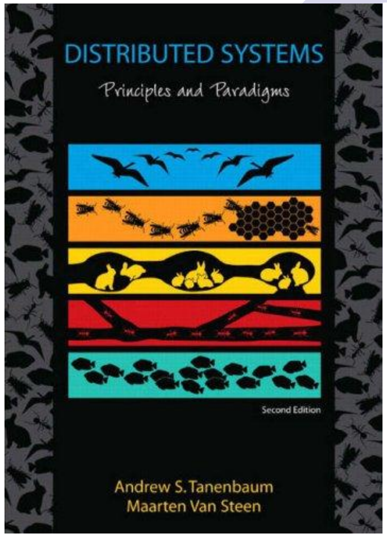
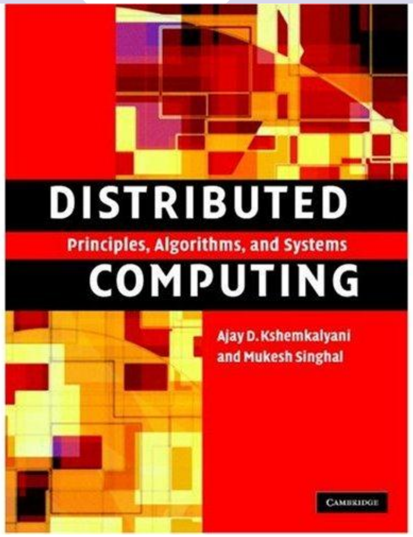
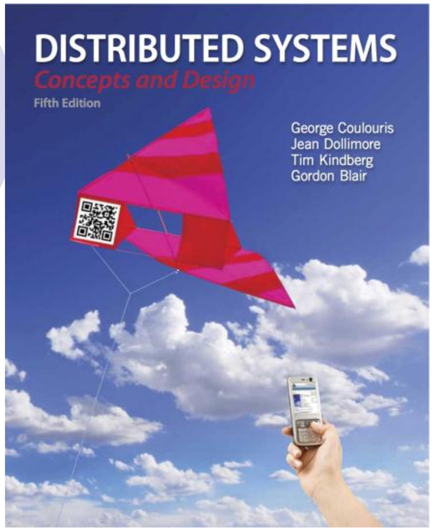
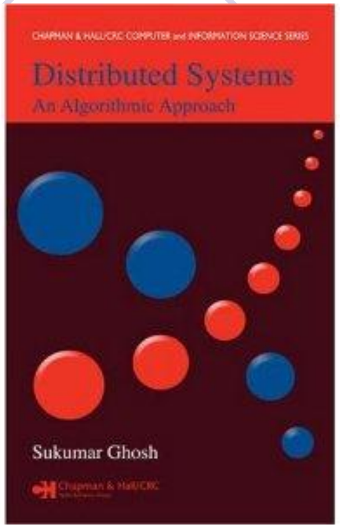
Principles and characteristics of distributed systems and environments









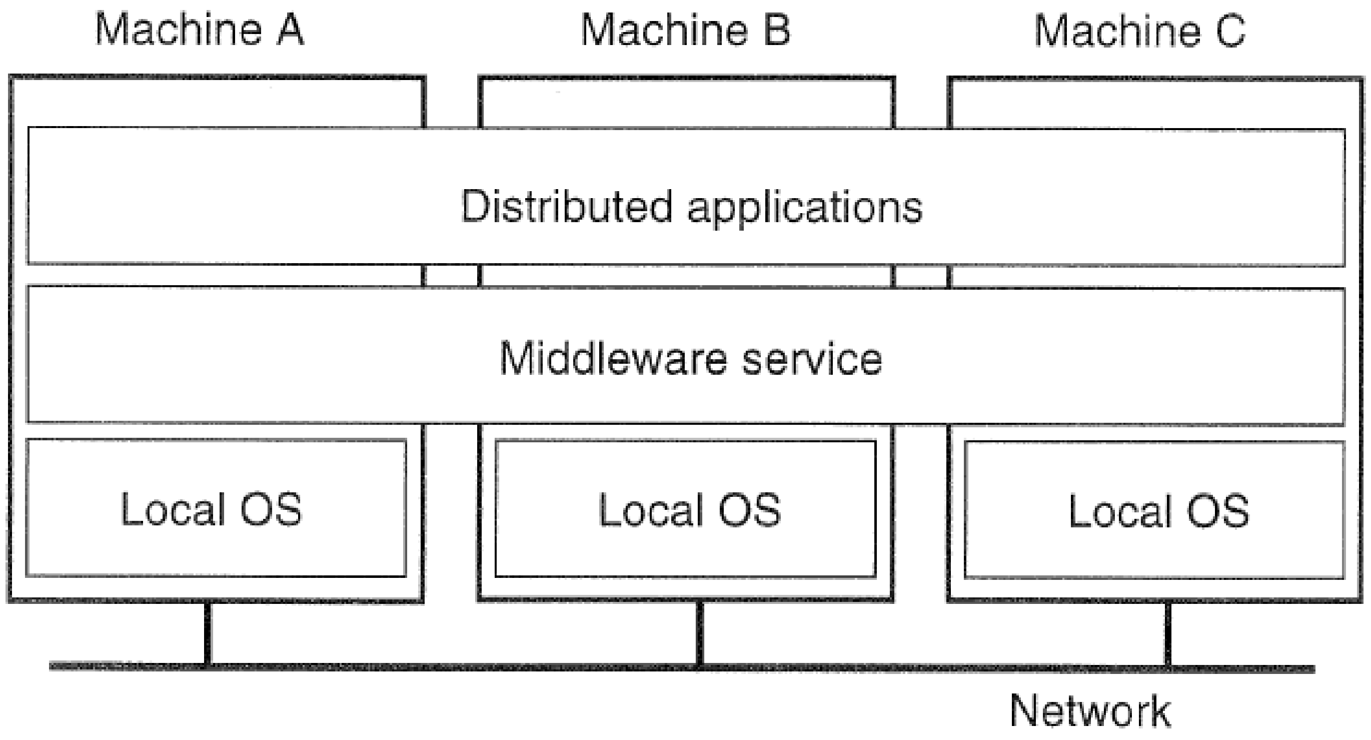
Definition of a distributed system

- ***Distributed system*** *is a collection of independent computers that appears to its users as a single coherent system.*
- ***Distributed computing*** *is decentralised and parallel computing, using two or more computers communicating over a network to accomplish a common objective or task. The types of hardware, programming languages, operating systems and other resources may vary drastically.*
- ***Leslie Lamport****: A distributed system is one I which the failure of a computer you didn’t even know existed can render your computer unusable.*

Definition cont.

- 2 aspects
  - Deals with hardware: machines are autonomous
  - Deals with software: the users think they are dealing with a single system
- Important characteristics
  - Communication is hidden from users
  - Applications can interact with uniform and consistent way
  - Relatively easy to expand or scale (DS will normally be continuously available)
  - A distributed system should be functionally equivalent to the systems of which it is composed.

Distributed system as Middleware service



If the system as a whole looks and acts like a classical single-processor timesharing system, it qualifies as a distributed system.

