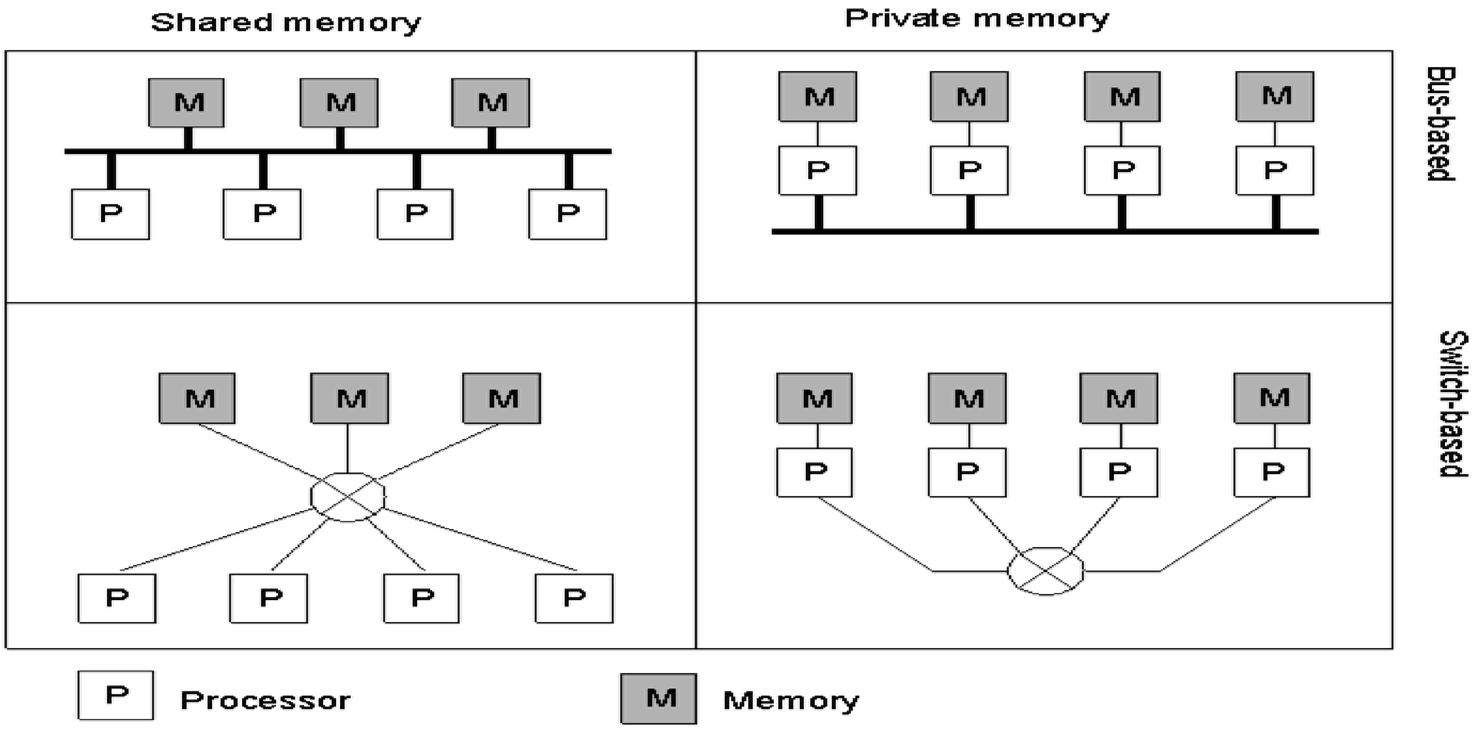
# Hardware Concepts

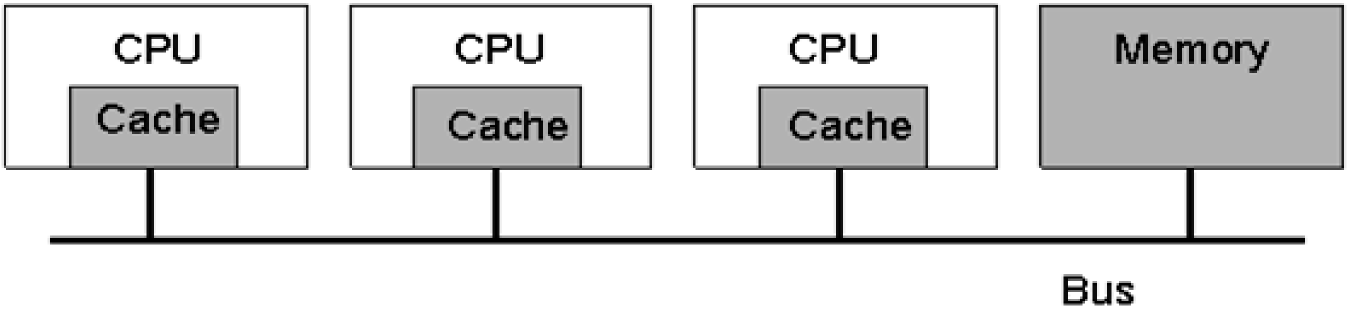
Hardware in Distributed Systems can be organized in 2 different ways: -

- Shared Memory (Multiprocessors , which have a single address space)
- Private Memory (Multicomputers, each CPU has a direct connection to its local memory).



