

**Operating Systems**  
**CSE 410, Spring 2004**

**Distributed Operating Systems**

Stephen Wagner

Michigan State University

# Distributed Systems

- Operating Systems were designed with a single computer in mind
  - Purpose of operating system was to make efficient use of processor
  - Provide users with access to shared I/O devices
- Increasingly I/O devices are shared among many machines
- Trend has been to distribute the OS across many machines
- Definition of distributed OS is fuzzy

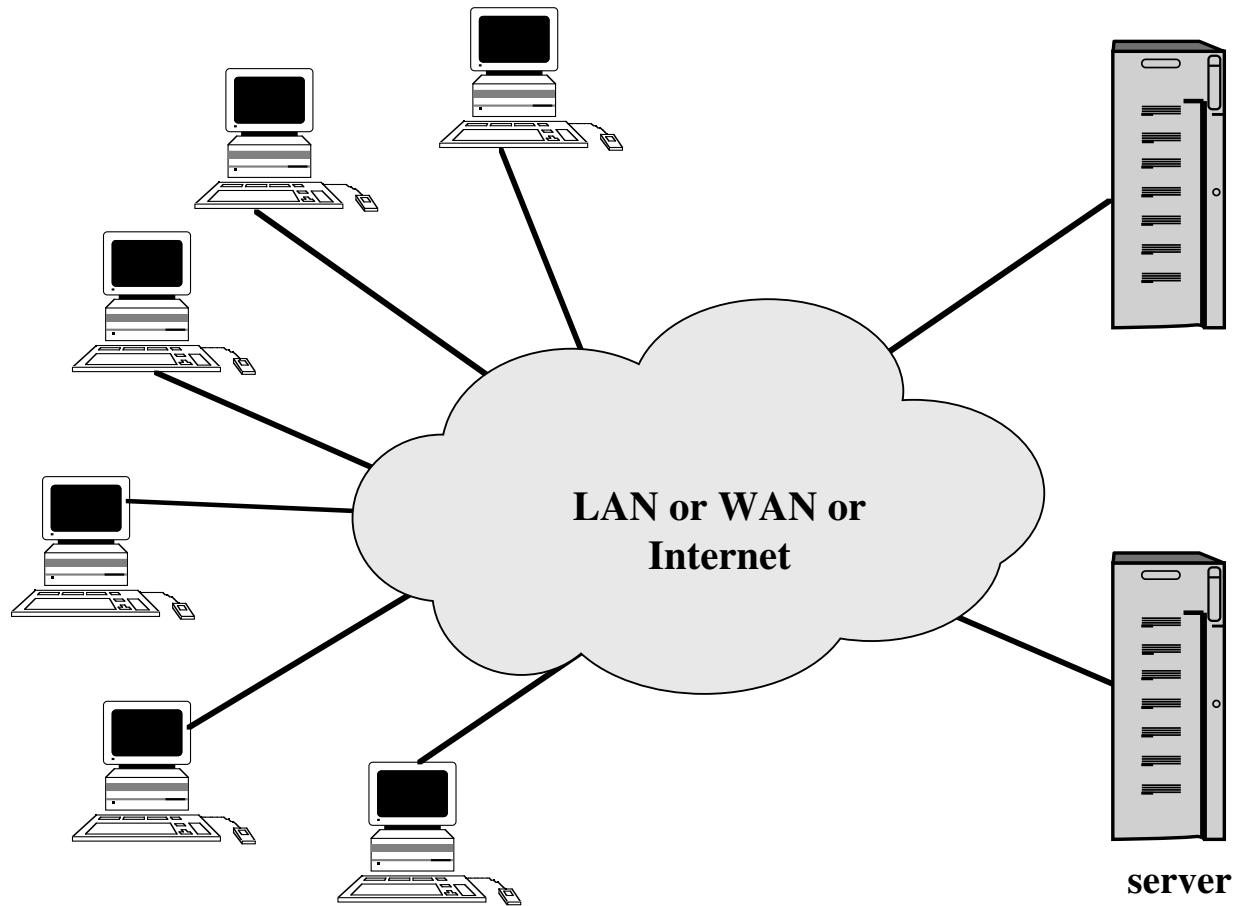
# Evolution of Distributed Systems

- Stand-alone machine
- Networked machines and FTP
- Remote Procedure Call (RPC)
- Distributed File Systems
- Process Migration

# Client/Server

- Basic model for modern distributed systems
- Server:
  - Controls a set of I/O devices or some data source
- Client:
  - Sends requests to the server for information
- Network is a critical component