Goals

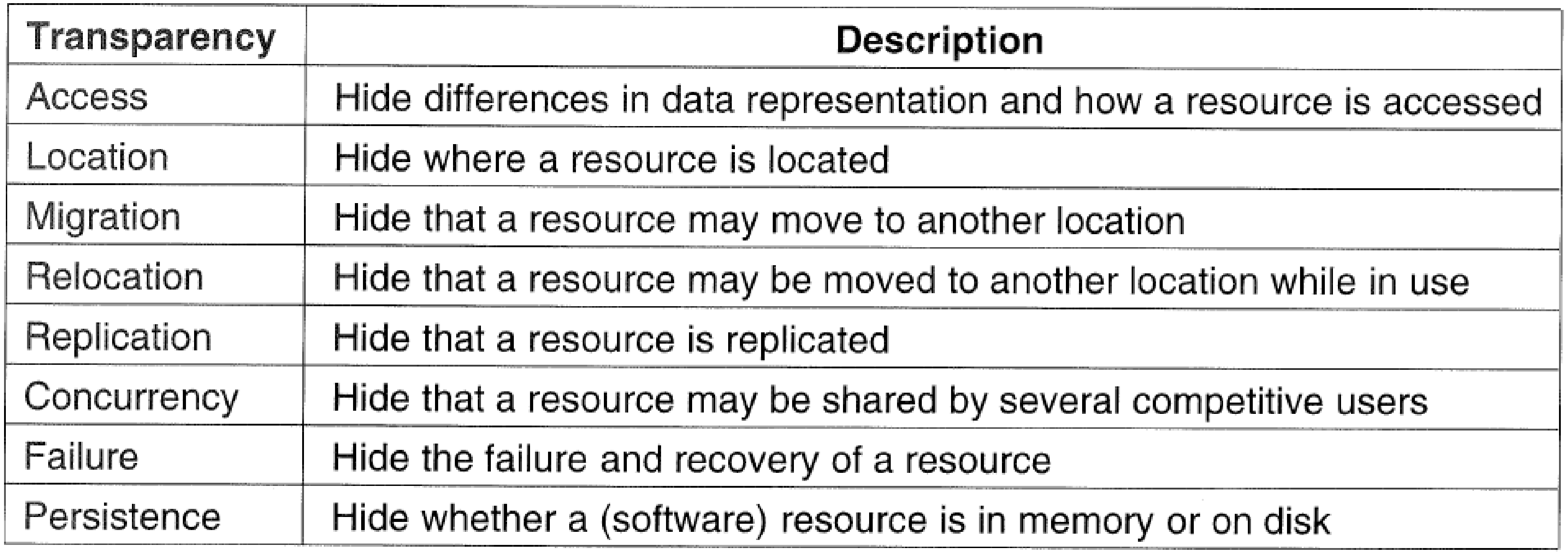
- A DS should easily connect users to resources; it should hide the fact that resources are distributed across a network; should be open and scalable.
- 4 goals:
  - Connecting users and resources
  - Transparency
  - Openness
  - Scalability

Connecting users and resources

- Share resources (hw components, data)
- Groupware
- Security aspects

Transparency

- Transparent ~ present itself to users and applications as if it were only a single computer system



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
Persistence	Hide whether a (software) resource is in memory or on disk

Different forms of transparency in a distributed system
ANSA Reference Manual [1989] and ISO Reference Model
For Open Distributed Processing (RM-ODP) [1992/7]

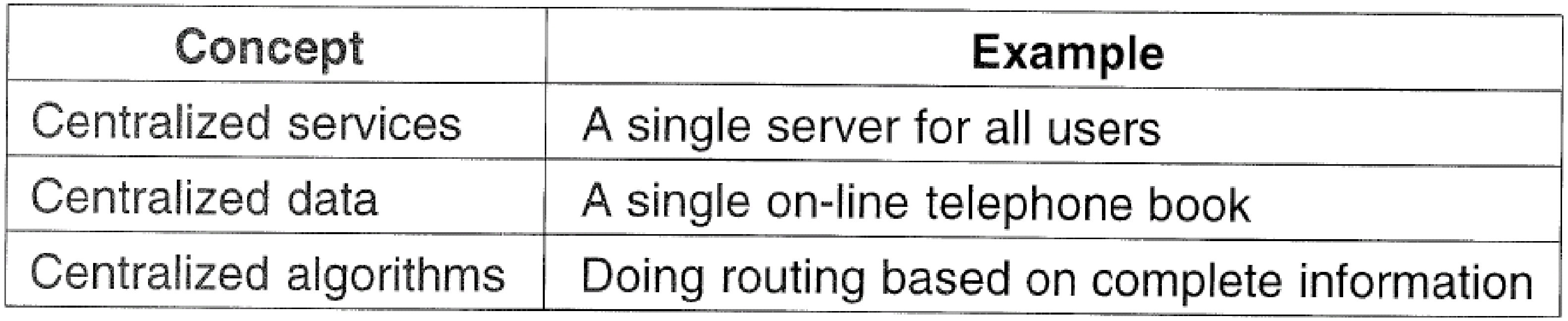
Openness

- System that offers services according to standard rules that describe the syntax and semantics
- Generally specified through **interfaces**, often described in **Interface Definition Language** (**IDL**)
- **Interoperability** – characterizes two implementations of system can co-exists and work together
- **Portability** – characterizes to what extent an application developed for a distributed system *A* can be executed, without modification, on distributed system *B* that implements the same interface as *A*
- Small replaceable and adaptable parts - provide definitions of not only the highest-level interfaces, those used by applications, but also definitions for interfaces to internal parts

Scalability

- Measure at least 3 different dimensions
  - Size – easily add more users and resources to system
  - Geographically scalable system – lie far apart
  - Administratively scalable – independent administrative organizations
- Scalability in one or more dimensions often means loss of performance – should not be significant.

Scalability cont.



Concept	Example
Centralized services	A single server for all users
Centralized data	A single on-line telephone book
Centralized algorithms	Doing routing based on complete information

Example of scalability limitations

- Performance
- Congestion of communication links
- Only decentralized algorithms should be used (no machine has complete information; decisions based on local information; failure of one machine does not ruin the algorithm; no implicit assumption of global clock)

Scaling techniques

- 3 techniques for scaling
  - Hiding communication latencies - Asynchronous communication
  - Distribution - domains
  - Replication – caching -> consistency problems !

Why Distributed Systems?

- Geographically distributed environment
- Speed up
  - Parallel vs. Distributed Systems (synchronous SIMD or MIMD; asynchronous process)
- Resource sharing
  - HW and SW (databases)
- Fault tolerance
  - Detection, Recover

Logical distribution of functional capabilities: multiple processes, interprocess communication, disjoint address space, collective goal

Issues in Distributed Systems

- Knowledge of process
  - Identity, Identity of it’s immediate neighbors, communication channel to neighbors
- Network topology
  - Completely or sparsely connected, message routing, channel uni/bi-directional
- Degree of synchronization
  - Clock drift, propagation delay
- Failures
  - Type of failure and duration of failure
- Scalabilty
  - Time and space complexity (O(log N) ~ excellent, O(N) ~ pure)