# Multiprocessors (1)

Have limited scalability – Cache Memory help avoid bus overloading

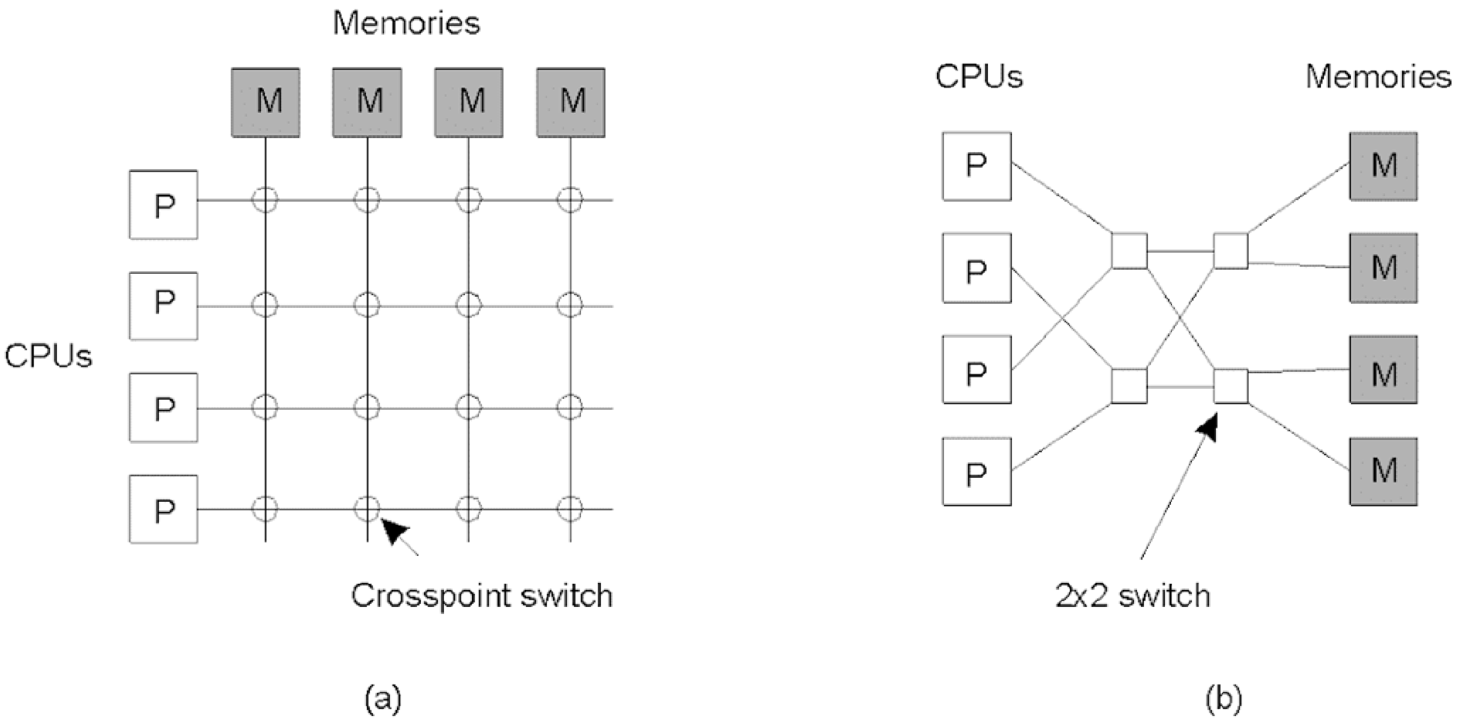


A bus-based multiprocessor.

# Multiprocessors (2)

## Multiprocessors – Switch Based

- Different CPUs can access different memories simultaneously
- The number of switches limits the number of CPUs that can access memory simultaneously



# Multicomputer Systems

- Multicomputers – closely coupled processors that do not physically share memory
  - Cluster computers
  - Networks or clusters of computers (NOWs or COWs)
  - Can grow to a very large number of processors
- Consist of
  - Processing nodes – CPU, memory and network interface (NIC)
  - I/O nodes – device controller and NIC
  - Interconnection network
    - Many topologies – e.g. grid, hypercube, torus
    - Can be packet switched or circuit switched

# Software Concepts

| 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 heterogeneous multicomputers (LAN and WAN) | Offer local services to remote clients |
| Middleware | Additional layer on the top of NOS implementing general-purpose services | Provide distribution transparency |

# Distributed Operating Systems

➢• Act as resource managers for the hardware while attempting to hide intricacies and the heterogeneous nature of the underlying hardware

