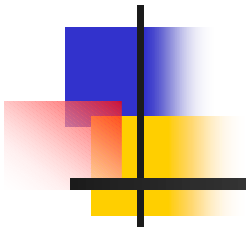


Distributed Systems

REK's adaptation of Prof. Claypool's adaptation of

Tanenbaum's

Distributed Systems

Chapter 1

The Rise of Distributed Systems

- *Computer hardware prices are falling and power increasing.*
- *Network connectivity is increasing.*
  - *Everyone is connected with fat pipes.*
- *It is easy to connect hardware together.*
- *Definition: a distributed system is*
  - *A collection of independent computers that appears to its users as a single coherent system.*

Forms of Transparency in a Distributed System

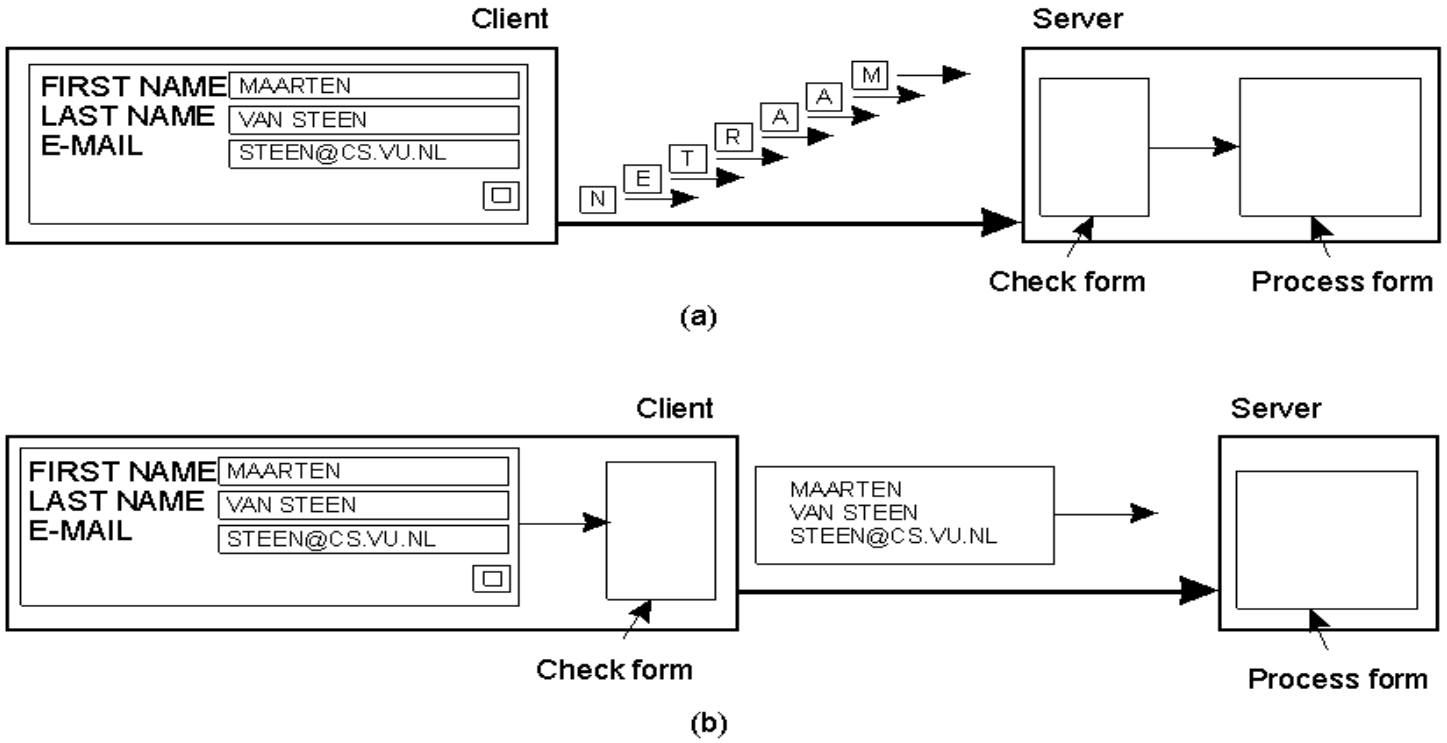
Transparency	Description
Access	Hide differences in data representation and how a resource is accessed
Location	Hide where a resource is located
Migration	Hide that a resource may move to another location
Relocation	Hide that a resource may be moved to another location while in use
Replication	Hide that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

Scalability Problems

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

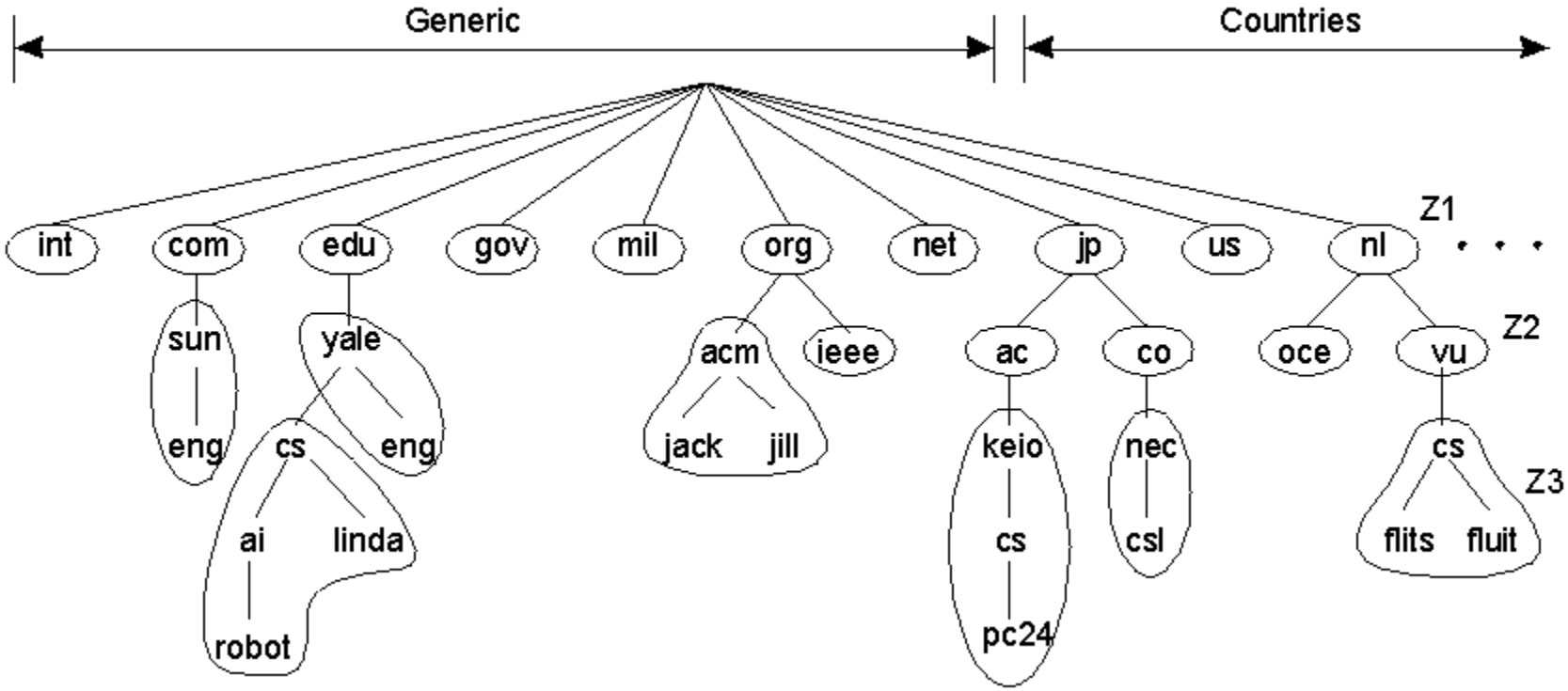
- As distributed systems grow, centralized solutions are limited.