Common Subproblems

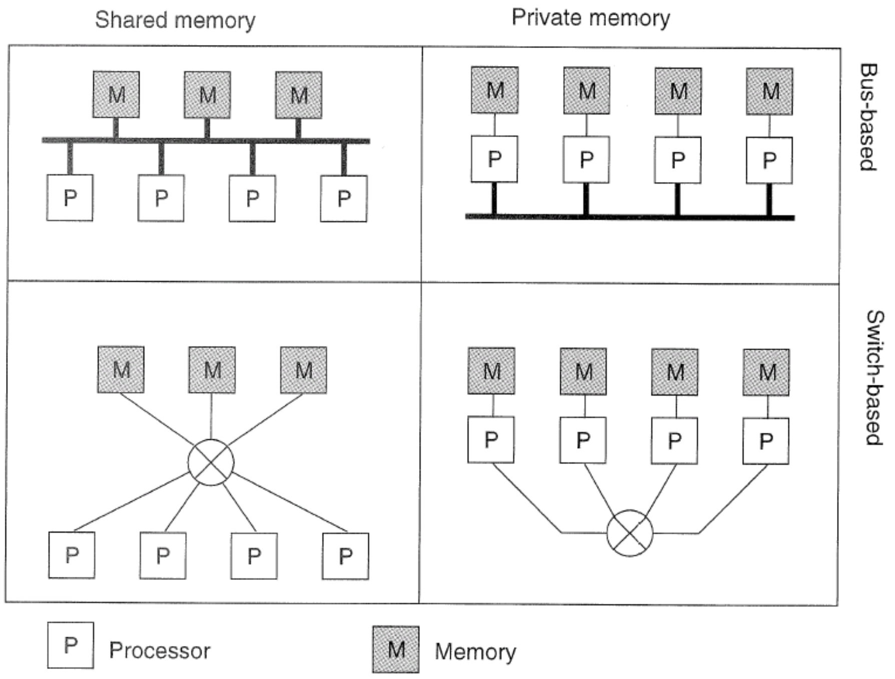
- Leader election
- Mutual exclusion
- Time synchronization
- Global state
- Replica management

RM-ODP

- Reference Model – Open Distributed Processing
- ITU-T X.9xx standards, X.901 – Overview
- Viewpoints (and their languages)
    - Enterprise
    - Information
    - Computational
    - Engineering
    - Technology

Hardware concepts

- DS consist of multiple CPU – several different ways of HW organization
- Multiprocessors – have shared memory
- Multicomputers – without shared memory

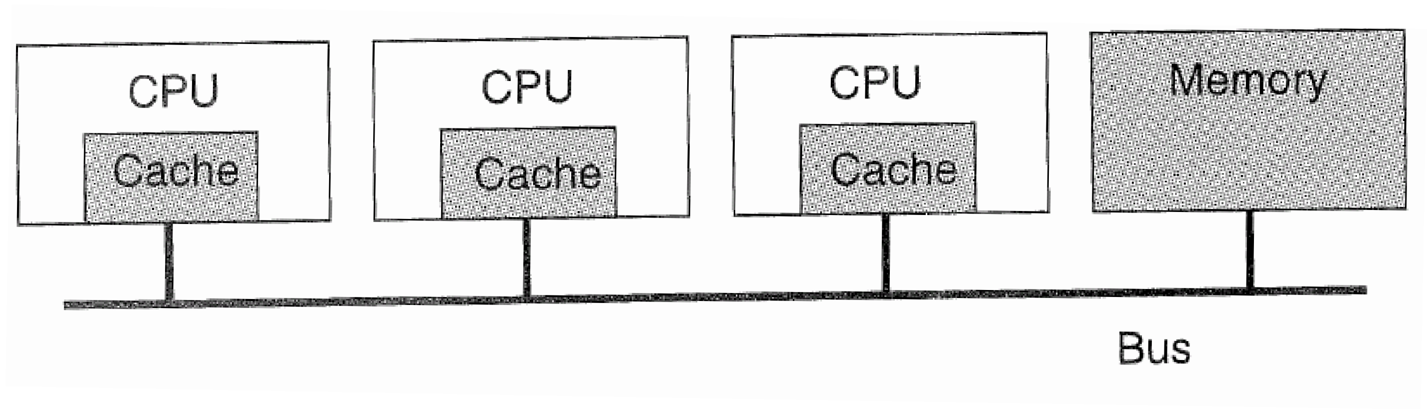


Hardware concepts cont.

- Architecture of interconnection network
  - Bus
  - Switched
- Distributed computer system (significant for multicomputers)
  - Homogenous – single interconnection network
  - Heterogeneous – different networks (LAN connected through FDDI, ATM)

Multiprocessors

- **Coherent**
- Overloaded bus –> cache memory between CPU and bus (hit rate)
- Bus suitable until 256 CPUs

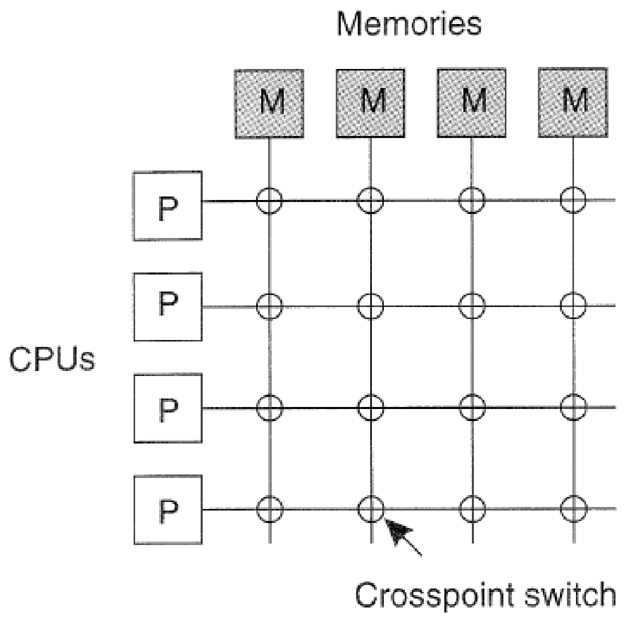


Multiprocessors cont.

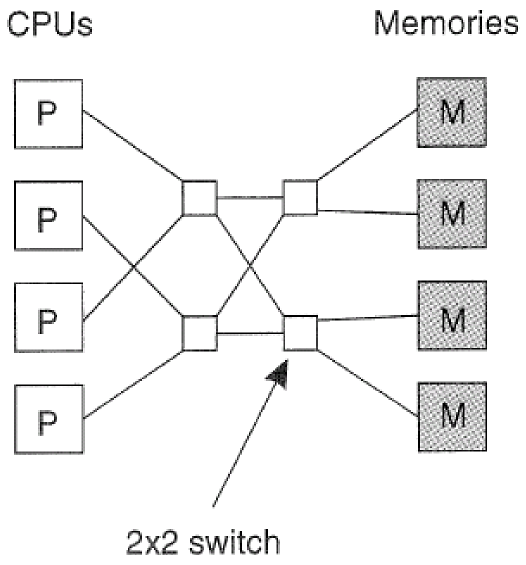
- Crossbar switch – crosspoint switch
  - *n* CPUs and *n* memories = n^2 crosspoint switches
- Omega network
  - 2x2 switches, each two inputs and two outputs
  - Fast switching, not cheap
- NUMA (NonUniform Memory Access)
  - Fast access to local memory, slow to other's
  - Better average access times than omega networks

Multiprocessors cont.

