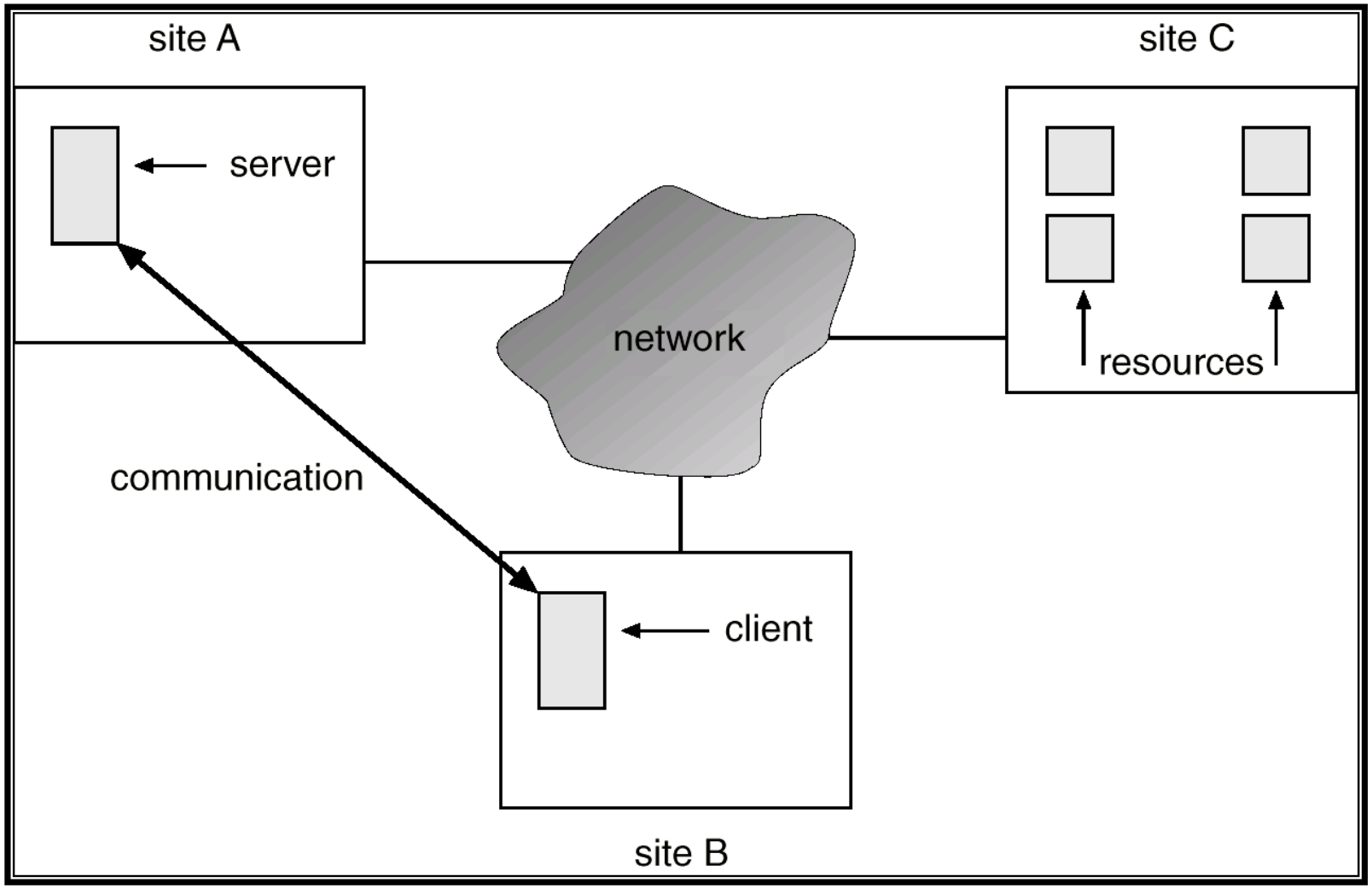
Distributed Systems

Definition of a Distributed System

- A distributed system is a collection of independent computers that appear to the users of the system as a single computer
- Two aspects:
  - Hardware: autonomous machines
  - Software: the users think of the system as a single computer