Hiding Communication Latency



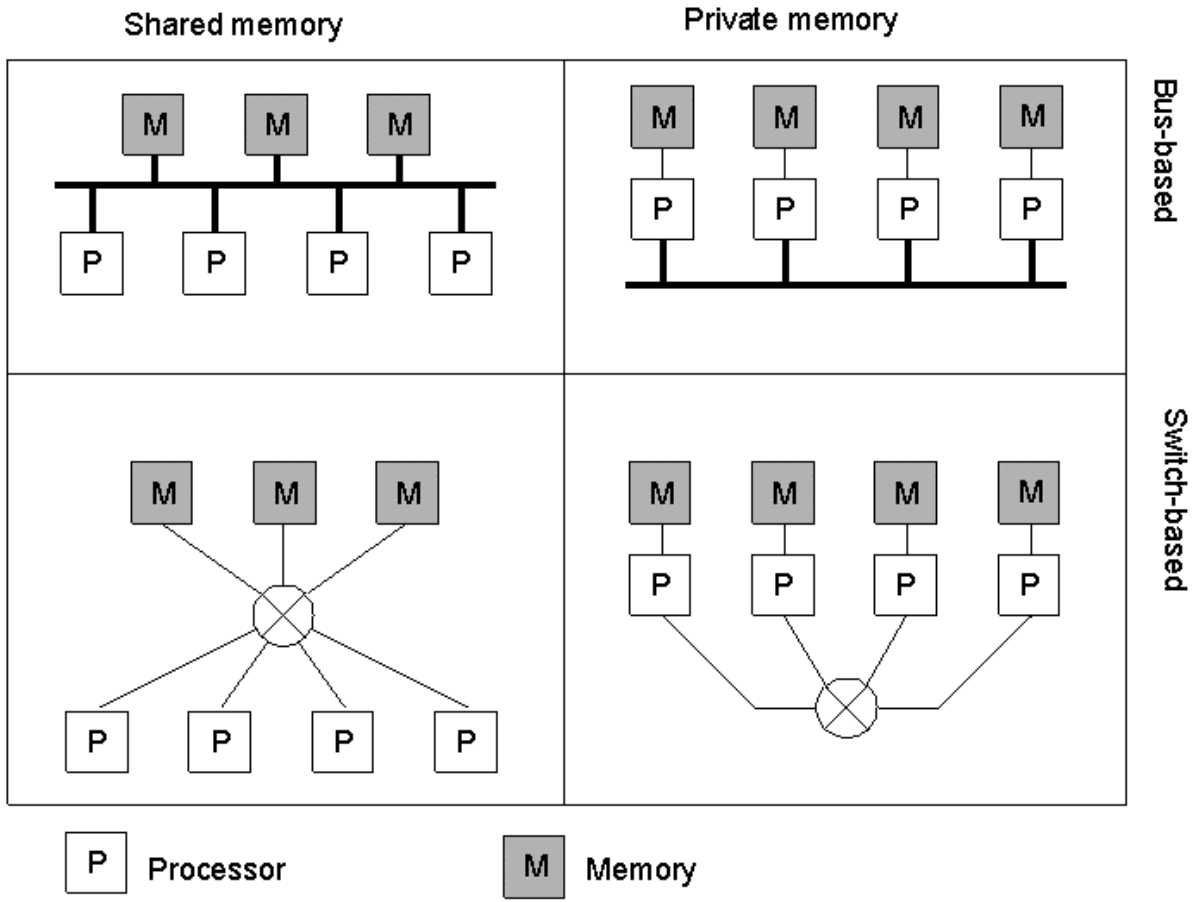
- This is especially important for interactive applications
- If possible, system can do *asynchronous communication.*
- The system can hide latencies.

Dividing the DNS name space into zones



Hardware Concepts

Basic organizations and memories in distributed computer systems



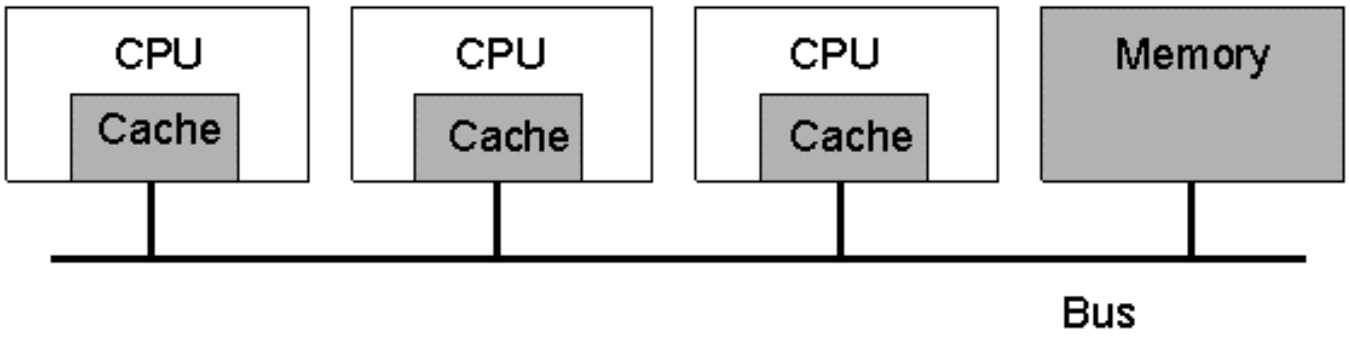
Hardware Considerations

- *General Classification:*
  - *Multiprocessor – a single address space among the processors*
  - *Multicomputer – each machine has its own private memory.*
- *OS can be developed for either type of environment.*

Multiprocessor Organizations

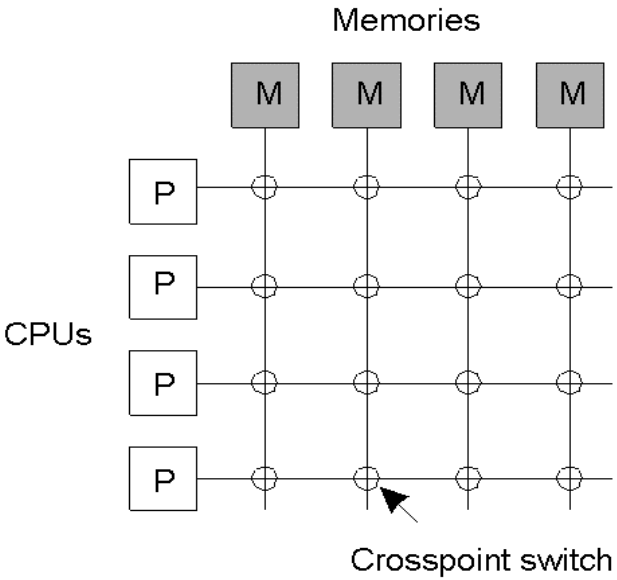
- *Uniform Memory Access [UMA]*
  - *Caching is vital for reasonable performance (e.g., caches on a shared memory multiprocessor).*
  - *Want to maintain cache coherency*
    - ***Write-through cache*** *:: any changes to cache are written through to memory.*