Crossbar switch



Omega network



Homogenous Multicomputer Systems

- Relative easy acording multiprocessors
- CPU-to-CPU and CPU-to-memory communication
- **System Area Networks** (**SANs**)
  - High performance interconnection network
- Bus-based – Fast Ethernet 25-100 nodes
- Switch-based – messages routed
  - Mesh, hypercube
  - Vary widely from **Massively Parallel Processors** (**MPPs**) to **Clusters of Workstations** (**COWs**)

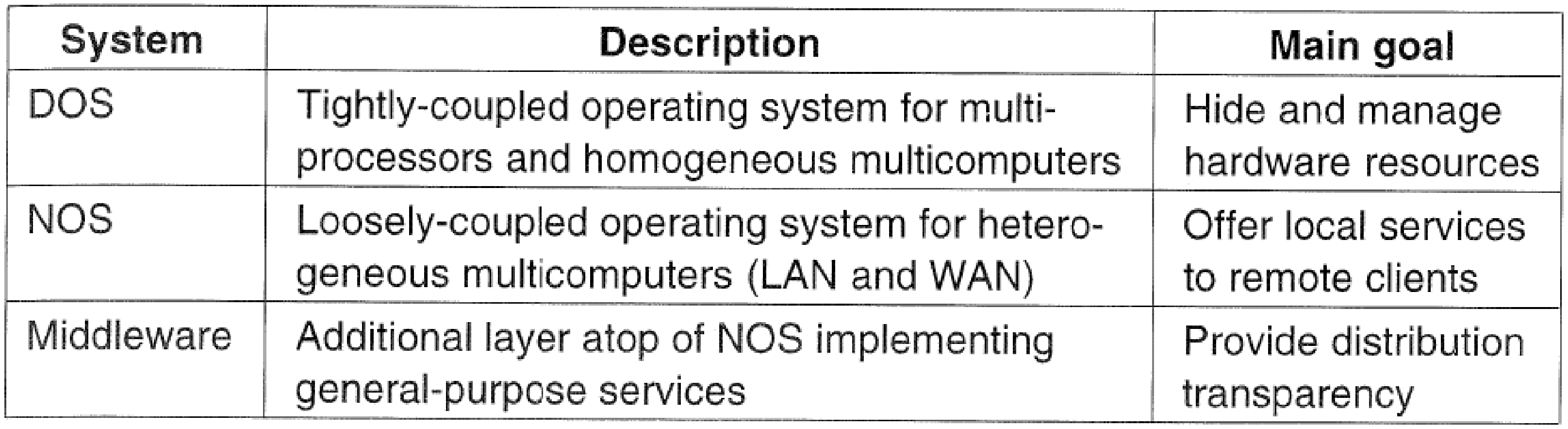
Heterogenous Multicomputer Systems

- Most distributed systems build on it
- Due to scale, inherent heterogeneity, and most of all, lack of global system view, sophisticated software is needed to build apps for heterogeneous multicomputers.
- Provide transparency for apps.
- Example
  - DAS http://www.cs.vu.nl/~bal/das.html
  - I-way project

Software concepts

- Determines what a DS actually looks like
- **Resource manager** for underlying hardware
- Operating systems for distributed computers
  - **Tightly-coupled** – distributed operating system (multiprocessors, homogenous multicomputer systems)
  - **Loosely-coupled –** network operating system (heterogeneous multicomputer systems)
  - Middleware – enhancements to services of network operating systems – distribution transparency

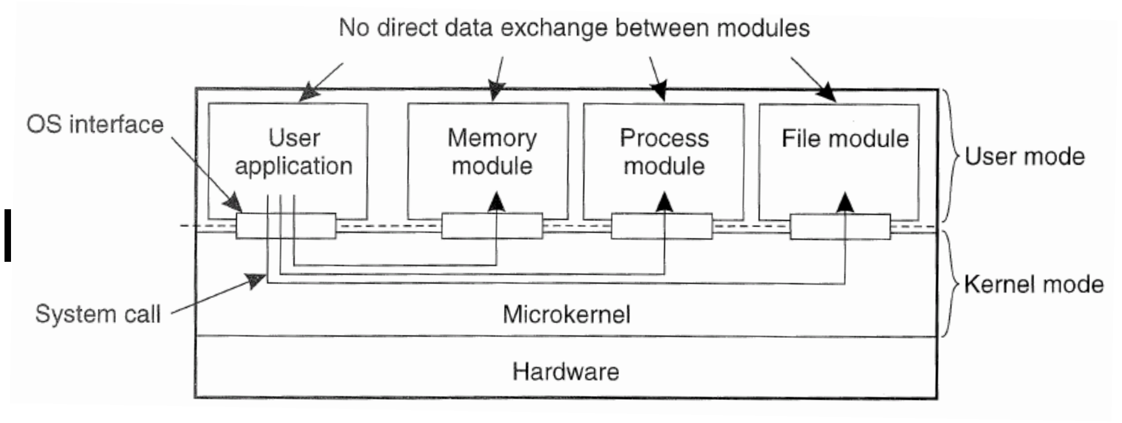
Software concepts cont.



System	Description	Main goal
DOS	Tightly-coupled operating system for multi-processors and homogeneous multicomputers	Hide and manage hardware resources
NOS	Loosely-coupled operating system for hetero-geneous multicomputers (LAN and WAN)	Offer local services to remote clients
Middleware	Additional layer atop of NOS implementing general-purpose services	Provide distribution transparency

Uniprocessor Operating Systems

- Implement a **virtual machine**
- Apps protected each other
- OS should be in full control of how hw resources are used and shared
  - **Kernel mode**
  - **User mode**
- Architecture
  - Monolithic kernel
  - Microkernel



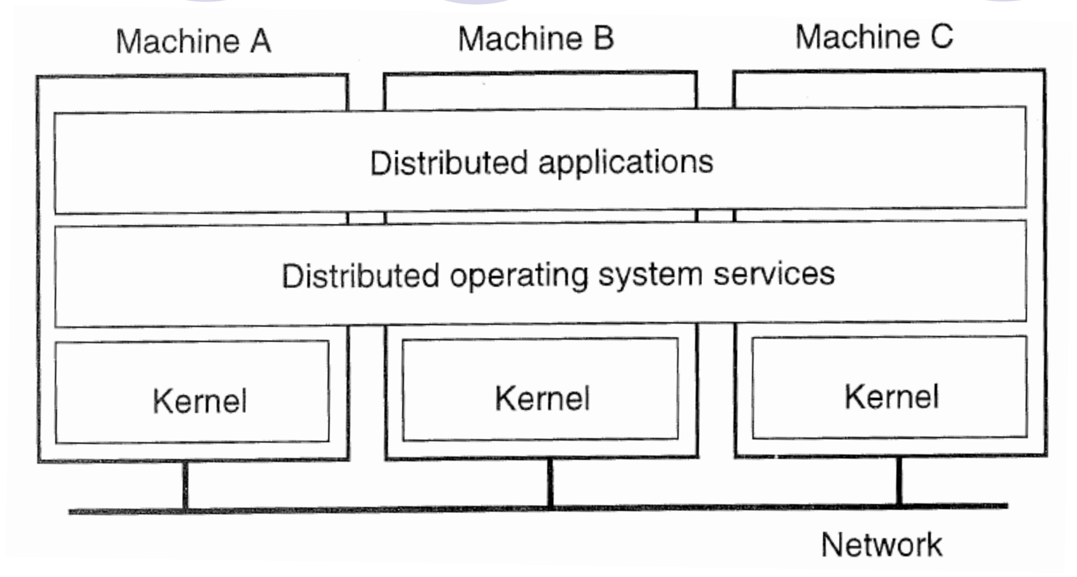
Multiprocessor Operating Systems

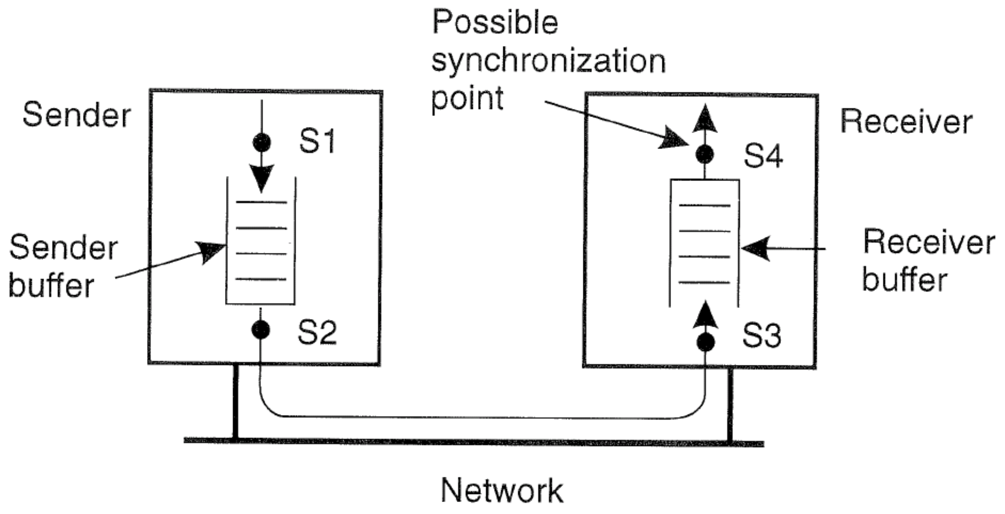
- Support for multiple CPUs having access to shared memory
- IPC done through memory – protect against simultaneous access
- Synchronization primitives
  - **Semaphore** – **atomic** operations down and up
  - **Monitor** – private variable and public procedures (operations)