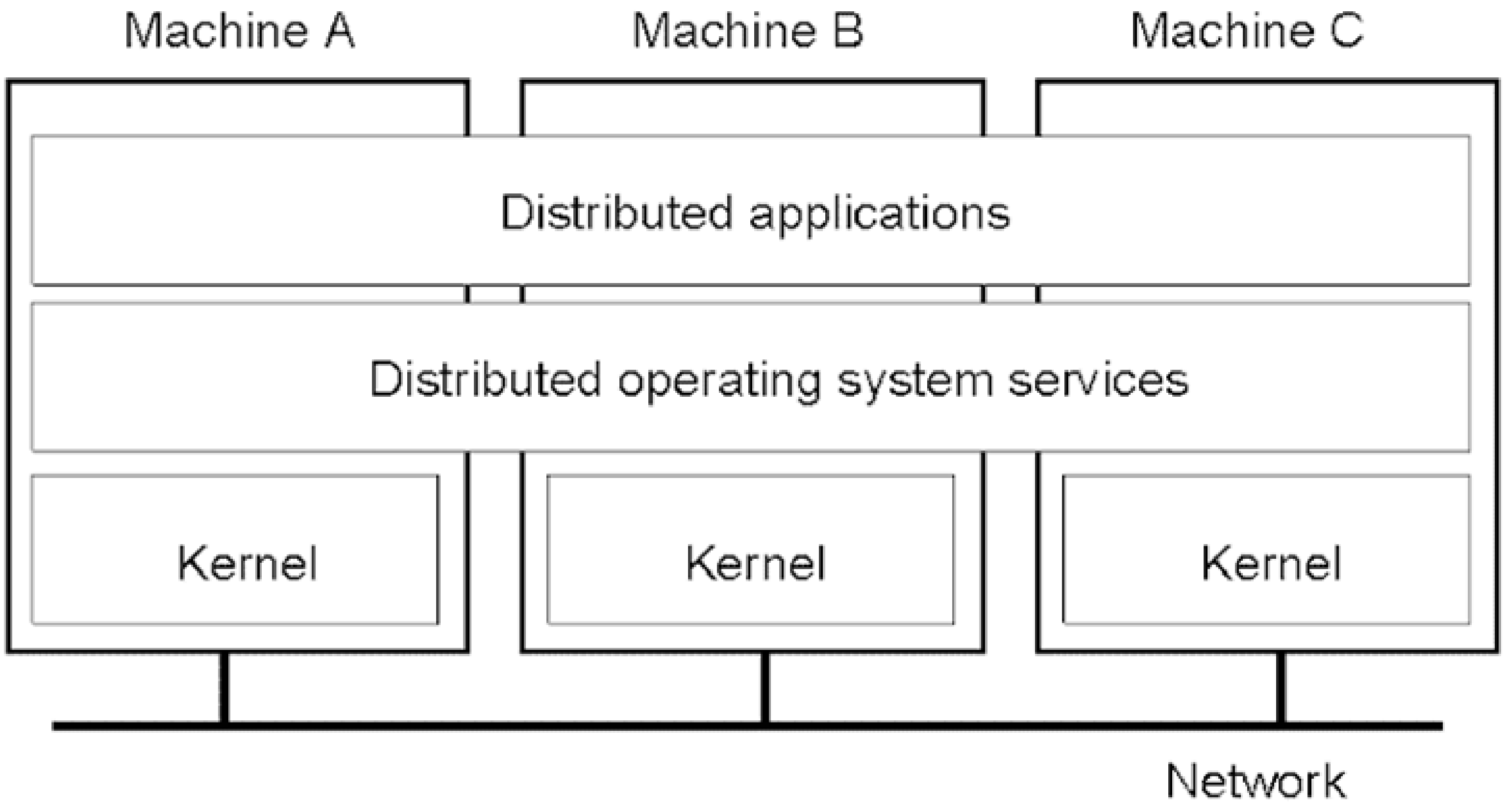
➢Users not aware of multiplicity of machines.

➢• Look to the user like a centralized OS – But operates on multiple independent CPUs .

➢• Provide transparency – Location, migration, concurrency, replication,…

➢• Present users with a virtual uniprocessor.

# Multicomputer Operating Systems (1)



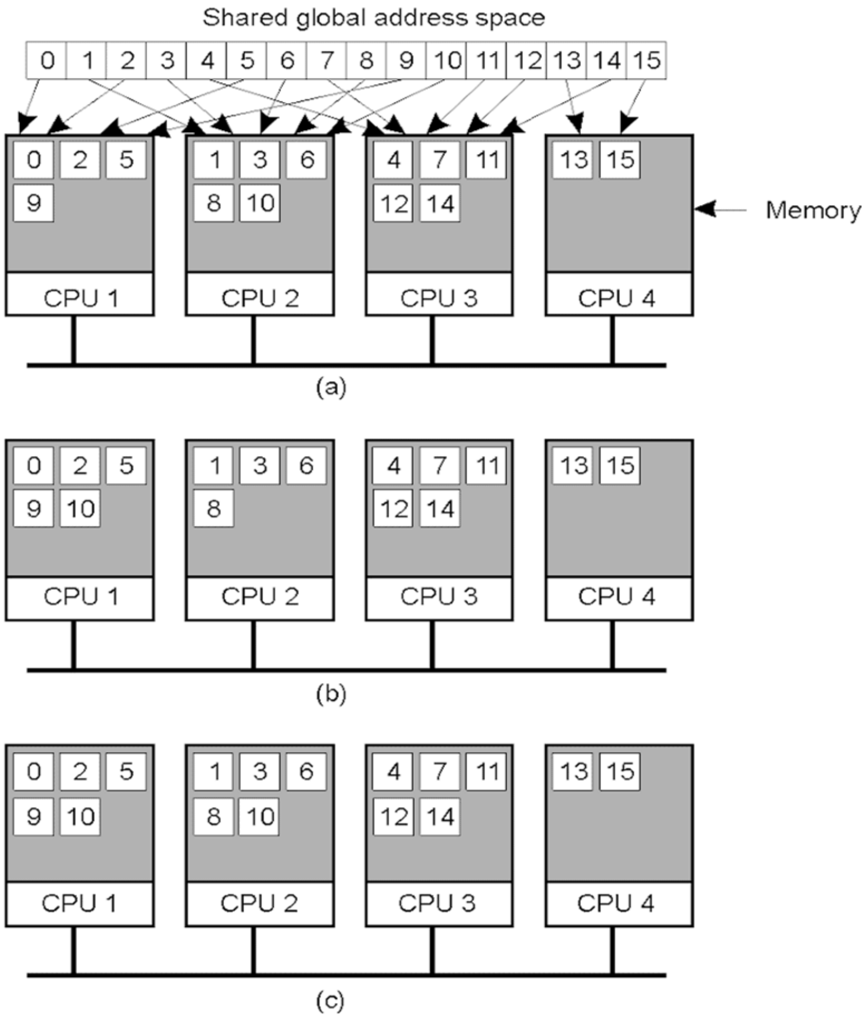
General structure of a multicomputer operating system