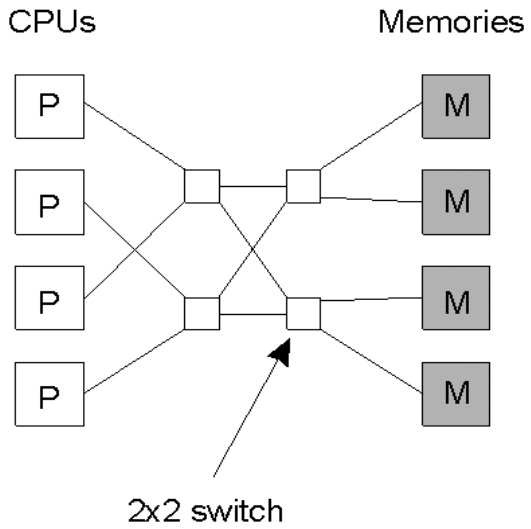
Multiprocessors



A bus-based multiprocessor.

Multiprocessors





A crossbar switch

An omega switching network

Multiprocessor Organizations

- *Non-Uniform Memory Access [NUMA]*
  - *A hierarchy where CPUs have their own memory (not the same as a cache).*
  - *Access costs to memory is non-uniform.*

Replication

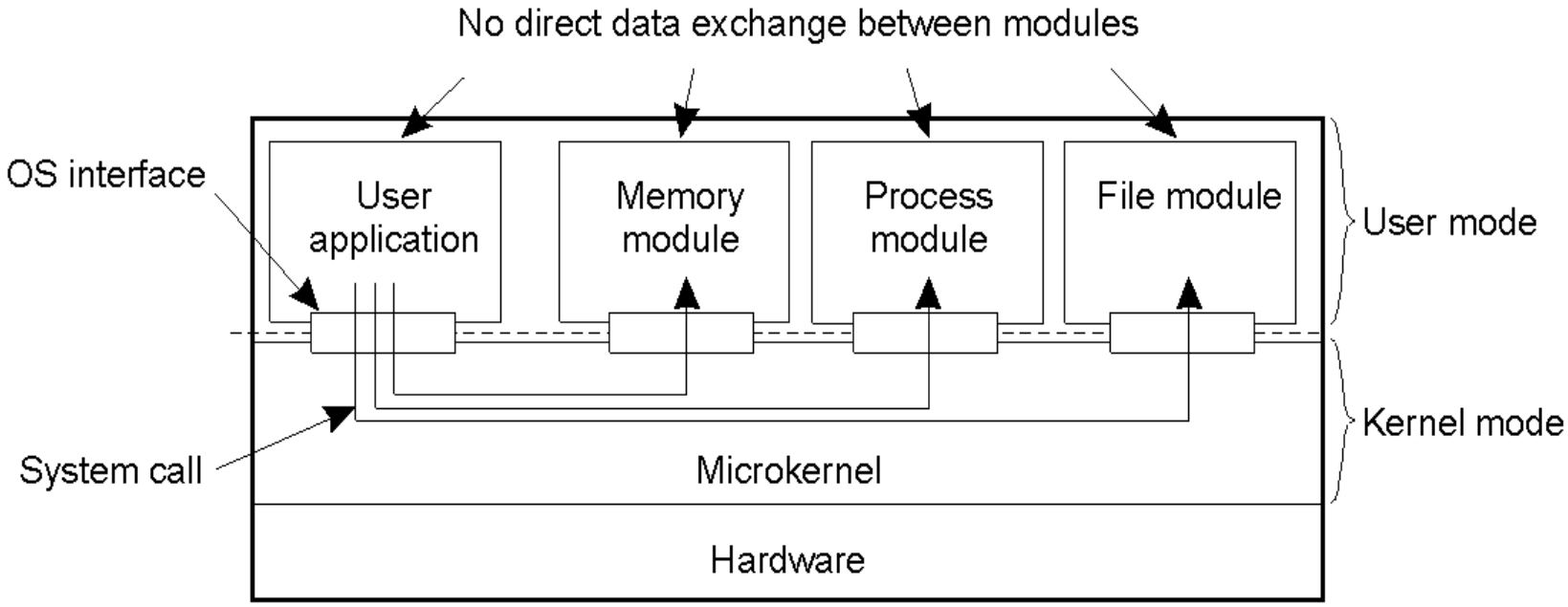
- *Make a copy of information to increase availability and decrease centralized load.*
  - *Example: P2P networks (Gnutella +) distribute copies uniformly or in proportion to use.*
  - *Example: CDNs (Akamai)*
  - *Example: Caching is a replication decision made by client.*
- *Issue: Consistency of replicated information*
  - *Example: Web Browser cache*

Software Concepts

- *DOS (Distributed Operating Systems)*
- *NOS (Network Operating Systems)*
- *Middleware*

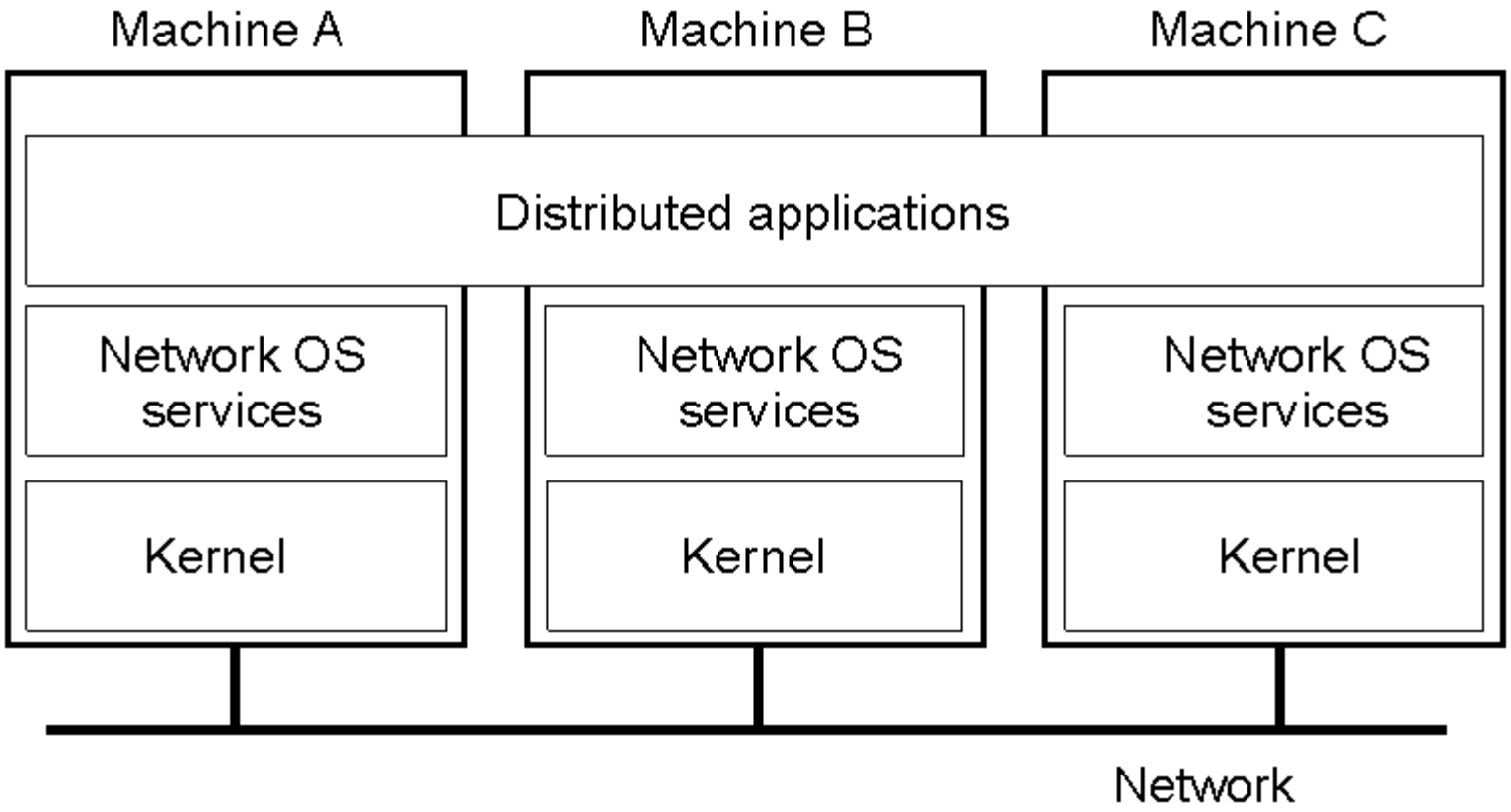
System	Description	Main Goal
DOS	Tightly-coupled operating system for multi-processors and homogeneous multicomputers	Hide and manage hardware resources
NOS	Loosely-coupled operating system for heterogeneous multicomputers (LAN and WAN)	Offer local services to remote clients
Middleware	Additional layer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