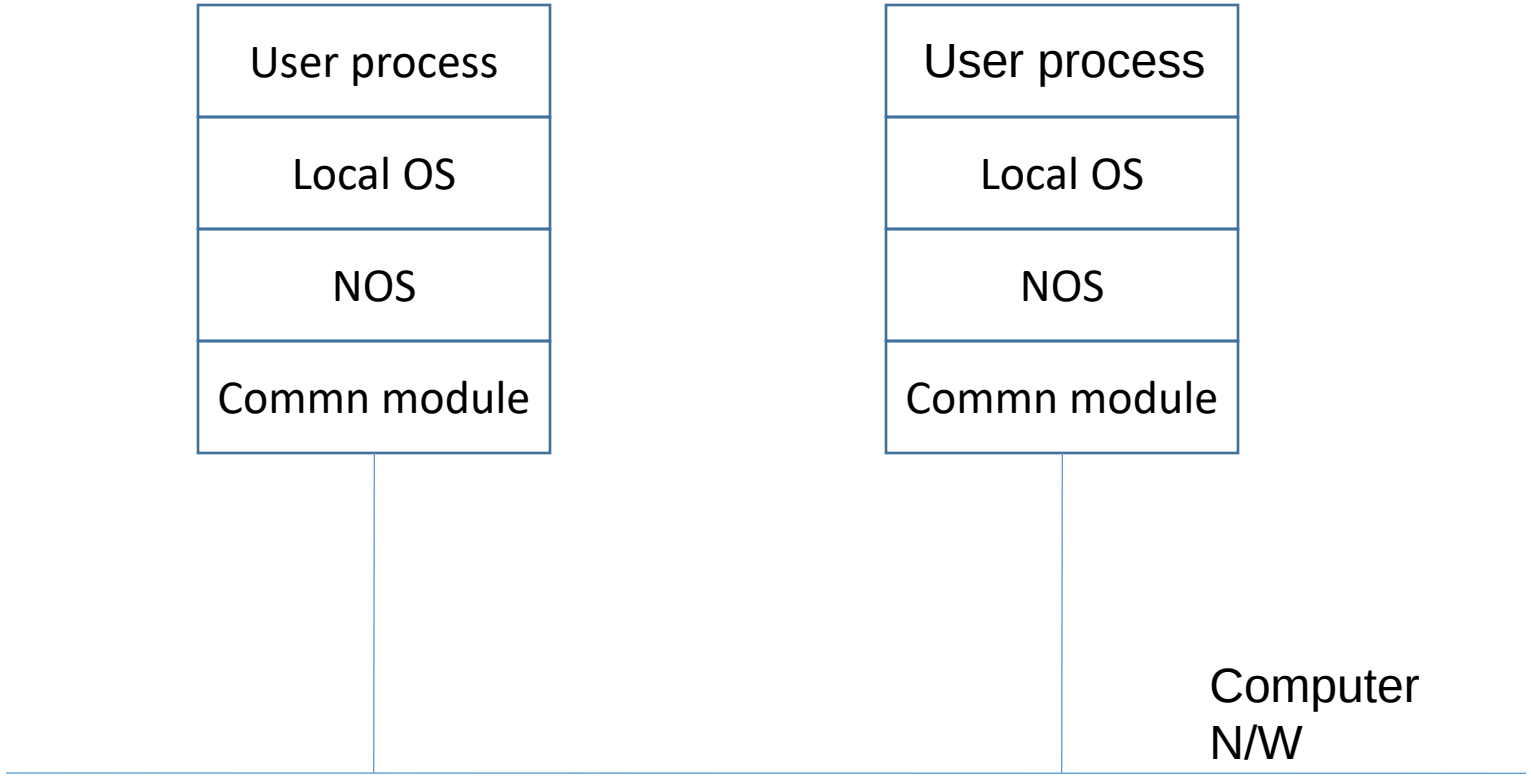
A Distributed System



Motivation

- Resource sharing
  - sharing and printing files at remote sites
  - processing information in a distributed database
  - using remote specialized hardware devices
- Computation speedup – *load sharing*
- Reliability – detect and recover from site failure, function transfer, reintegrate failed site
- Communication – message passing

Network OS


User process
Local OS
NOS
Commn module
User process
Local OS
NOS
Commn module
Computer
N/W

Network OS

- Users are aware of multiplicity of machines. Access to resources of various machines is done explicitly by:
  - Remote logging into the appropriate remote machine.
  - Transferring data from remote machines to local machines, via the File Transfer Protocol (FTP) mechanism.