# Distributed Shared Memory Systems (1)

a) Pages of address space distributed among four machines

a) Situation after CPU 1 references page 10

a) Situation if page 10 is read only and replication is used

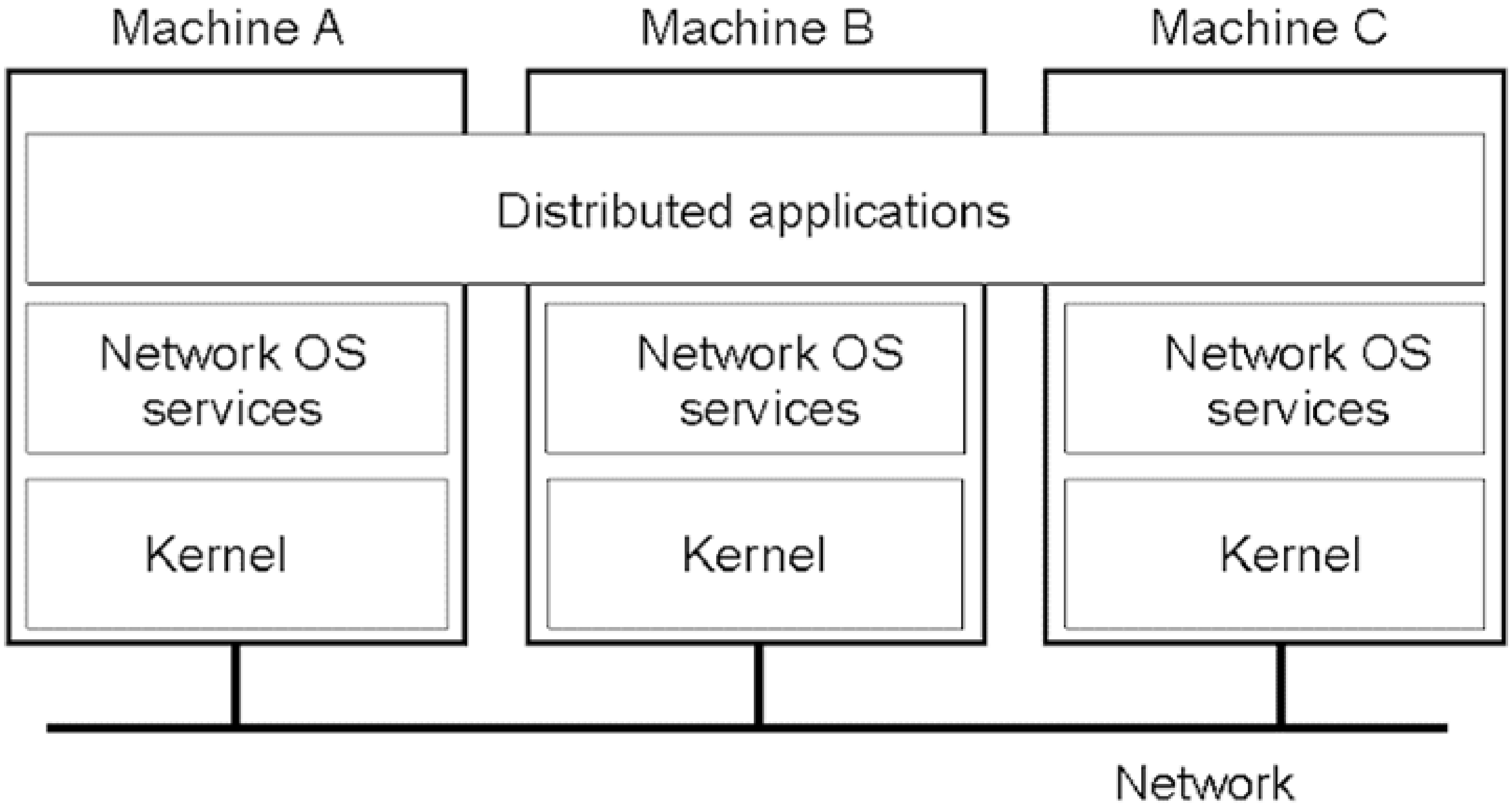


(a)

(b)

(c)

# Network Operating System (1)



General structure of a network operating system.