Uniprocessor Operating Systems



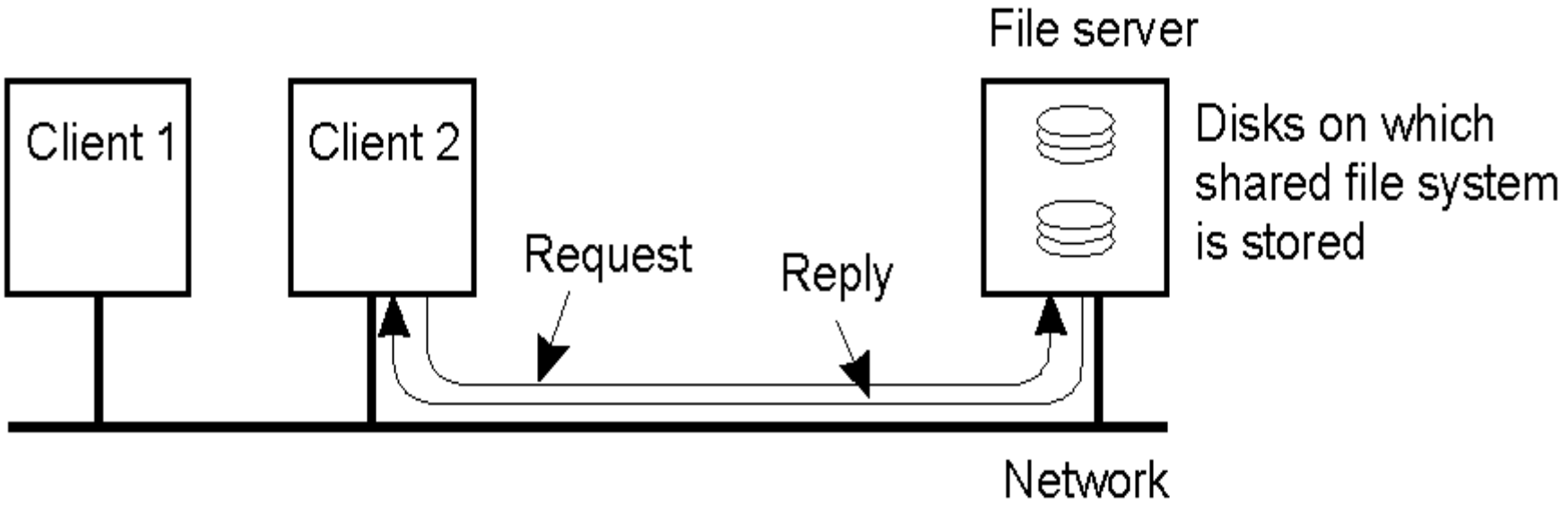
- *Separating applications from operating system code through a microkernel*
  - *Can extend to multiple computers*

Network Operating System



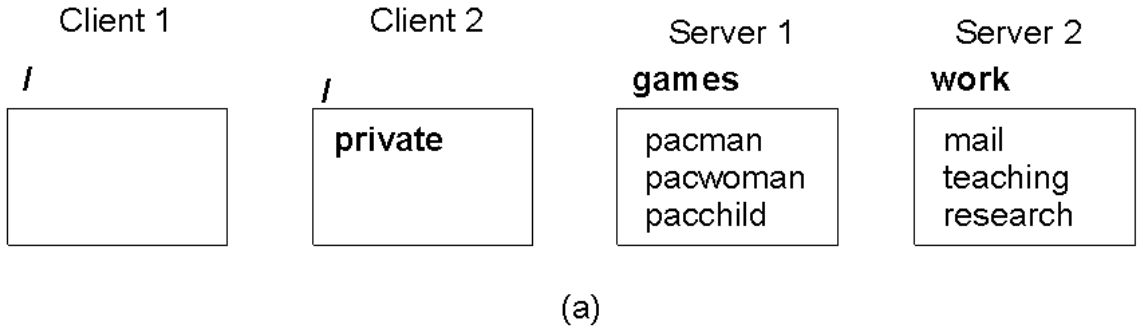
- *OSes can be different (Windows or Linux)*
- *Typical services: rlogin, rcp*
  - *Fairly primitive way to share files*