Characteristics

- Local OS
- Own computer or designated computer or remote login
- Explicit file transfer commands
- Little or no fault tolerance

Communication and information sharing

- Shared Global File system
- Use of file servers
- Hierarchical file system

Distributed OS

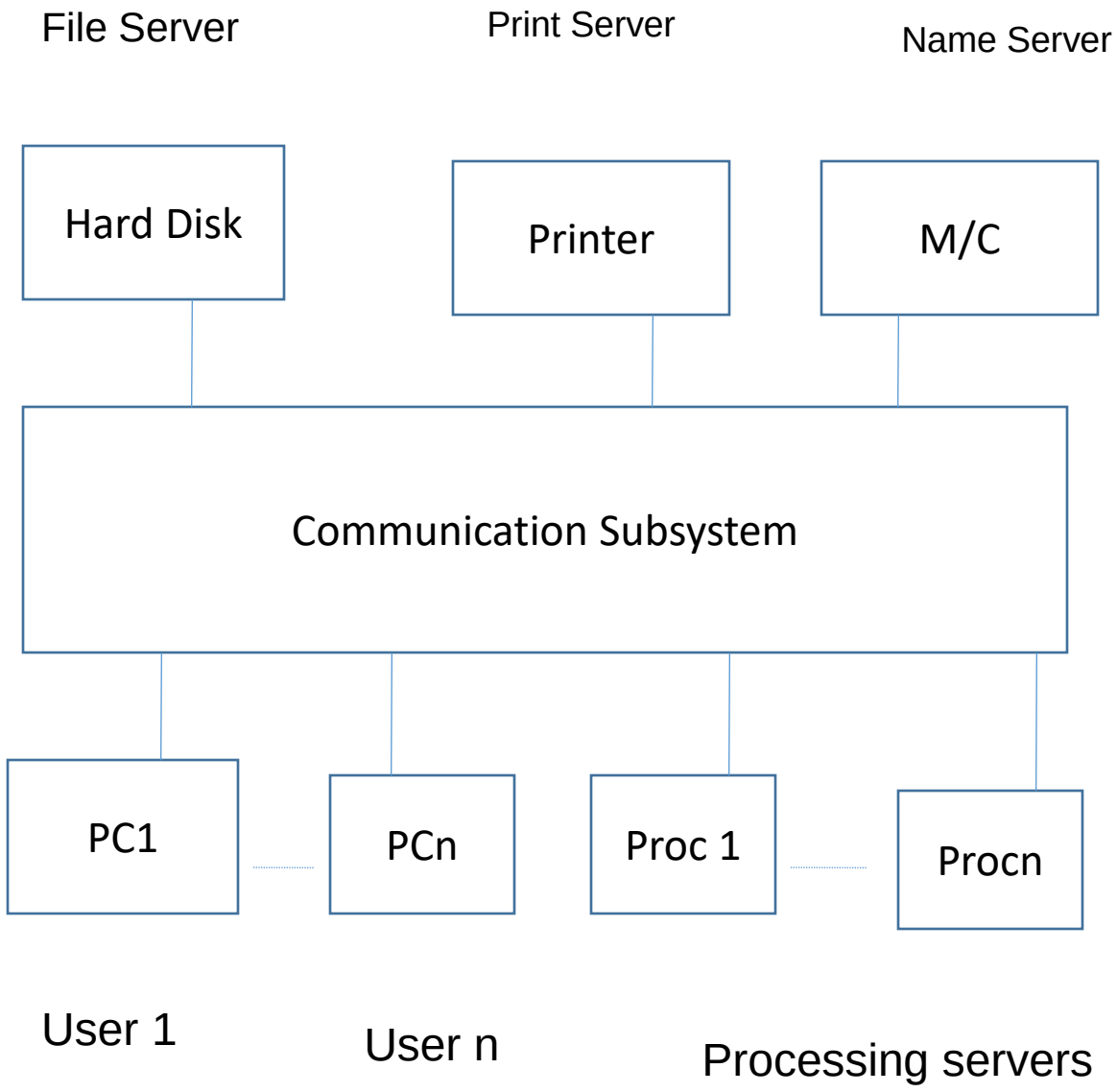
Users not aware of multiplicity of machines. Access to remote resources similar to access to local resources.

- Data Migration – transfer data by transferring entire file, or transferring only those portions of the file necessary for the immediate task.
- Computation Migration – transfer the computation, rather than the data, across the system.

