Distributed Systems

Chapter 1

Outline

- Definition of a Distributed System
- Goals of a Distributed System
- Types of Distributed Systems

What Is A Distributed System?

- A collection of independent computers that appears to its users as a single coherent system.
- Features:
  - No shared memory – message-based communication
  - Each runs its own local OS
  - Heterogeneity
- Ideal: to present a single-system image:
  - The distributed system “looks like” a single computer rather than a collection of separate computers.

Distributed System Characteristics

- To present a single-system image:
  - Hide internal organization, communication details
  - Provide uniform interface
- Easily expandable
  - Adding new computers is hidden from users
- Continuous availability
  - Failures in one component can be covered by other components
- Supported by middleware

Definition of a Distributed System

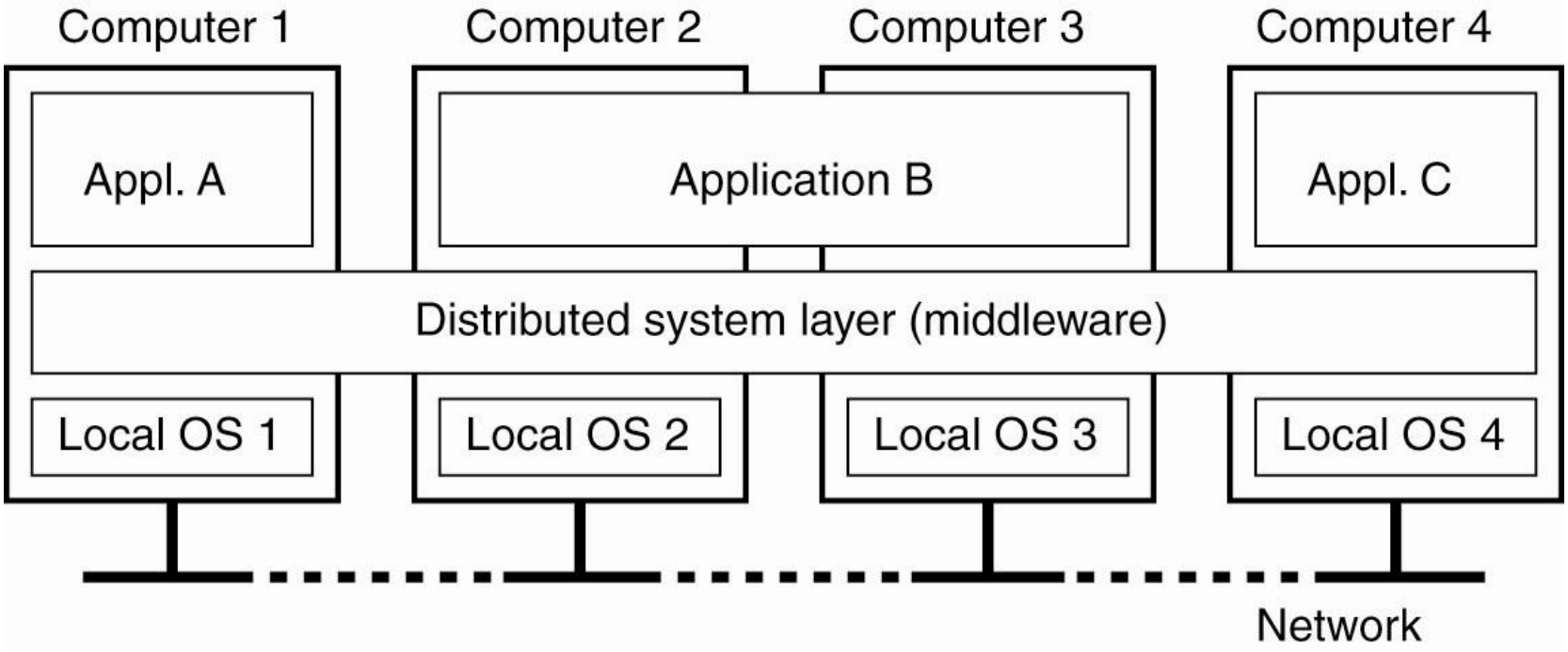


Figure 1-1. A distributed system organized as middleware. The middleware layer runs on all machines, and offers a uniform interface to the system

Role of Middleware (MW)

- In some early research systems: MW tried to provide the illusion that a collection of separate machines was a single computer.
  - E.g. NOW project: GLUNIX middleware
- Today:
  - clustering software allows independent computers to work together closely
  - MW also supports seamless access to remote services, doesn't try to look like a general-purpose OS

Middleware Examples

- All of the previous examples support communication across a network:
- They provide protocols that allow a program running on one kind of computer, using one kind of operating system, to call a program running on another computer with a different operating system
  - The communicating programs must be running the *same* middleware.

Distributed System Goals

- Resource Accessibility
- Distribution Transparency
- Openness
- Scalability

Goal 1 – Resource Availability

- Support user access to remote resources (printers, data files, web pages, CPU cycles) and the fair sharing of the resources
- Economics of sharing expensive resources
- Performance enhancement – due to multiple processors; also due to ease of collaboration and info exchange – access to remote services
  - Groupware: tools to support collaboration
- Resource sharing introduces security problems.