Multicomputer Operating Systems

- Different structure and complexity than multiprocessor operating systems
- Communication done by **message passing**
- *Software implementation* of shared memory – not always
- Message passing primitives may vary widely in different OS

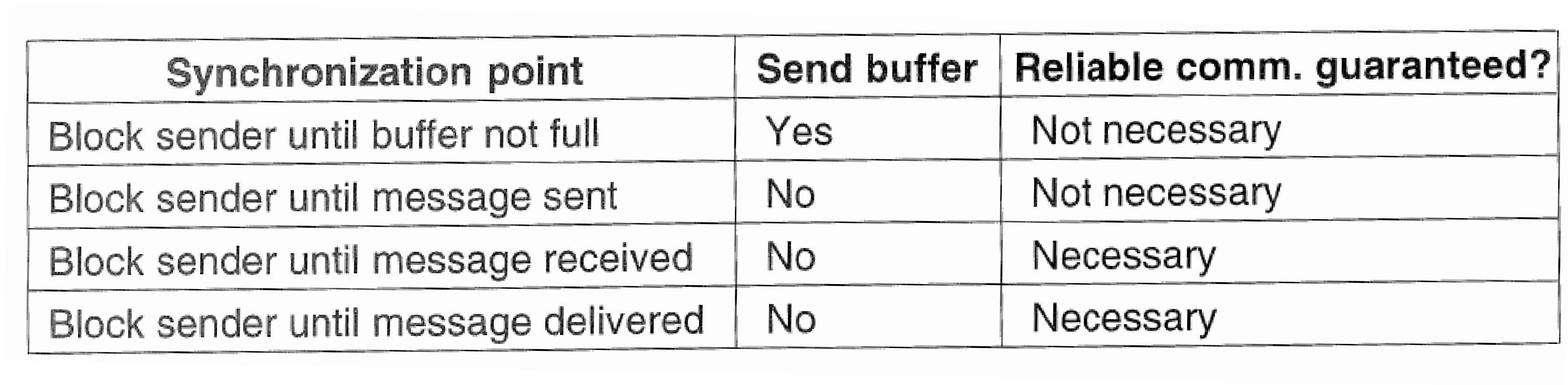
Multicomputer Operating Systems cont.





Multicomputer Operating Systems cont.

- Reliability of message delivery ?

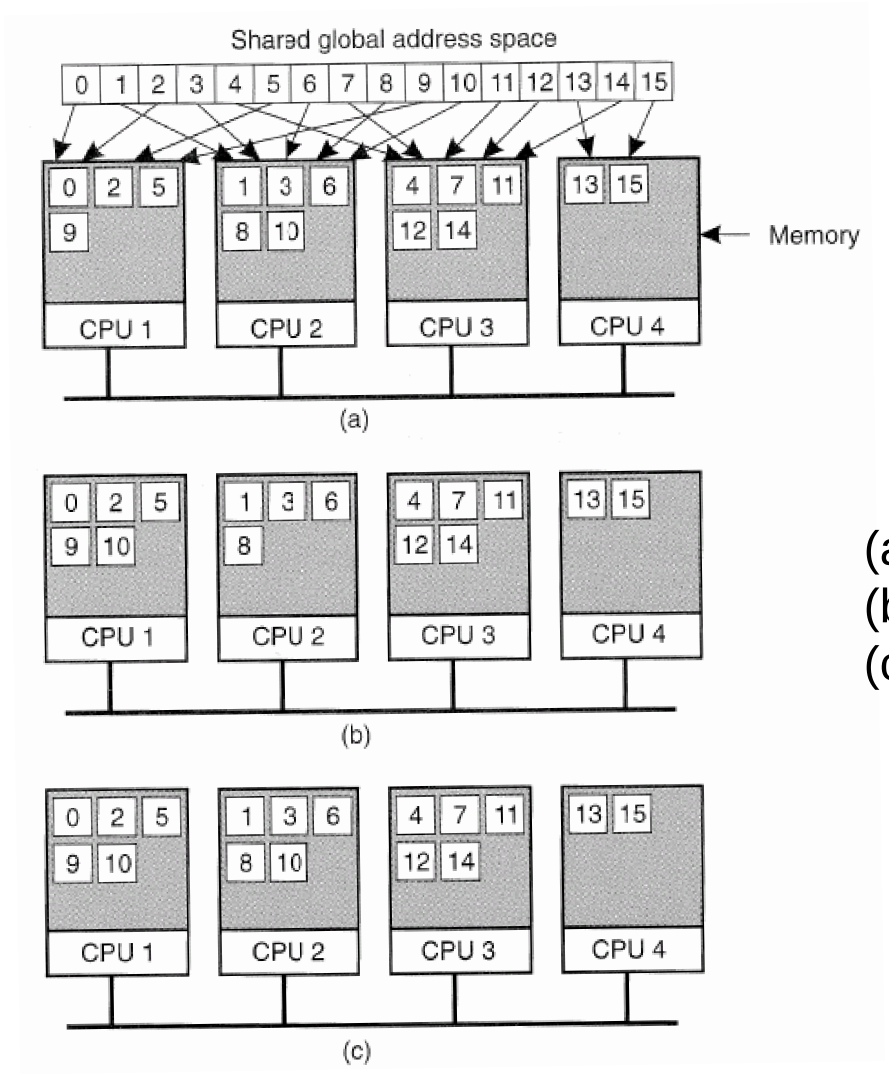


Synchronization point	Send buffer	Reliable comm. guaranteed?
Block sender until buffer not full	Yes	Not necessary
Block sender until message sent	No	Not necessary
Block sender until message received	No	Necessary
Block sender until message delivered	No	Necessary

Distributed Shared Memory Systems

- Page-based distributed shared memory
  - Address space divided into pages (4kB or 8kB)
  - When CPU references address not present locally, trap occurs, OS fetches the page
  - Essential normal paging

Distributed Shared Memory Systems cont.