# Network-Operating Systems

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

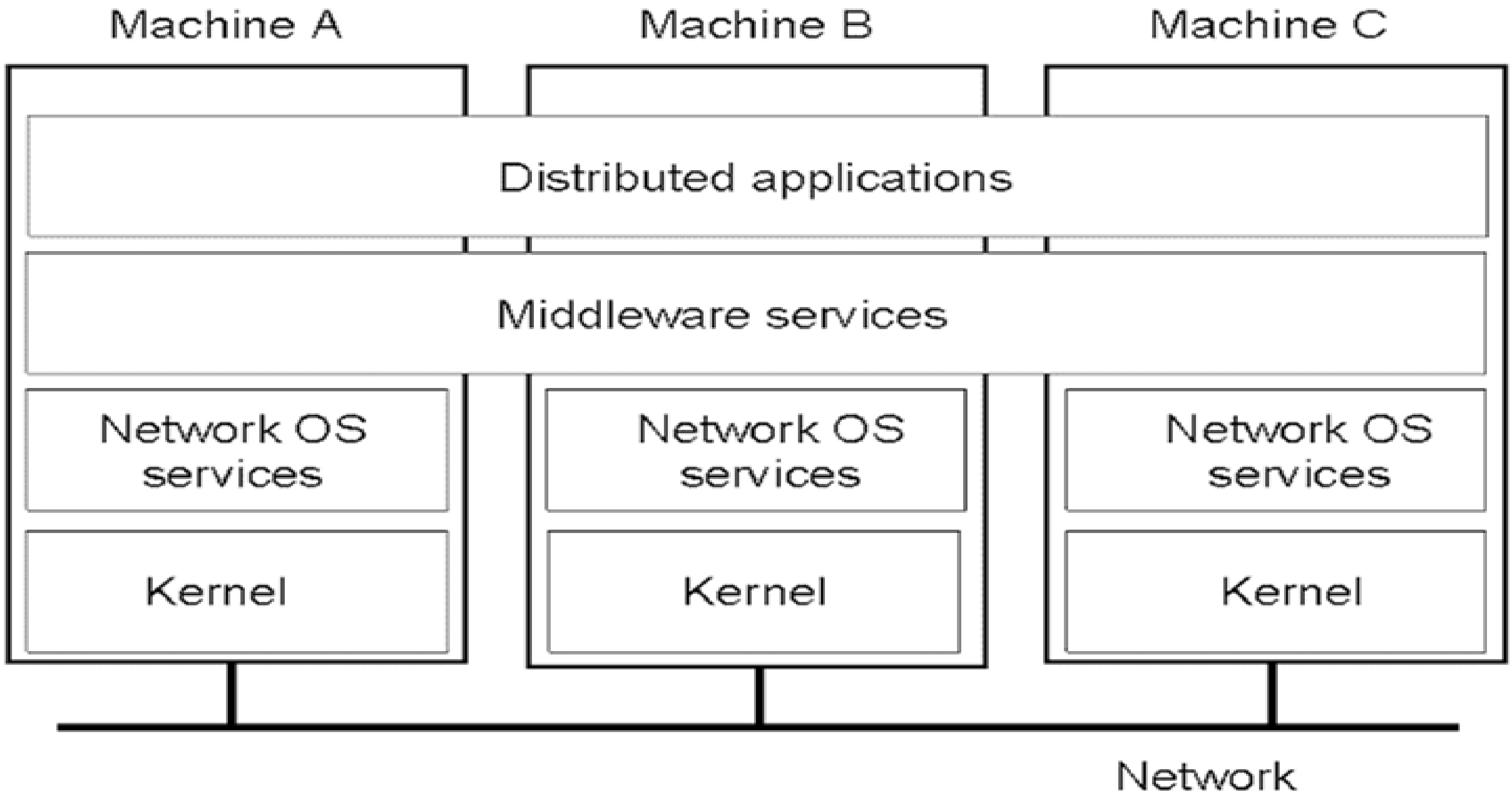
# Network Operating Systems (cont.)

- Employ a client-server model

Pros: – Minimal OS kernel
– Easy to add new machines to the system

Cons: – Lack transparency

# Positioning Middleware



General structure of a distributed system as middleware.

# Middleware Based Systems

- Middleware: A software layer placed between the application/user layer and the operating system layer. Allows users and applications to “ignore” the differences in lower layers (OS, hardware, etc.)
- Introduce transparency

# Comparison between Systems

| Item | Distributed OS | | Network OS | Middleware-based OS |
|---|---|---|---|---|
| | 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 |

A comparison between multiprocessor operating systems, multicomputer operating systems, network operating systems, and middleware based distributed systems.

# Client-server model

- Most important and most widely distributed system architecture.