Network Operating System

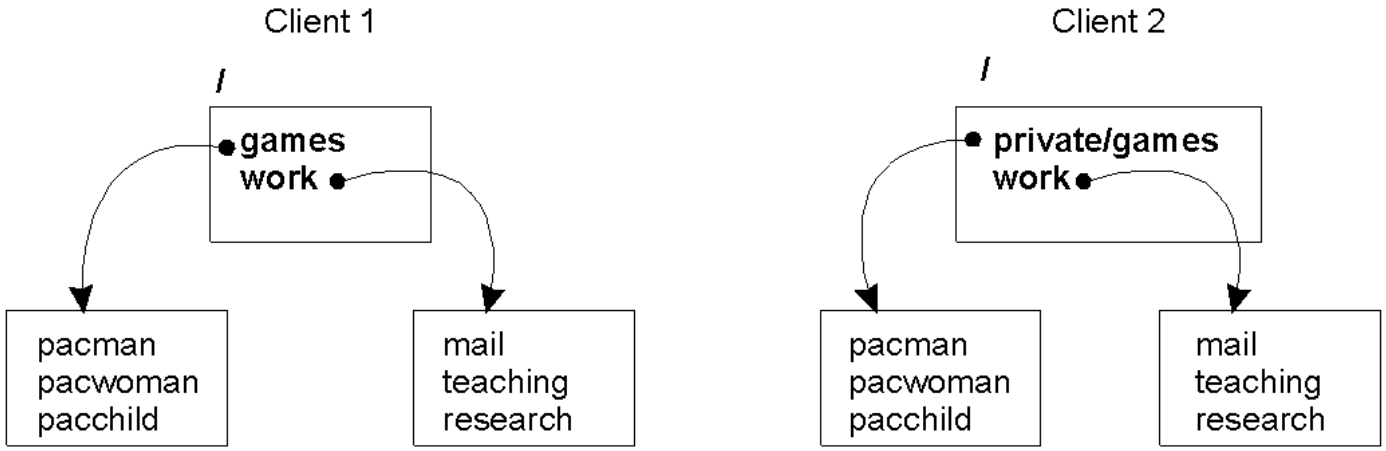


- *Can have one computer provide files transparently for others (NFS)*
  - *(try a "*`df`*" on the WPI hosts to see. Similar to a "mount network drive" in Windows)*

Network Operating System



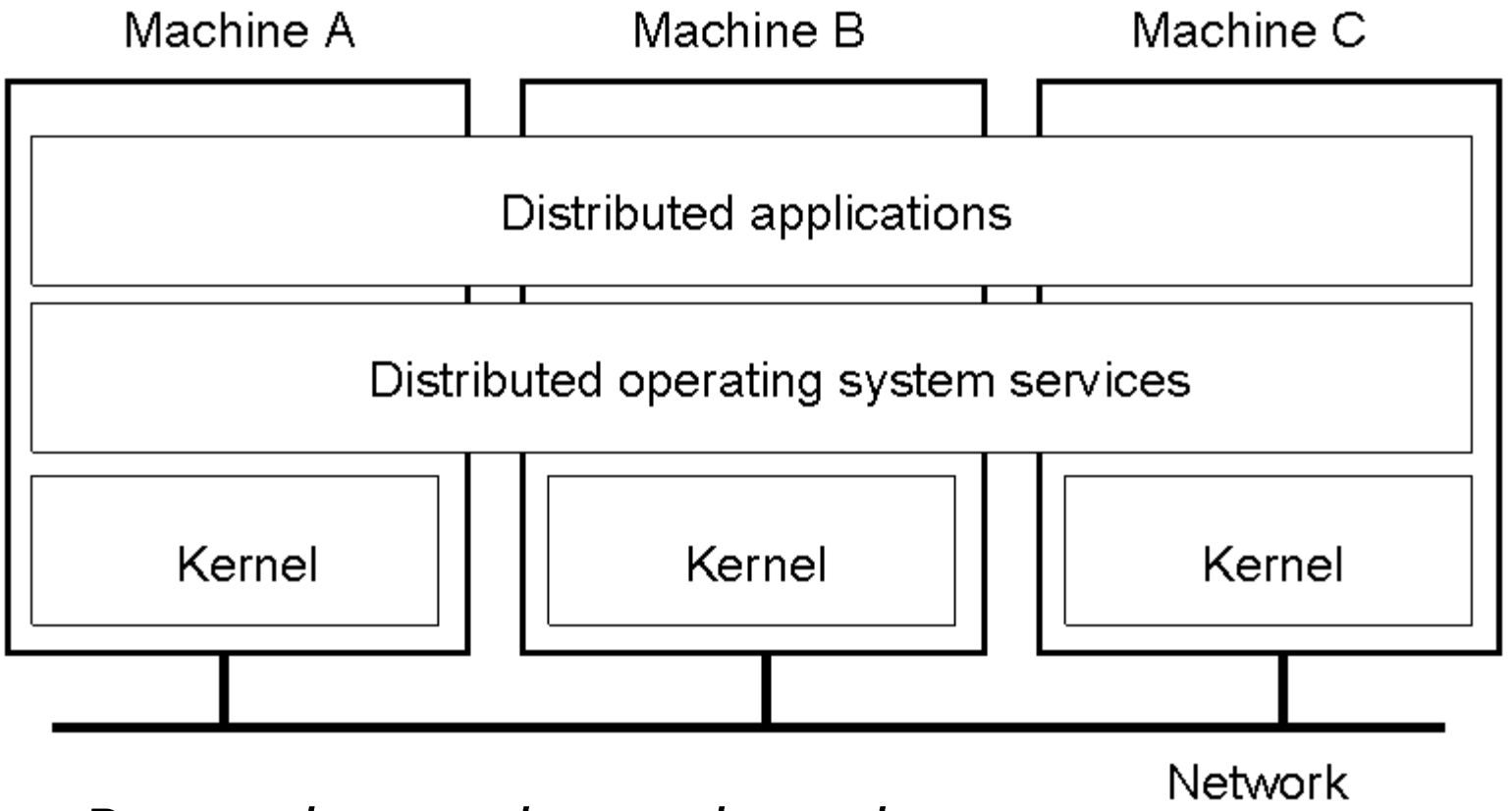
(a)



(b) (c)

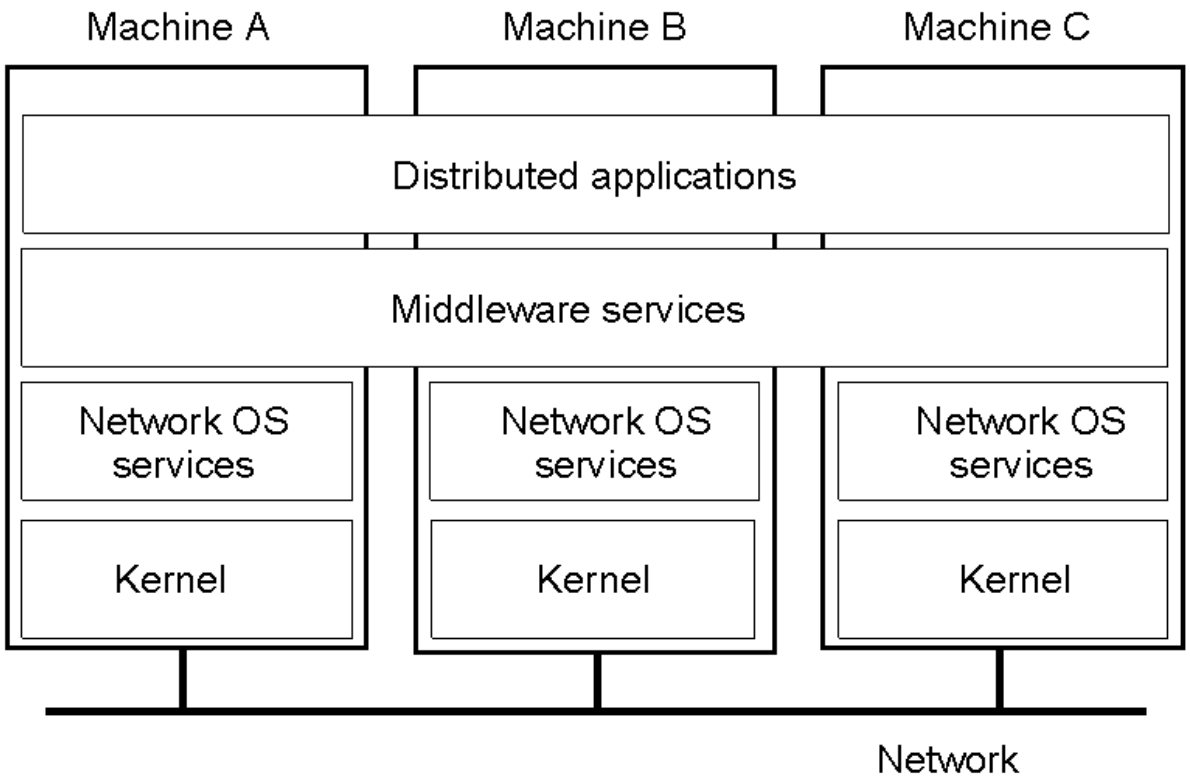
- *Different clients may mount the servers in different places*
- *Inconsistencies in view make NOS's harder, in general for users than DOS's.*
  - *But easier to scale by adding computers*