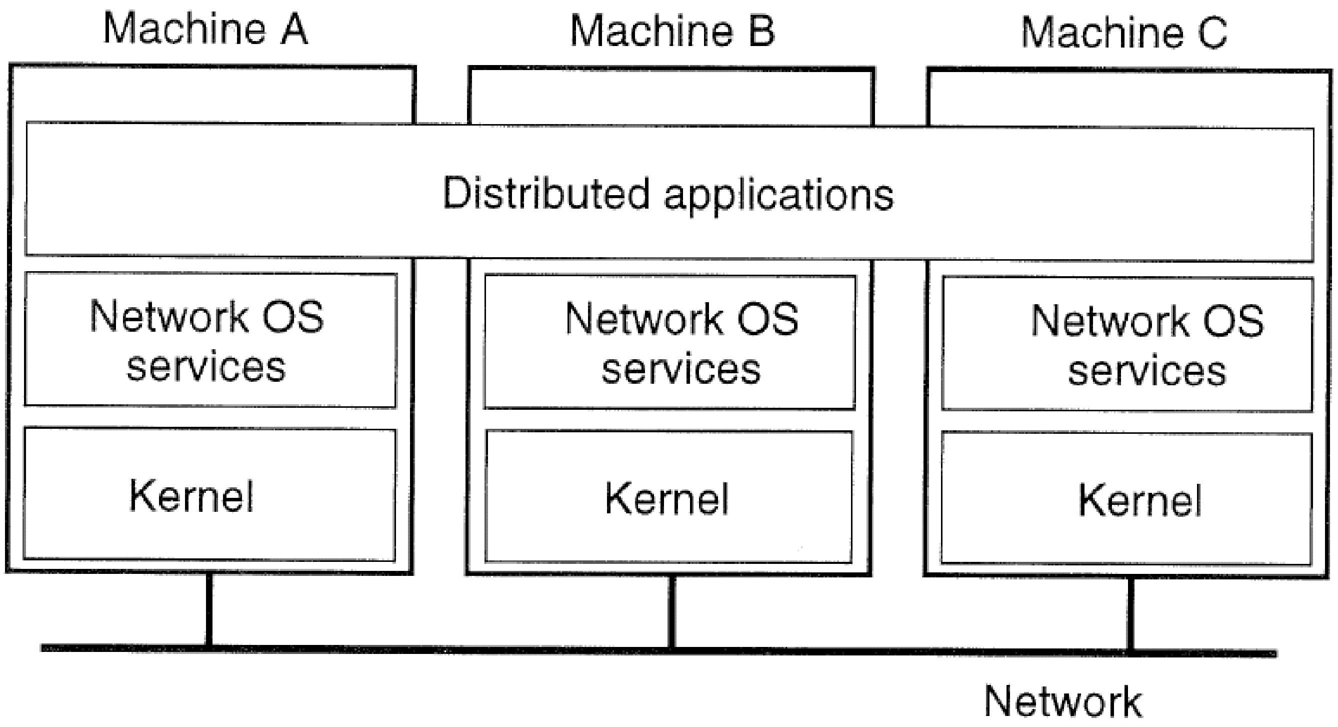


(a) Pages in DSM
(b) CPU 1 references page 10
(c) Situation if page 10 is read only and replication used

Network Operation Systems

- Do not assume that underlying hardware is homogeneous and that it should be managed as if it were a single system
- Facilities to allow users to make use of the services available on specific machine
- Service accessed commonly using commands
  - *rlogin machine*
  - *rcp machine1:file1 machine2:file2*

Network Operation Systems cont.

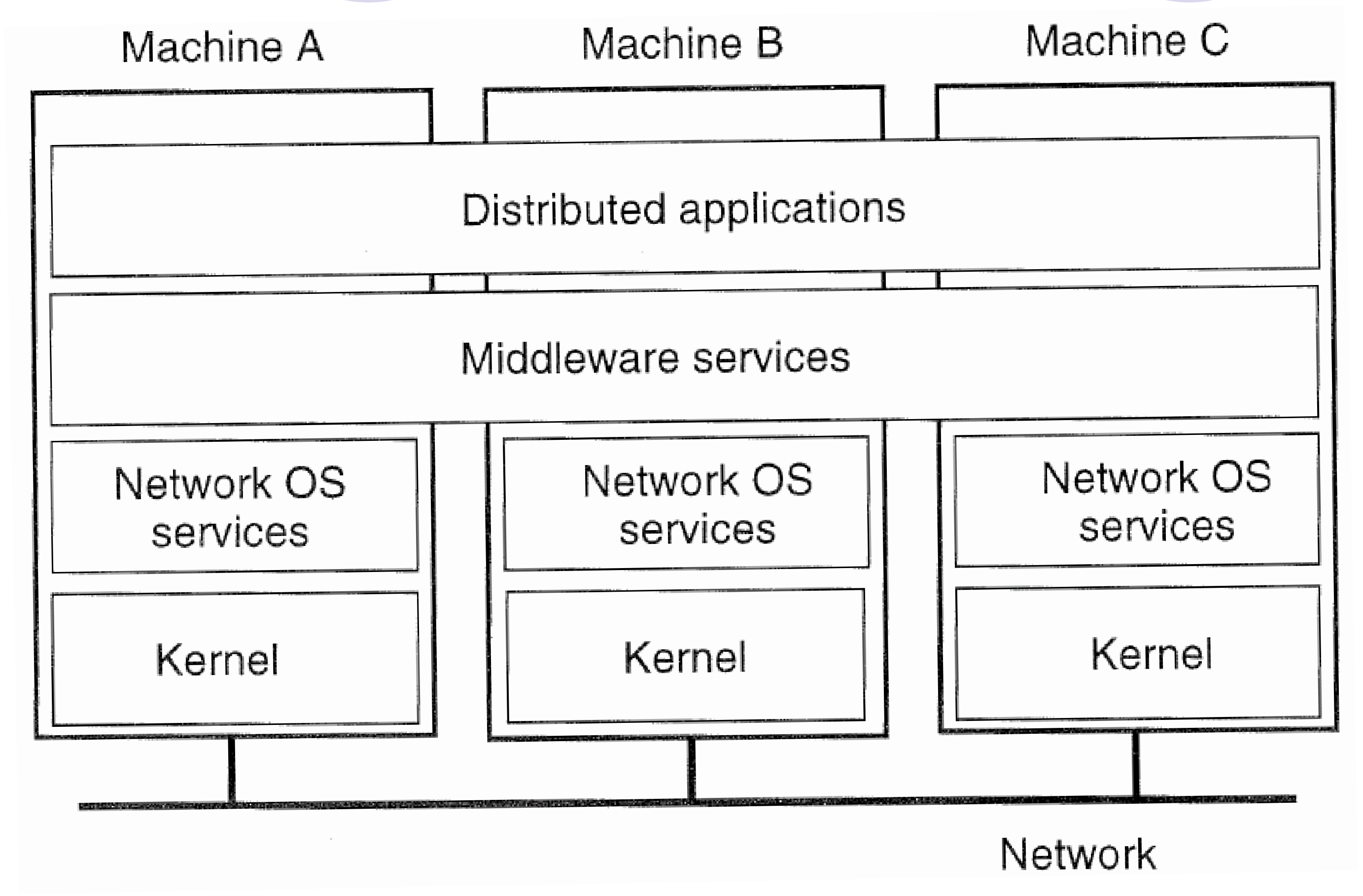


Network operating systems are clearly more primitive than distributed operating systems. The main distinction between the two types of operating systems is that distributed operating systems make serious attempt to realize full transparency, that is, provide a single-system view.

Middleware

- DOS and NOS don’t qualify definition of DS
- Modern distributed systems are constructed by means of an additional layer called **middleware.**
- Additional layer of software between applications and the network operating system

Middleware cont.



Goal is to hide heterogeneity of the underlying platforms from applications

Middleware cont.