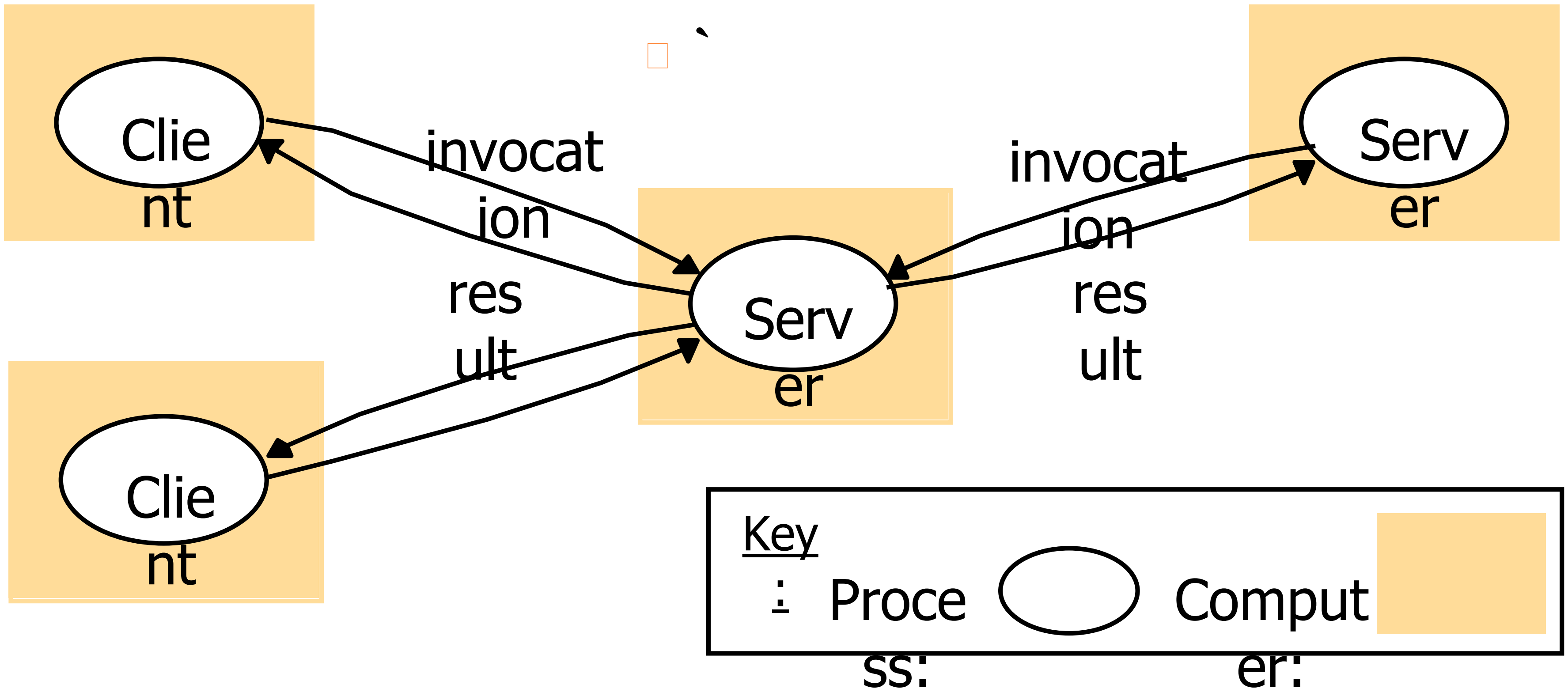
- Client and server roles are assigned and changeable.
  - Servers may in turn be clients of other servers.

- Services may be implemented as several interacting processes in different host computers to provide a service to client processes:

  - Servers partition the set of objects on which the service is based and distribute them among themselves (e.g. Web data and web servers)

  - Servers maintain replicated copies of the service objects on several hosts for reliability and performance (e.g. AltaVista)