# Client/Server



# Client/Server

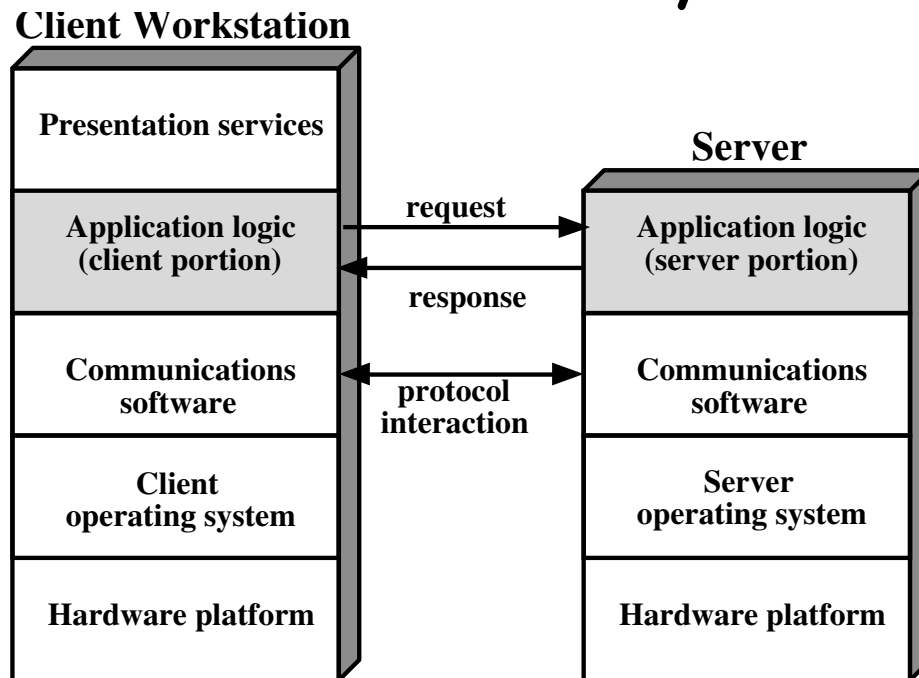


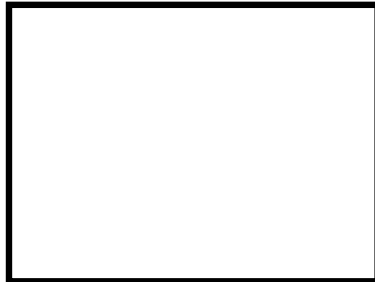
Figure 13.2 Generic Client/Server Architecture

# Classes of Client/Server Applications

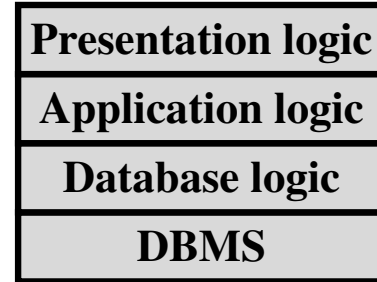
- Host based processing
  - client is essentially a dumb terminal
  - all work is done on the server, including presentation
- Server-based processing
  - client is only responsible for presentation
  - all real work is done on server
  - most common model

# Classes of Client/Server Applications

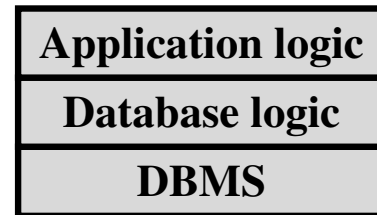
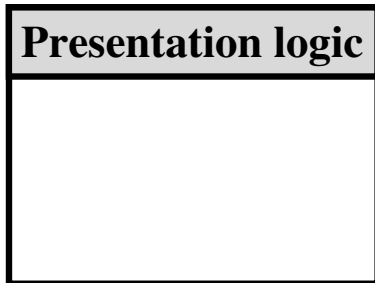
**Client**



**Server**



**(a) Host-based processing**

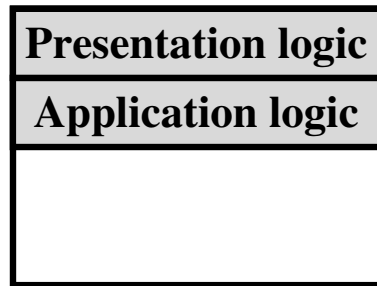


**(b) Server-based processing**

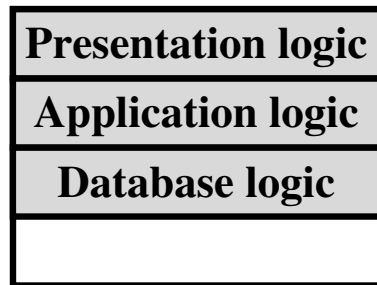
# Classes of Client/Server Applications

- Cooperative processing
  - client and server share the work
- Client-based processing
  - client does most of the work
  - server is just a passive data source