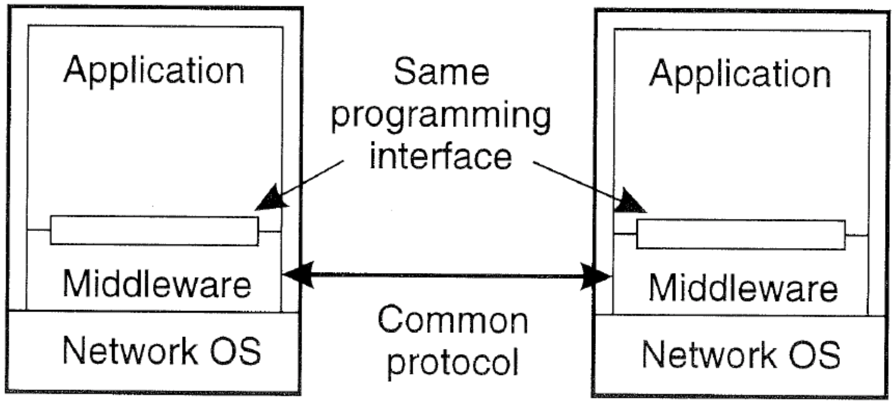
- Middleware models (*paradigm*)
  - Everything as file (simple) – Plan 9, UNIX
  - Distributed File Systems – reasonable scalable which contributes popularity
  - Remote Procedure Calls – unaware of network communication
  - Distributed Objects – unaware of network communication
  - Distributed documents - WWW

Middleware cont.

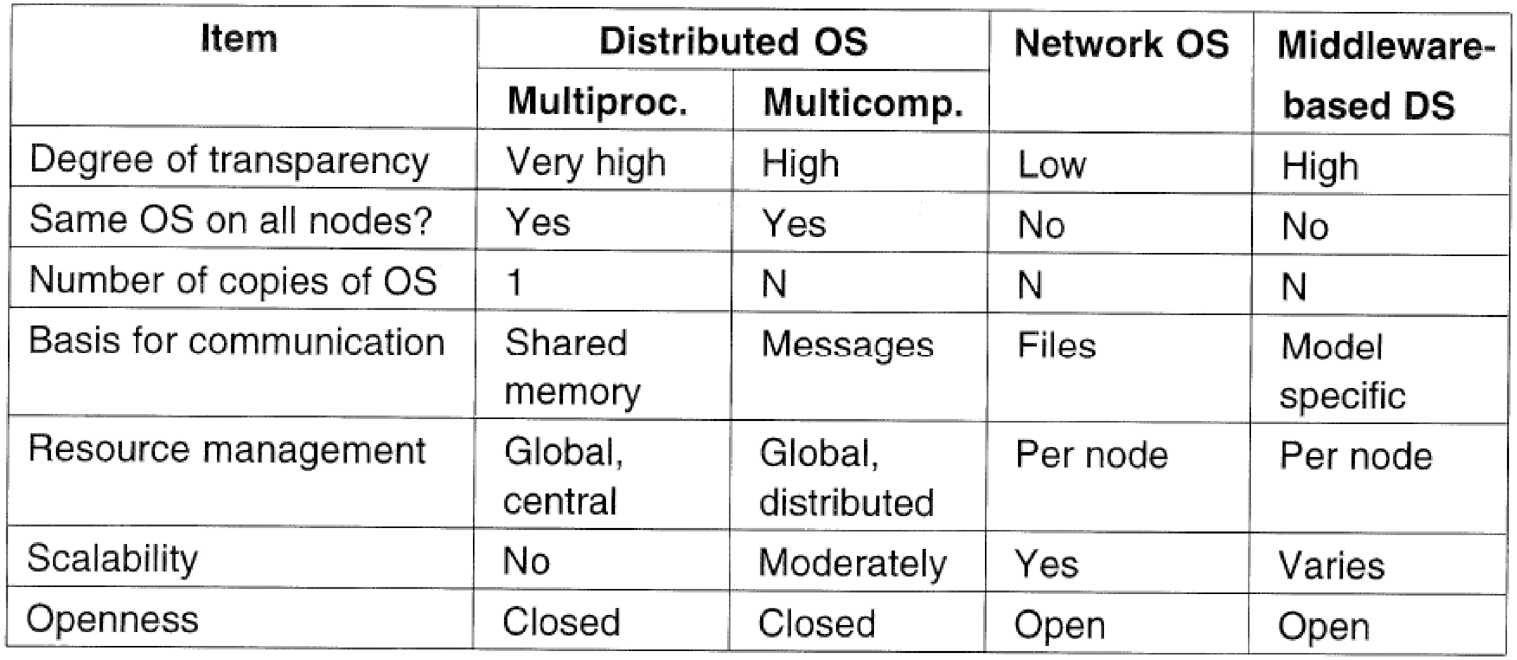
- Middleware services
  - high-level **Communication facilities -** *Access transparency*
  - **Naming** – allows entities to be shared and accessed
  - **Persistence** – through distributed file system or database
  - **Distributed transactions –** atomic multiple read and write operations; write operation succeeds or fails
  - **Security** – can’t rely on the underlying local operating system, must be implemented in middleware, security has turned out to be one of the hardest services to implement

Middleware and Openness

- Modern distributed systems as middleware
- Applications are independent of operating system
- Independence replaced by strong dependency of specific middleware
- Need that middleware protocols and interfaces are the same in different implementations



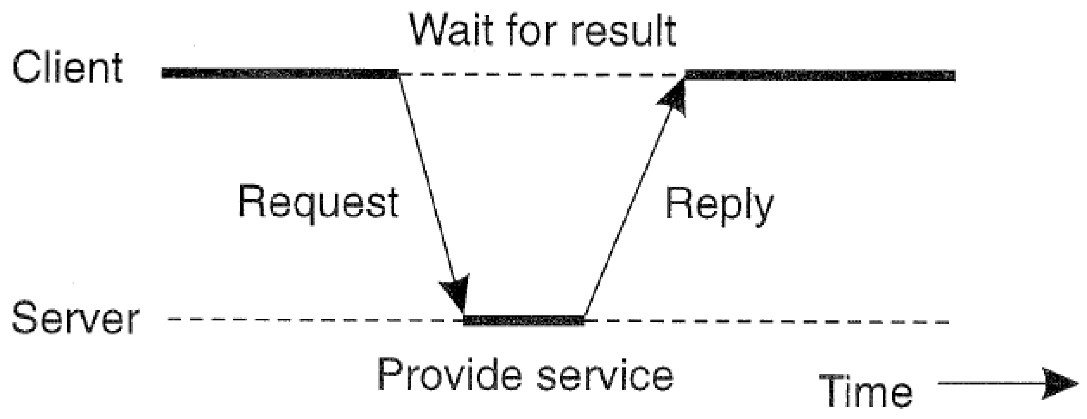
Comparison between Systems



Item	Distributed OS		Network OS	Middleware-based DS
	Multiproc.	Multicomp.		
Degree of transparency	Very high	High	Low	High
Same OS on all nodes?	Yes	Yes	No	No
Number of copies of OS	1	N	N	N
Basis for communication	Shared memory	Messages	Files	Model specific
Resource management	Global, central	Global, distributed	Per node	Per node
Scalability	No	Moderately	Yes	Varies
Openness	Closed	Closed	Open	Open