# SYSTEM ARCHITECTURES

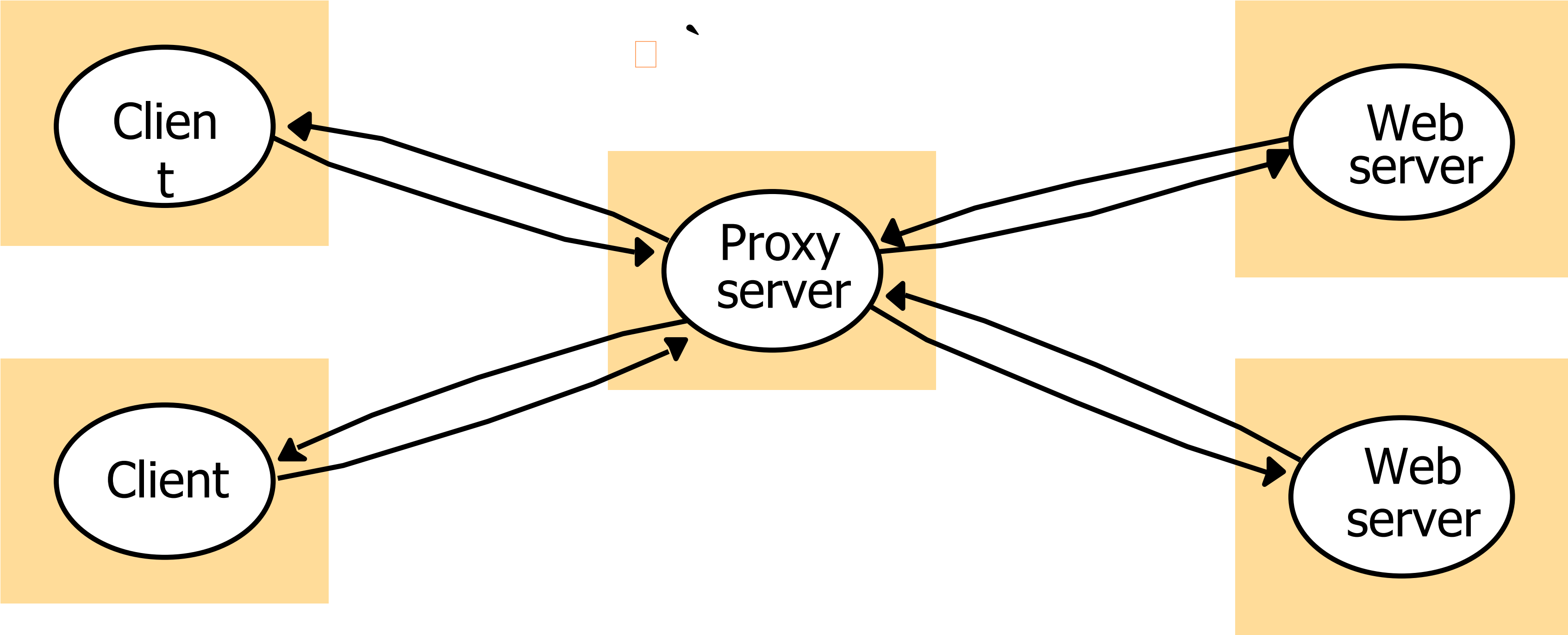


Clients invoke individual servers

# Caches and proxy servers:

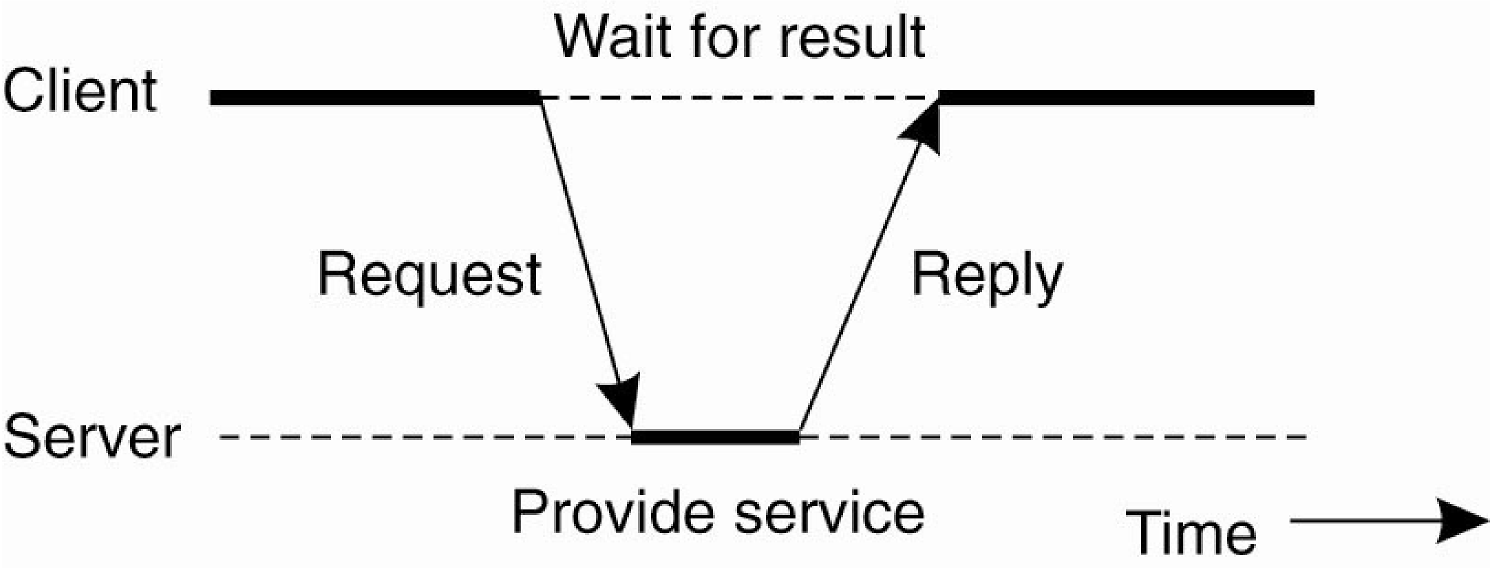
- **Cache:**
  - A store of recently used data objects that is closer to the client process than those remote objects.
  - When an object is needed by a client process the caching service checks the cache and supplies the object from there in case of an up-to-date copy is available.
- **Proxy server:**
  - Provides a shared cache of web resources for client machines at a site or across several sites.
  - Increase availability and performance of a service by reducing load on the WAN and web servers.
  - May be used to access remote web servers through a firewall.