Goal 2 – Distribution Transparency

- Software hides some of the details of the distribution of system resources.
  - Makes the system more user friendly.
- A distributed system that appears to its users & applications to be a single computer system is said to be *transparent*.
  - Users & apps should be able to access remote resources in the same way they access local resources.
- Transparency has several dimensions.

Types of Transparency

Transparency	Description
Access	Hide differences in data representation & resource access (enables interoperability)
Location	Hide location of resource (can use resource without knowing its location)
Migration	Hide possibility that a system may change location of resource (no effect on access)
Replication	Hide the possibility that multiple copies of the resource exist (for reliability and/or availability)
Concurrency	Hide the possibility that the resource may be shared concurrently
Failure	Hide failure and recovery of the resource. How does one differentiate betw. slow and failed?
Relocation	Hide that resource may be moved during use

Figure 1-2. Different forms of transparency in a distributed system (ISO, 1995)

Goal 2: Degrees of Transparency

- Trade-off: transparency versus other factors
  - Reduced performance: multiple attempts to contact a remote server can slow down the system – should you report failure and let user cancel request?
  - Convenience: direct the print request to my local printer, not one on the next floor
- Too much emphasis on transparency may prevent the user from understanding system behavior.

Goal 3 - Openness

- An **open distributed system** "…offers services according to standard rules that describe the syntax and semantics of those services." In other words, the interfaces to the system are clearly specified and freely available.
  - Compare to network protocols
  - *Not* proprietary
- **Interface Definition/Description Languages (IDL):** used to describe the interfaces between software components, usually in a distributed system
  - Definitions are language & machine independent
  - Support communication between systems using different OS/programming languages; e.g. a C++ program running on Windows communicates with a Java program running on UNIX
  - Communication is usually RPC-based.

Open Systems Support …

- **Interoperability**: the ability of two different systems or applications to work together
  - A process that needs a service should be able to talk to any process that provides the service.
  - Multiple implementations of the same service may be provided, as long as the interface is maintained
- **Portability**: an application designed to run on one distributed system can run on another system which implements the same interface.
- **Extensibility**: Easy to add new components, features

Goal 4 - Scalability

- Dimensions that may scale:
  - With respect to size
  - With respect to geographical distribution
  - With respect to the number of administrative organizations spanned
- A scalable system still performs well as it scales up along any of the three dimensions.

Size Scalability

- Scalability is negatively affected when the system is based on
  - Centralized server: one for all users
  - Centralized data: a single data base for all users
  - Centralized algorithms: one site collects all information, processes it, distributes the results to all sites.
    - Complete knowledge: good
    - Time and network traffic: bad

Decentralized Algorithms

- No machine has complete information about the system state
- Machines make decisions based only on local information
- Failure of a single machine doesn't ruin the algorithm
- There is no assumption that a global clock exists.

Geographic Scalability

- Early distributed systems ran on LANs, relied on **synchronous communication.**
  - May be too slow for wide-area networks
  - Wide-area communication is unreliable, point-to-point;
  - Unpredictable time delays may even affect correctness
- LAN communication is based on broadcast.
  - Consider how this affects an attempt to locate a particular kind of service
- Centralized components + wide-area communication: waste of network bandwidth

Scalability - Administrative

- Different domains may have different policies about resource usage, management, security, etc.
- Trust often stops at administrative boundaries
  - Requires protection from malicious attacks

Scaling Techniques

- Scalability affects performance more than anything else.
- Three techniques to improve scalability:
  - Hiding communication latencies
  - Distribution
  - Replication

Hiding Communication Delays

- Structure applications to use **asynchronous communication** (no blocking for replies)
  - While waiting for one answer, do something else; *e.g.*, create one thread to wait for the reply and let other threads continue to process or schedule another task
- Download part of the computation to the requesting platform to speed up processing
  - Filling in forms to access a DB: send a separate message for each field, or download form/code and submit finished version.
  - i.e., shorten the wait times

Distribution

- Instead of one centralized service, divide into parts and distribute geographically
- Each part handles one aspect of the job
  - Example: DNS namespace is organized as a tree of domains; each domain is divided into zones; names in each zone are handled by a different name server
  - WWW consists of many (millions?) of servers

Third Scaling Technique - Replication

- Replication: multiple identical copies of something
  - Replicated objects may also be distributed, but aren't necessarily.
- Replication
  - Increases availability
  - Improves performance through load balancing
  - May avoid latency by improving proximity of resource

Summary
Goals for Distribution

- Resource accessibility
  - For sharing and enhanced performance
- Distribution transparency
  - For easier use
- Openness
  - To support interoperability, portability, extensibility
- Scalability
  - With respect to size (number of users), geographic distribution, administrative domains

Issues/Pitfalls of Distribution

- Requirement for advanced software to realize the potential benefits.
- Security and privacy concerns regarding network communication
- Replication of data and services provides fault tolerance and availability, but at a cost.
- Network reliability, security, heterogeneity, topology
- Latency and bandwidth
- Administrative domains