Distributed Operating Systems



- *But no longer have shared memory*
  - *Provide message passing*
  - *Can try to provide distributed shared memory*
    - *But tough to get acceptable performance*

Distributed System as Middleware



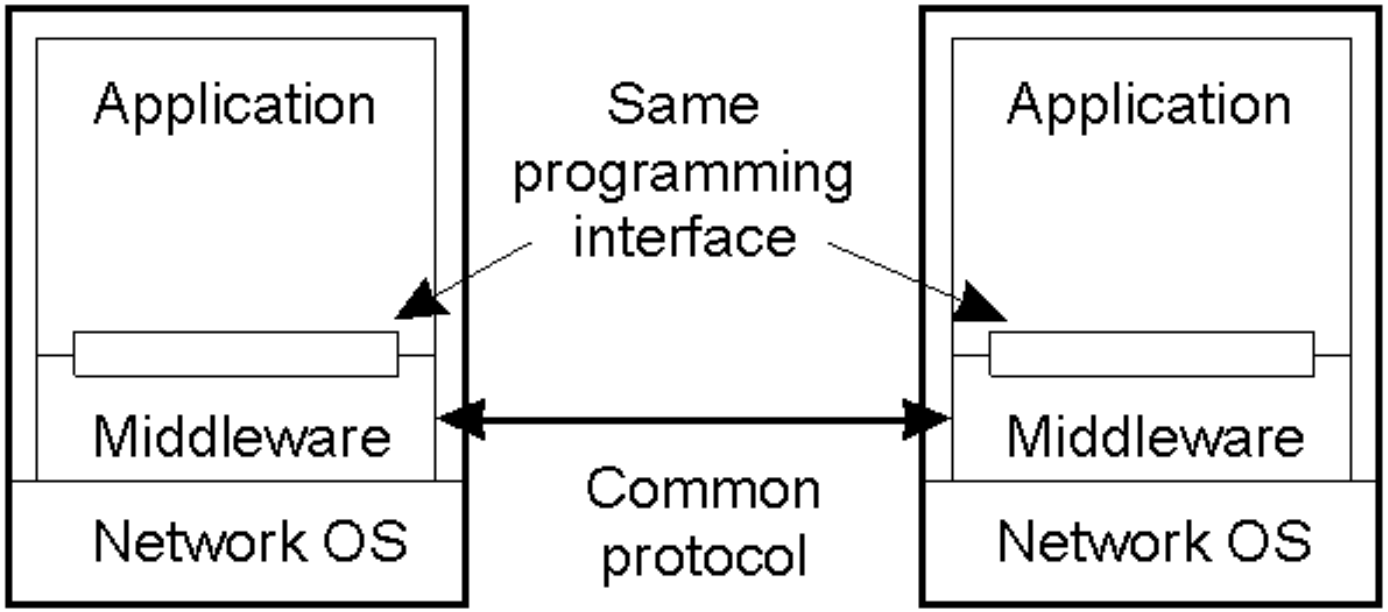
Positioning Middleware

- *Network OS's are not transparent.*
- *Distributed OS's are not independent of computers.*
- *Middleware can help.*

Middleware Models

- View everything as a file - Plan 9.
- Less strict – distributed file systems.
- Make all procedure calls appear to be local – Remote Procedure Calls (RPC).
- Distributed objects (oo model).
- The Web – distributed documents.

Middleware and Openness

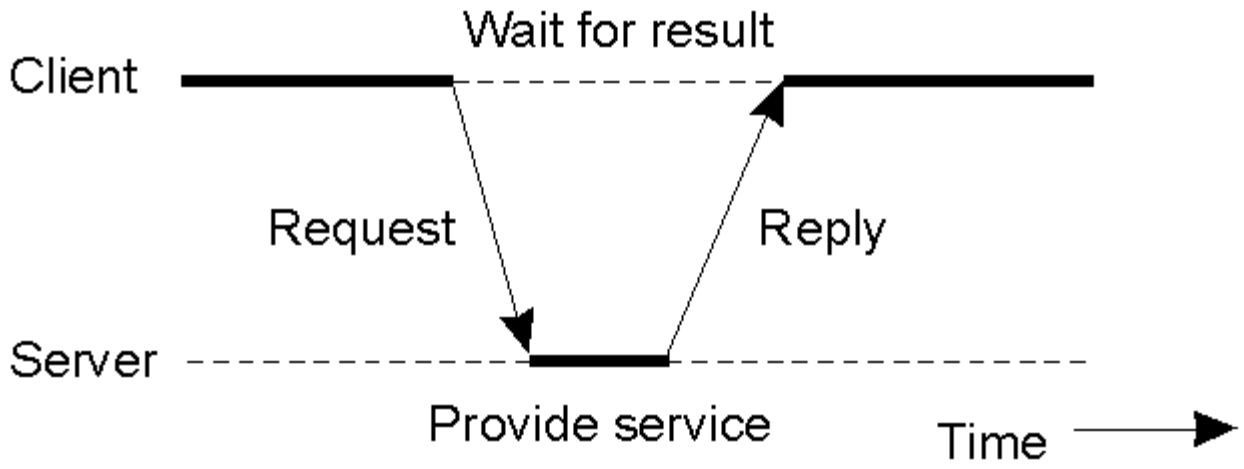


- *In an open middleware-based distributed system, the protocols used by each middleware layer should be the same, as well as the interfaces they offer to applications.*
  - *If different, there will be compatibility issues*
  - *If incomplete, then users will build their own or use lower-layer services (frowned upon)*