# SYSTEM ARCHITECTURES



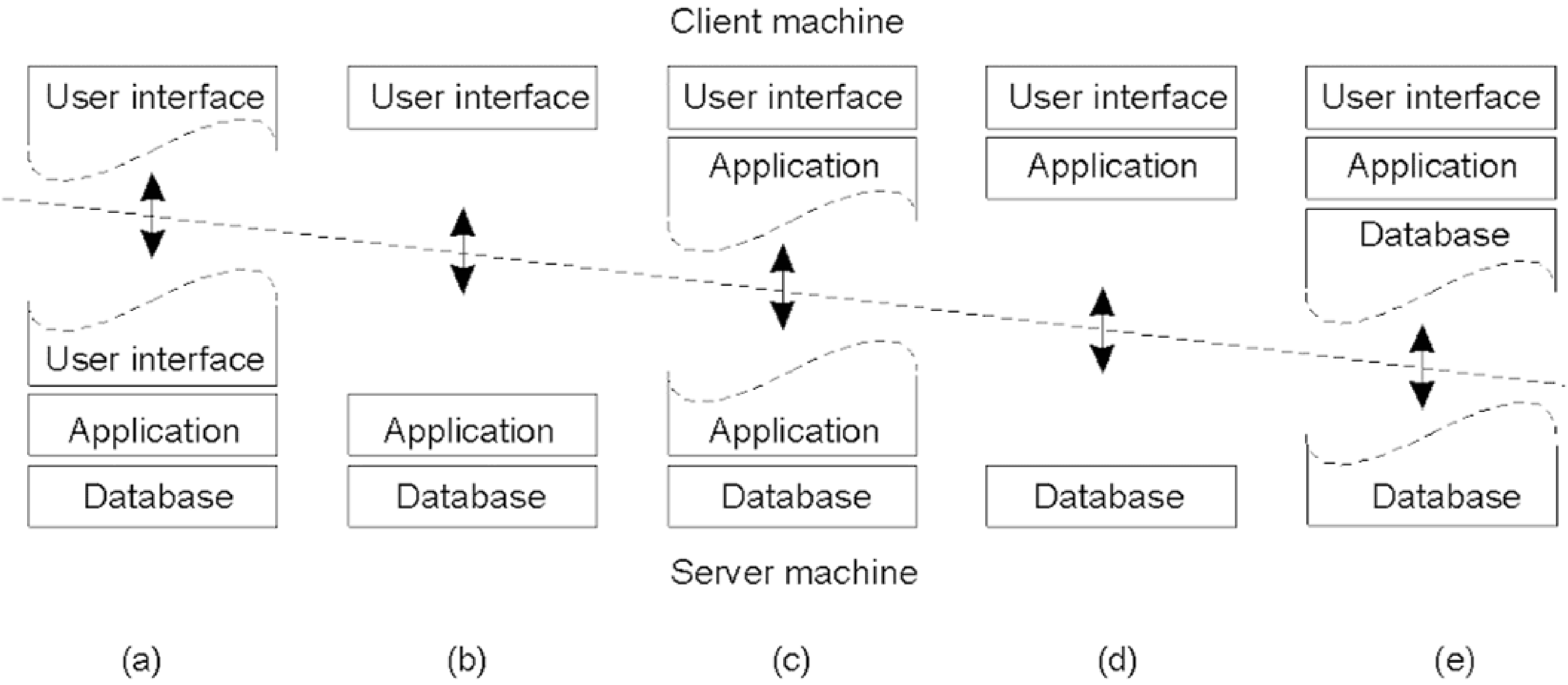
Web proxy server

# C/S Architectures



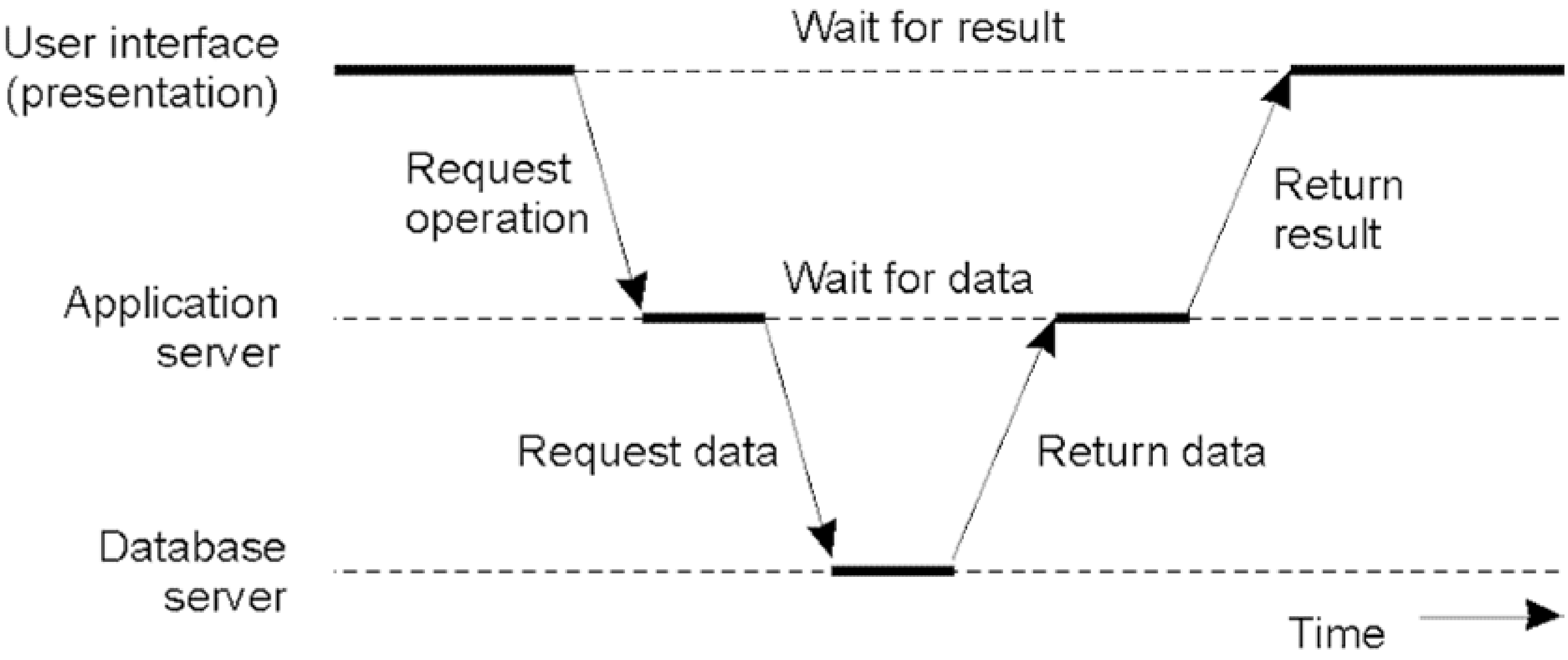
General interaction between a client and a server.

# Multitiered Architectures (1)



Alternative client-server organizations (a) – (e).

# Multitiered Architectures (2)



An example of a server acting as a client.

# Model of middleware

- RMI
- RPC
- MOM
- CORBA
- DCOM
- SOA