# Classes of Client/Server Applications



(c) Cooperative processing

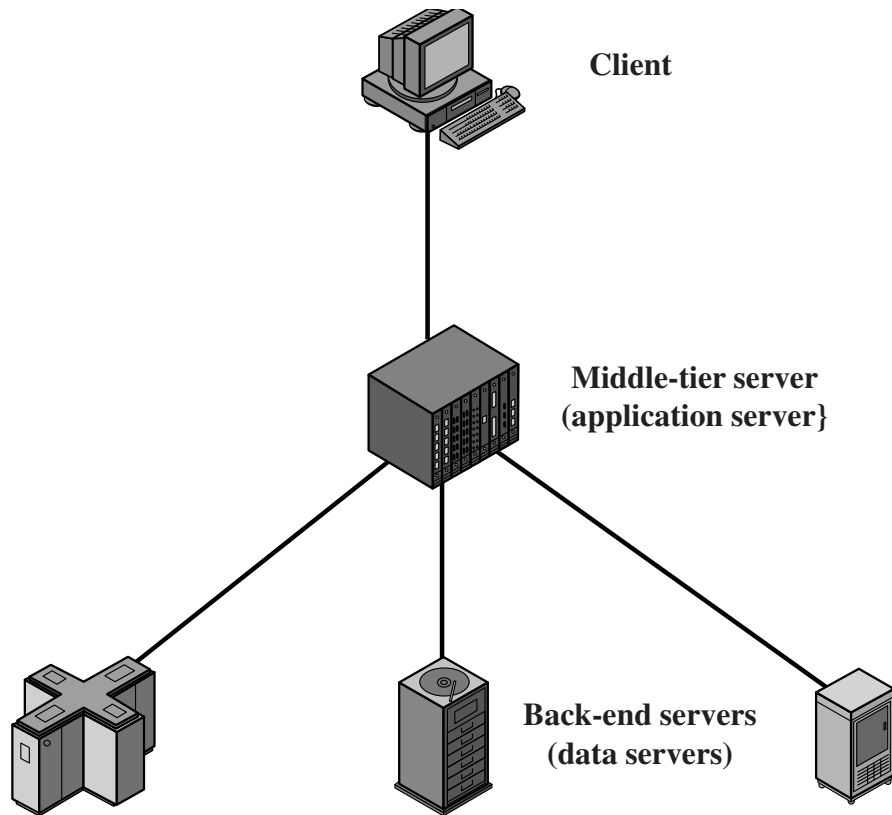


(d) Client-based processing

# Client/Server: Three Tier Architecture

- Traditional Client/Server involves two tiers
- Three Tier systems include a middle tier between clients and servers
- The middle tier provides a common gateway between clients and a number of servers
  - support old servers
  - provide security

