communication means to other computers.
    - Canonical Examples
      - Intelx86 windows
      - Intel x86 Linux
      - Intel x86 Solaris

The Middleware
- There is a type of distributed system software that connects different kinds of applications and provides distributed transparency to its connected applications. Middleware does not manage an individual node.
- The purpose is to mask heterogeneity present in distributed systems and to provide a convenient programming model to application developers.

  Examples
  - Sun RPC (Remote Procedure Calls)
  - OMG CORBA (Common Object Request Broker Architecture)
  - Microsoft D-COM (Distributed Components Object Model)

- Sun Java RMI

  Modern Middleware:
  - □ Manjrasoft Aneka– for Cloud computing
  - □ IBM WebSphere
  - □ Microsoft .NET
  - □ Sun J2EE
  - □ Google AppEngine, etc.

1.3.2 Middleware (1)

➢ **Middleware position:** between applications and operating system layers

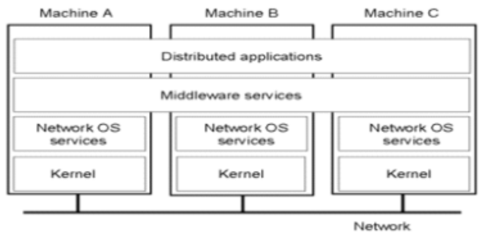


Figure 1-1. A distributed system organized as middleware. The middleware layer extends over multiple machines, and offers each application the same interface.

- Middleware Models
  - To make development and integration of distribution applications as simple as possible, mode middleware is based on a certain model for describing distribution and communication.

  - Some early models:
    - Unix file systems
      - Everything is treated as a file, and whether a file is local or remote makes no difference.
      - An application open a file, reads and write data, and closes it again.
      - Communication efficiency is reduced when a file is accessed by several processes at the same time.

  - Some early models
  - Remote Procedure calls (RPCs)
    - The emphasis is on hiding network communication by allowing a process to call a procedure being executed on a remote machine.
    - The calling process remains unaware of the fact that the network communication took place.

  - Recent Models:
    - Distributed objects
      - A middle ware model based on distributed objects should be able to hide not only network communication offered by RPCs, but also to invoke objects residing in remote machines in a transparent fashion.
      - Each object implements an interface that hides all the internal details of the objects from its users. The only thing that a process sees of an object is its interface.

    - Distributed Documents (WWW)
      - Information is organised into documents, with each document residing at a machine transparently located somewhere in the world.
      - Documents contain links that refer to other documents. By following a link, the document to which that link refers to is fetched from its location and displayed on the users screen.

MiddleWare services
- Communication Services
  - Middleware offers high-level communication services that hide the low-level message passing through computer networks.
- Naming
  - Name services allow entities to be shared and looked up (as in directories), similar to phone books.
- Persistence
  - Many middle ware systems offer special facilities for storage, referred to as persistence .
- Distributed Transactions
  - Many middleware systems offer special facilities for distributed transactions. Distributed transaction services allow multiple read and write to occur atomically, I.e a transaction either succeeds, so that all its write operation are actually performed, or fails, leaving all referenced data unaffected.