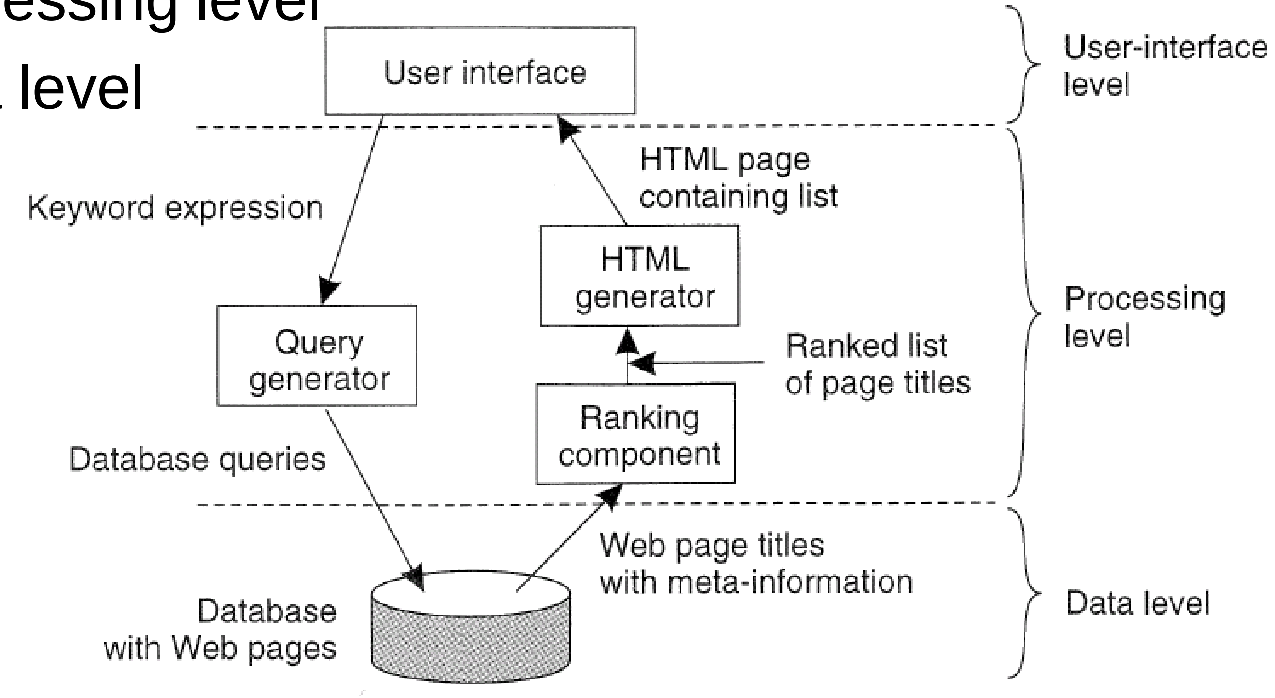
Client – Server Model

- **Server** – process implementing a specific service
- **Client** – process that requests service from server by sending request and waiting for response



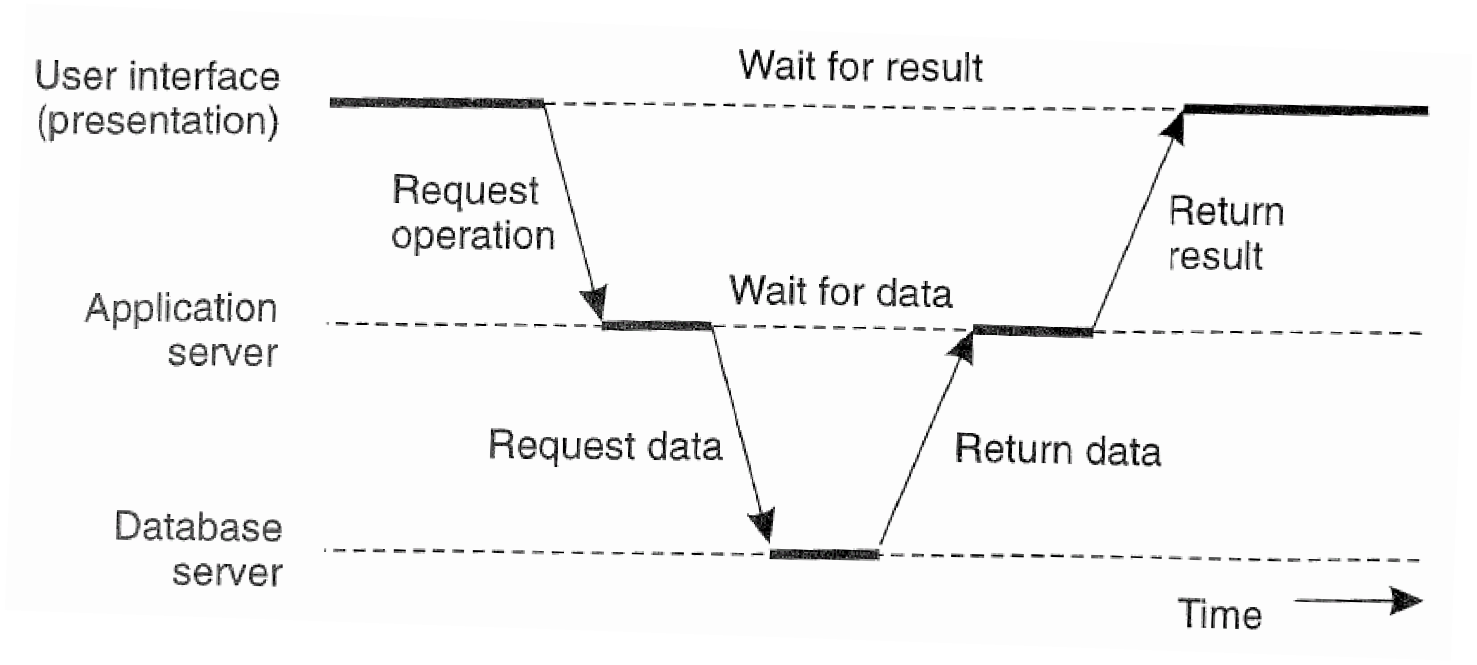
Client – Server Model cont.

- Application layering
  - Distinction of 3 levels
    - The user-interface level
    - The processing level
    - The data level



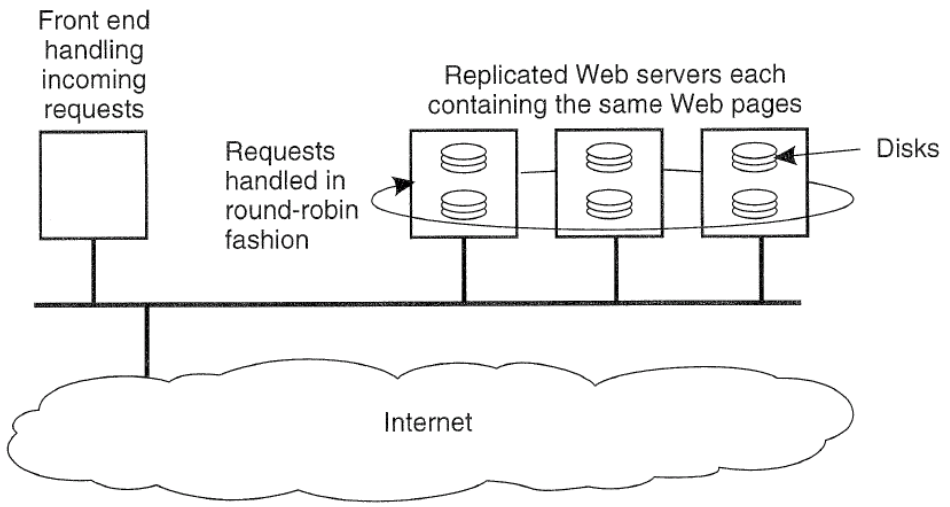
Multitiered architectures

- Three-tier architecture – physically



Modern architectures

- Vertical distribution – multitiered architectures
- Horizontal distribution – distribution split into equivalent parts (clients or servers)


Front end handling incoming requests
Requests handled in round-robin fashion
Replicated Web servers each containing the same Web pages
Disks
Internet

- Peer-to-peer distribution