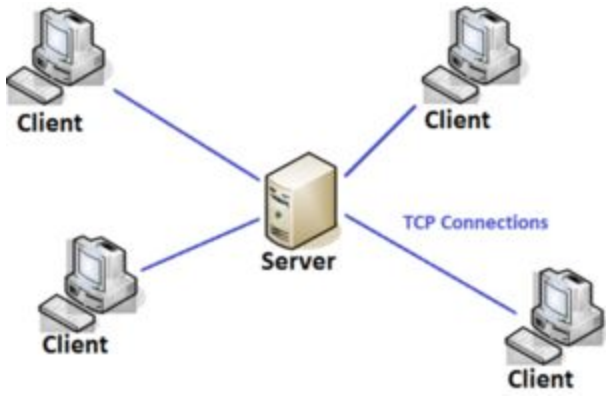
Distributed Systems

- Early distributed systems emphasized the single system image – often tried to make a networked set of computers look like an ordinary general purpose computer
- Examples: Amoeba, Sprite, NOW, Condor (distributed batch system), …

Types of Distributed system

Client -server system

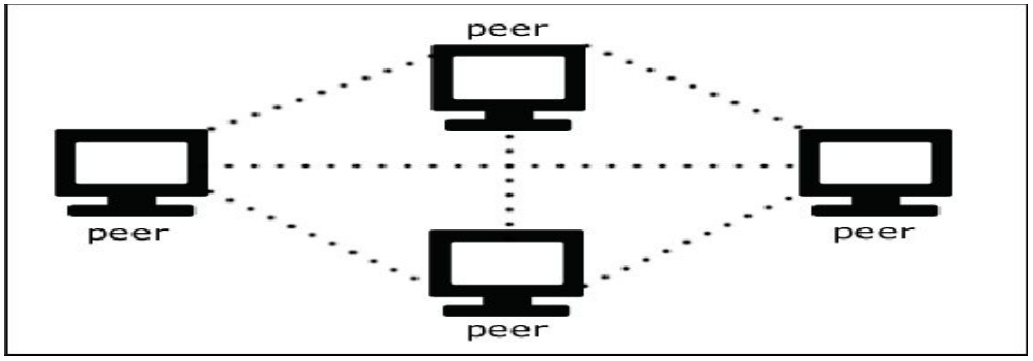
- This type of system requires the client to request a resource, after which the server gives the requested resource. When a client connects to a server, the server may serve multiple clients at the same time.
- Client-Server Systems are also referred to as "Tightly Coupled Operating Systems". This system is primarily intended for multiprocessors and homogenous multicomputer. Client-Server Systems function as a centralized server since they approve all requests issued by client systems.



Types of Distributed system

Peer to peer system

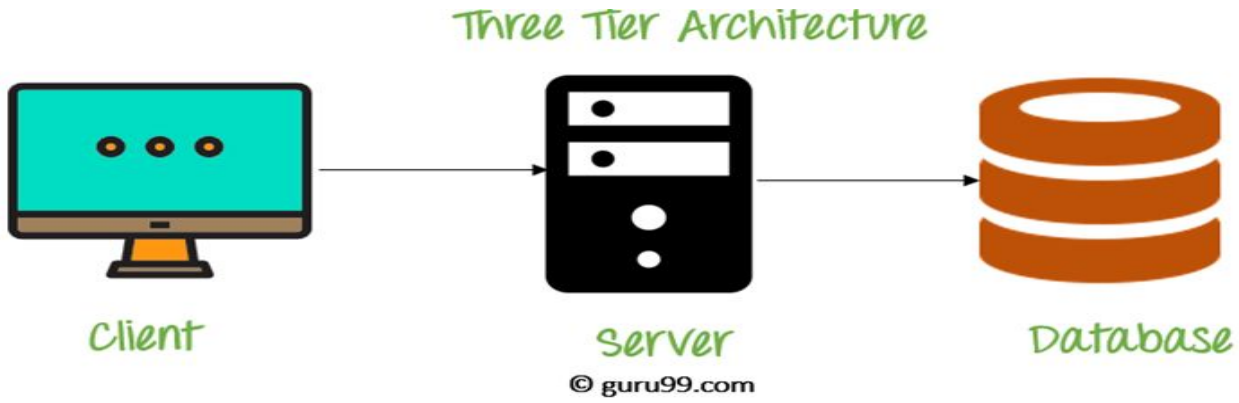
- The nodes play an important role in this system. The task is evenly distributed among the nodes. Additionally, these nodes can share data and resources as needed. Once again, they require a network to connect.
- The Peer-to-Peer System is known as a "Loosely Couple System". This concept is used in computer network applications since they contain a large number of processors that do not share memory or clocks. Each processor has its own local memory, and they interact with one another via a variety of communication methods like telephone lines or high-speed buses.



Types of Distributed system

Three -tire

As a simple example of 3-tier architecture, suppose you are looking to find movie times in your area using a web application. First, the presentation layer displays a web page with some fields for you to enter, like the date you want to view the movie and your zip code. This information is then passed to the application layer, which formats a query and passes it to the database layer



Types of Distributed system

N-tire

When a server or application has to transmit requests to other enterprise services on the network, n-tier systems are used.