- Process Migration – execute an entire process, or parts of it, at different sites.
  - Load balancing – distribute processes across network to even the workload.
  - Computation speedup – subprocesses can run concurrently on different sites.
  - Hardware preference – process execution may require specialized processor.
  - Software preference – required software may be available at only a particular site.
  - Data access – run process remotely, rather than transfer all data locally.

A DOS should

- Control n/w resource allocation
- Provide virtual computer
- Hide the distribution of resources
- Provide protection mechanisms
- Provide secure communication


File Server
Print Server
Name Server
Hard Disk
Printer
M/C
Communication Subsystem
PC1
PCn
Proc 1
Procn
User 1
User n
Processing servers

Characteristics

- Global system wide OS
- Dynamic allocation of processes to CPU’s
- File placement managed by OS
- Fault tolerance
- Single Global IPC mechanism

Middleware based Distributed Systems

- Middleware- Additional layer on top of NOS implementing general purpose services and it hides more or less the heterogeneity of the collection of underlying platforms.

Middleware Services:

Communication facilities

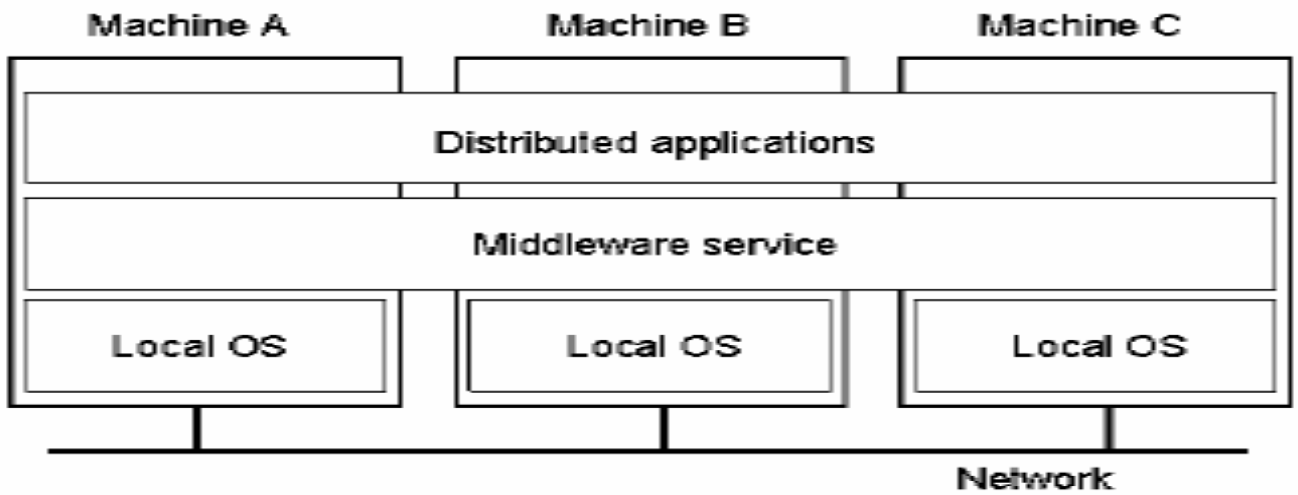
Naming

Persistence

Security

Openness

Scalability



A distributed system organized as middleware.

Goals

- A distributed system should easily connect users to resources
- Why sharing?
- To make it easy for users to access remote resources, and to share them with other users in a controlled way
- Reason for sharing resources: economics

  Typical resources : Printers, computers, storage facilities, data, files
  - Easier to collaborate and exchange information
    – Internet, groupware
- Problems with sharing
  - Security is becoming more and more important as connectivity and sharing increase
  – Tracking communication to build up a preference profile of a specific use
  - Unwanted Communication

Goals

- It should hide the fact that resources are distributed across a network
- It should be open
- It should be scalable

Design Issues

- Transparency – concealment from the user and the application programmer of the separation of components in a distributed systems so that the system is perceived as a whole rather than as a collection of independent components
- Flexibility- ease of modification and ease of enhancement
- Openness – The openness of a computer system is the characteristics that determines whether the system can be extended and re-implemented
- Scalability – A system is scalable, if it will remain effective when there is a significant increase in the numbers of resources and number of users
- Heterogeneity – Variety and differences
- Security- CIA
  - Confidentiality – protection against disclosure to unauthorized individuals
  - Integrity – Protection against alteration or corruption
  - Availability – Protection against interference with the means to access the resources.