Comparison between Systems

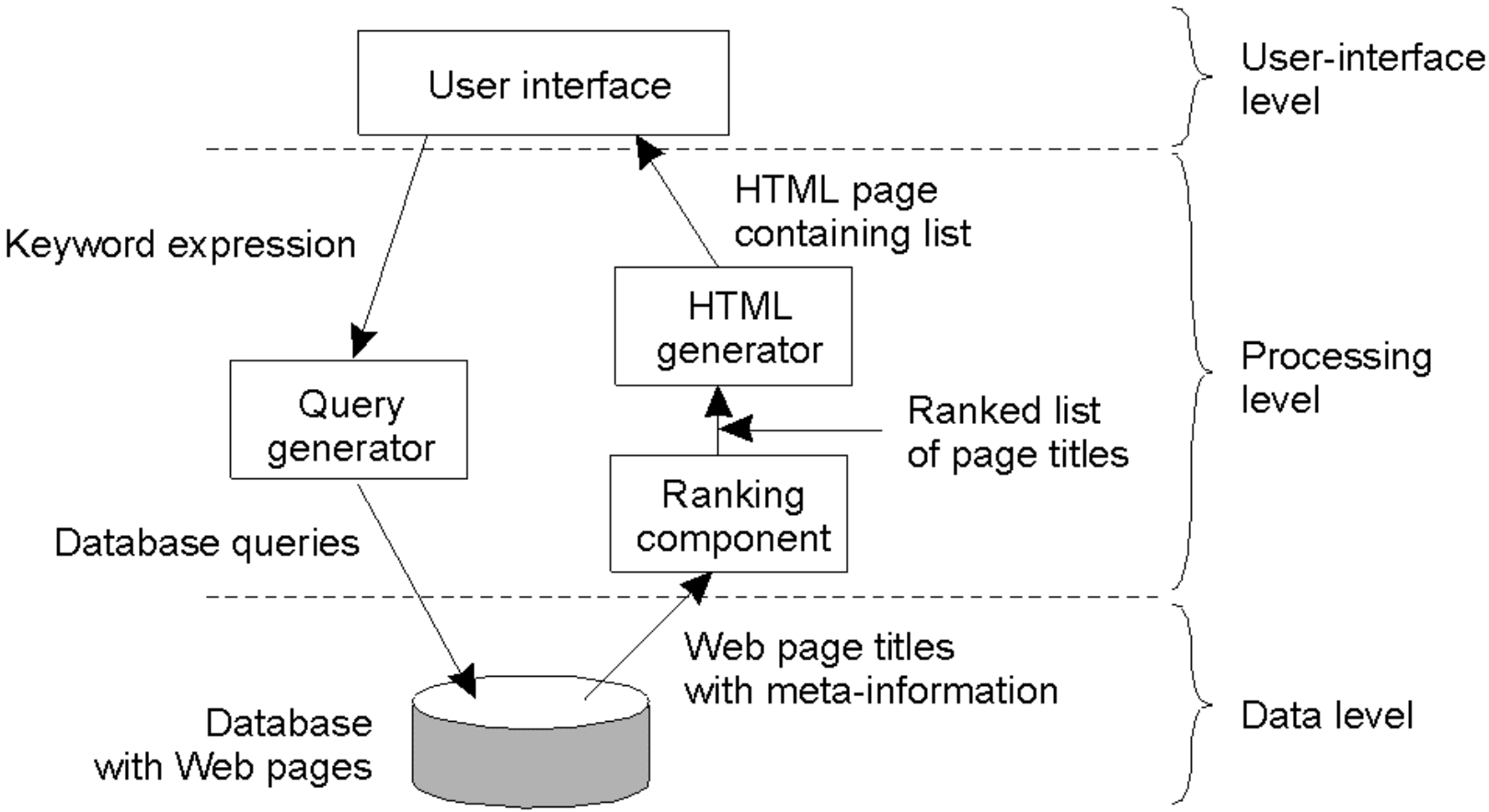
Item	Distributed OS		Network OS	Middleware -based OS
	Multiproc.	Multicomp.		
Degree of transparency	Very High	High	Low	High
Same OS on all nodes	Yes	Yes	No	No
Number of copies of OS	1	N	N	N
Basis for communication	Shared memory	Messages	Files	Model specific
Resource management	Global, central	Global,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

Client-Server Model

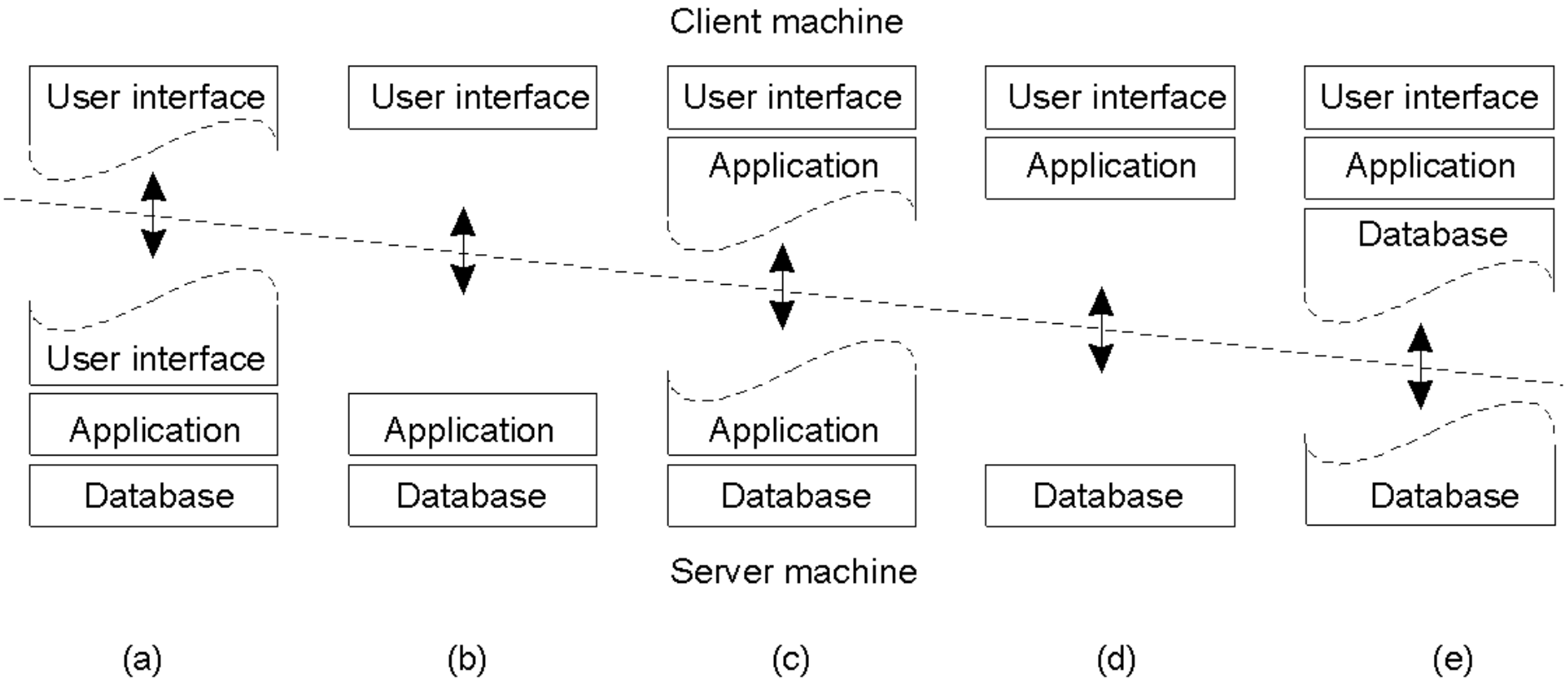


- Use TCP/IP for reliable network connection.
  - This implies the client must establish a connection before sending the first request.