# Client/Server: Three Tier Architecture

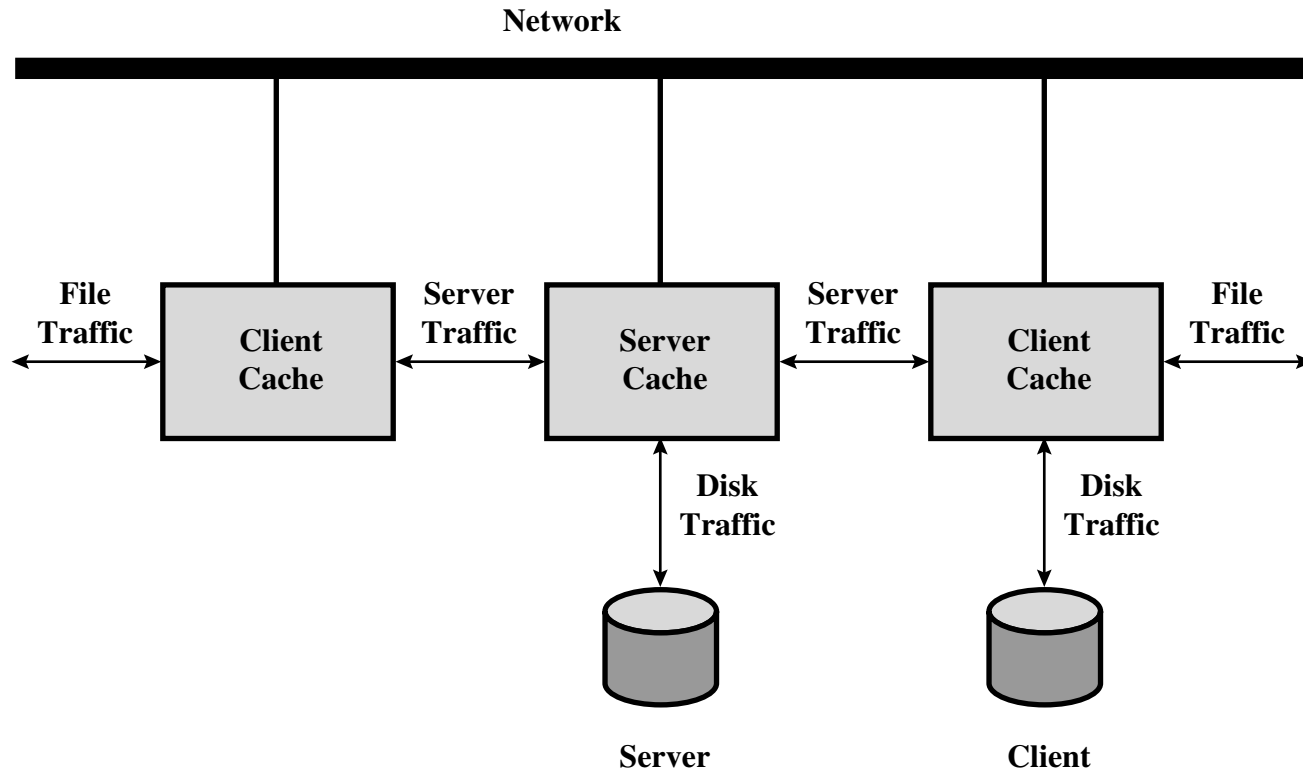


**Figure 13.6 Three-tier Client/Server Architecture**

# Client/Server and Caches

- Caches are used to improve performance
  - used at all levels
- Recently used items will be stored in the caches
- Both the clients and servers will have caches
- Reduces network traffic

# Client/Server and Caches



# Cache Consistency

- If data is only being read, there is no problem
- If data is updated, different caches may have conflicting data
- Challenging problem
  - limiting concurrent access solves the problem, but limits performance

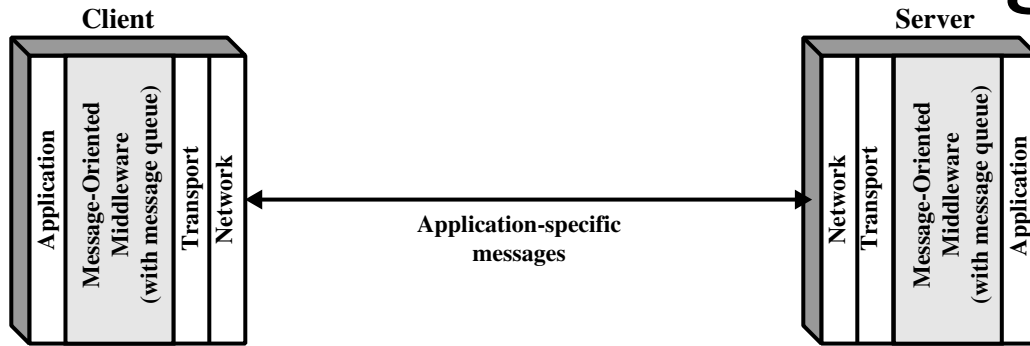
# Middleware

- There is a lot of diversity in the client/server model
- Standardized protocols are known as *middleware*
  - exist “between” client and server
  - simplify the development of client/server applications
  - many middleware packages with a wide range of complexity