Transparency in a Distributed System

Access transparency

- Hide differences in data representation and how a resource is accessed
- Intel (little endian format- *least* significant byte in the smallest address)/Sun SPARC (big endian- most significant byte in the smallest address) (order of bytes)
- OS with different file name conversions

Transparency in a Distributed System

- **Location transparency**

  Hide where a resource is located importance of naming, e.g., URLs
- **Migration transparency**

  Hide that a resource may move to another location
- **Relocation transparency**

  Hide that a resource may move to another location while in use example, mobile users

Transparency in a Distributed System

- **Replication transparency**

  Hide that a resource is replicated subsumes that all replicas have the same name (and thus location transparency)

- **Concurrency transparency**

  Hide that a resource may be shared by several competitive users leave the resource in a consistent state more refined mechanism transactions

Transparency in a Distributed System

- **Failure transparency**

  Hide the failure and recovery of a resource

- **Persistent transparency**

  Hide whether a (software) resource is in memory or Disk

  Important problem: inability to distinguish between a dead resource and a painfully slow one

Design Issues: Transparency

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

Openness

Open distributed system

- Be able to interact with services from other open systems, irrespectively of the underlying environment
- Offers services according to standard rules that describe the **syntax and the semantics of these services**

Rules formalized in protocols

- Services specified through interfaces (described in an Interface Definition Language (IDL) (but only the syntax part)
- Neutral and complete specifications (with regards to a potential implementation)

Openness

- Interoperability: to what extend can work together
- Portability: to what extend an application developed for A can be executed on B that implements the same interface with A

Openness

- Separate Policy from Mechanism
  - A system organized as a collection of relatively small and easily replaceable or adaptable components
  - Provide definitions of the internal parts of the system as well
  - A distributed system provides only **mechanisms.**

Policies are specified by applications and users.

Example policies:

- What level of consistency do we require for client-cached data?
- Which operations do we allow downloaded code to perform?
- Which QoS requirements do we adjust in the face of varying bandwidth?
- What level of secrecy do we require for communication?

Scalability

- size (number of users and/or processes)
- geographical (maximum distance between nodes)
- administrative (number of administrative domains)
- Solution: Powerful server

Scalability Problems

Centralized algorithms - Doing routing based on complete information

Centralized data - A single on-line telephone book

Centralized services - A single server

Major problem : Single point of failure.

Decentralized algorithms

- No complete information about the system state
- Make decision only on local information
- Failure of one machine does not ruin the algorithm
- No assumption of a global clock

Scalability

- Synchronous communication

  In WAN, Unreliable and point-to-point

- Geographical scalability

  How to scale a distributed system across multiple, independent administrative domains: conflicting policies with respect to resource usage (and payment), management and security

- Expand to a new domain

  Protect itself against malicious attacks from the new domain

  The new domain has to protect itself against malicious attacks from the distributed system

Scaling Techniques

- Hiding communication latencies
- Distribution
- Replication

Scaling Techniques

Hiding communication latencies:

- Try to avoid waiting for responses to remote service requests as much as possible
- asynchronous communication (do something else)
- moving part of the computation to the client process

Scaling Techniques

Distribution

- Taking a component, splitting into smaller parts, and spreading these parts across the system
- Example:
  - (1) The World Wide Web
  - (2) Domain Name Service (DNS)
- Hierarchically organized into a tree of domains
- Each domain divided into non overlapping zones
- The names in each domain handled by a single name server

Replication
Caching (client-driven)

- increase availability
- balance the load
- reduce communication latency
- but, consistency problems

Flexibility

Monolithic kernel

The collective kernel structure, Includes file system, directory, and process management

Micro-kernel

- An interprocess communication mechanism
- Some memory management
- A small amount of low-level process management and scheduling
- Low level input/output
- File server
- Directory server
- Process server

Reliability / *Dependability*

- Fault tolerance

  Redundancy techniques

  Distributed control
- Fault avoidance

  Design of components
- Fault detection and recovery

  Atomic transactions, stateless servers, ack and time out based retransmission.

- Performance
  - Response time, throughput, system utilization, bandwidth requirements
- Global knowledge
  - No shared memory, no global clock, how to find global state