Internet Search Engine

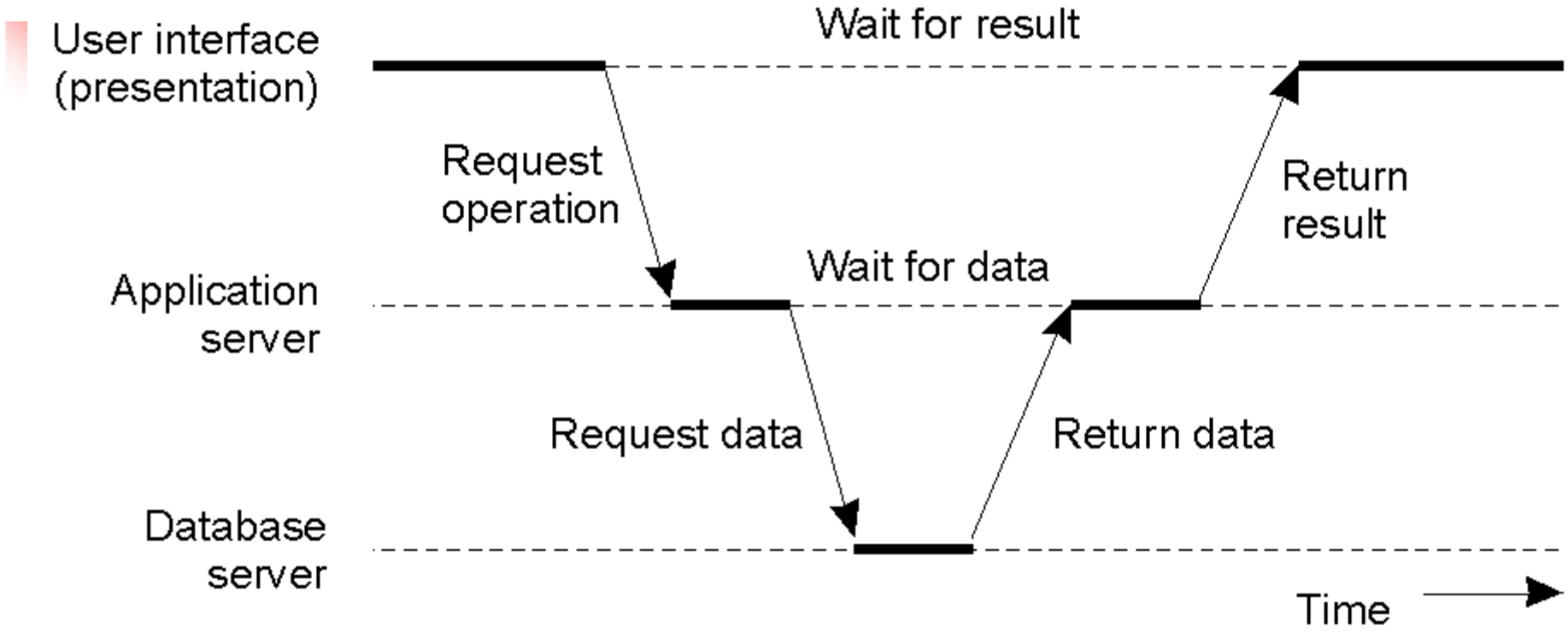


Multitiered Architectures



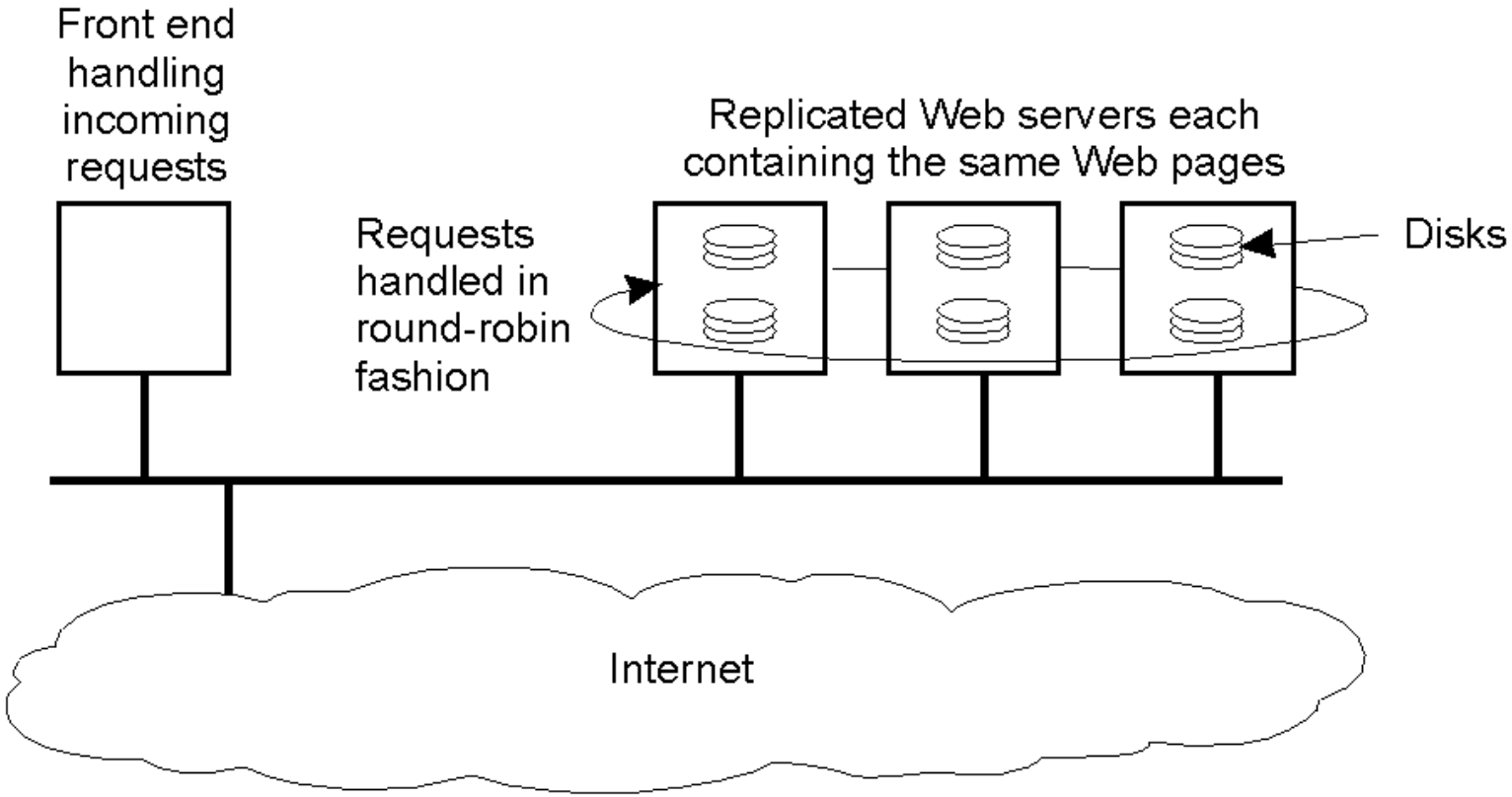
- *Thin client (a) to Fat client (e)*
  - *(d) and (e) popular for NOS environments*

Multitiered Architectures: 3 tiers



- *Server may act as a client*
  - *Example would be transaction monitor across multiple databases*

Horizontal Distribution



- *Distribute servers across nodes*
  - *E.g., Web server "farm" for load balancing*
- *Distribute clients in peer-to-peer systems.*