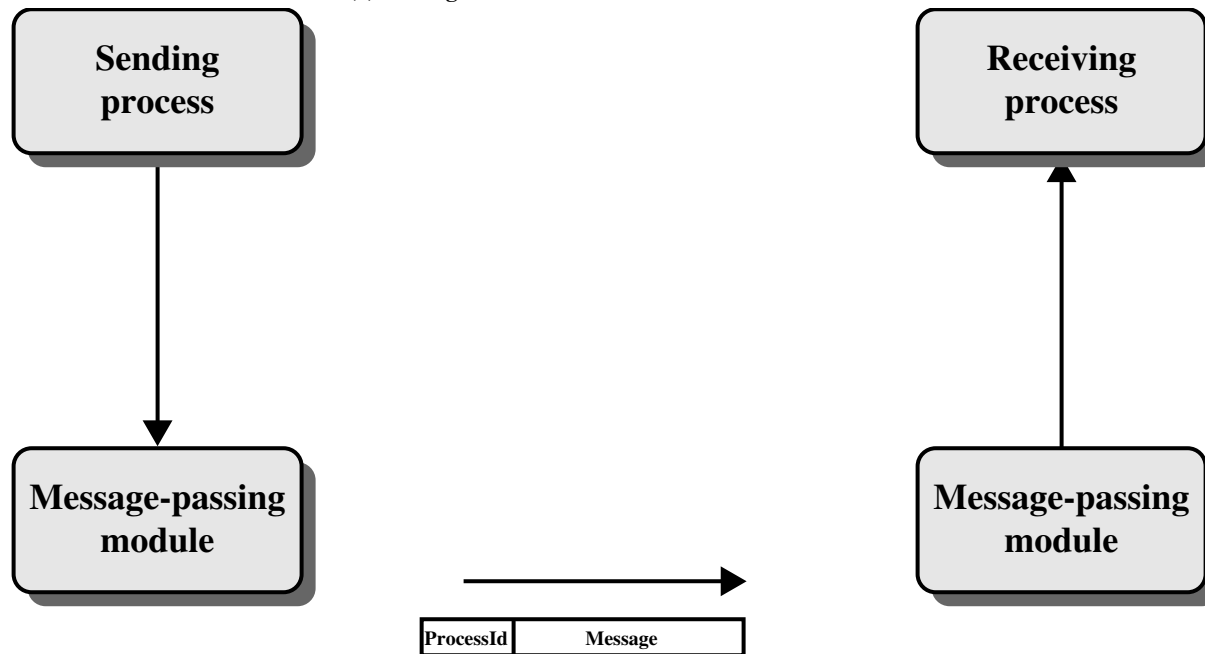
# Middleware

- Three Common Middleware Packages
  - Message Passing
  - Remote Procedure Calls
  - Object request broker

# Distributed Message Passing



(a) Message-Oriented Middleware



# Reliability

- Message Reliability
  - Is message delivery guaranteed?
  - How reliable is the network?
- The simple solution is to require applications to provide reliability
  - applications send acknowledgment messages
- A more sophisticated middle-ware package will provide guaranteed message delivery

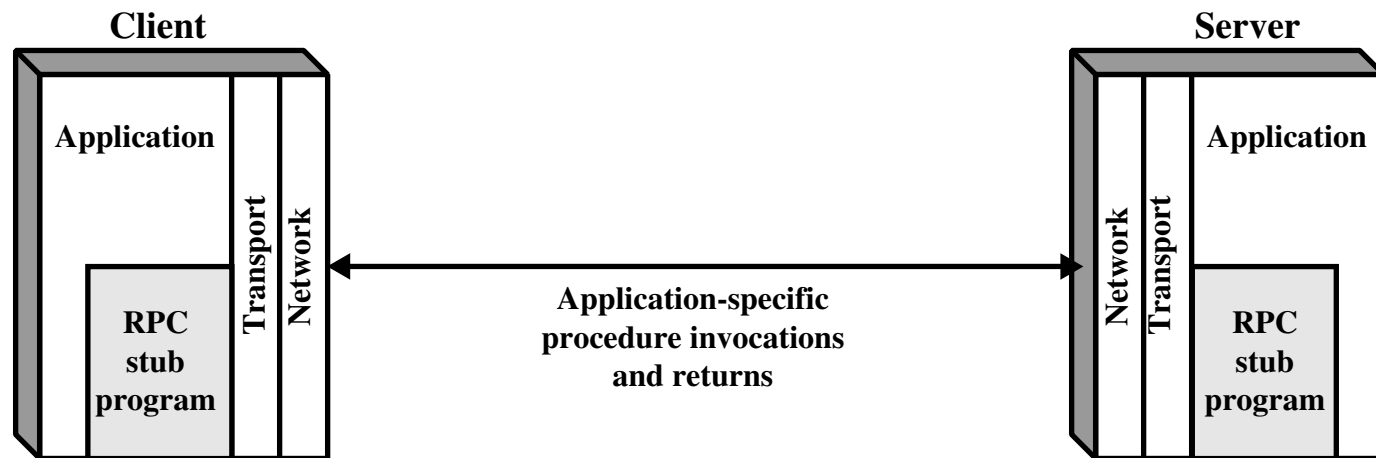
# Blocking vs. Nonblocking

- In a blocking system, send and receive block the process that calls them
  - simplifies application logic
  - a blocking send does not return until the message has been delivered in a reliable service
  - limits concurrency
- In a nonblocking system, a process does not have to wait for send and receive to complete
  - more flexible
  - susceptible to timing errors, making application logic and debugging harder

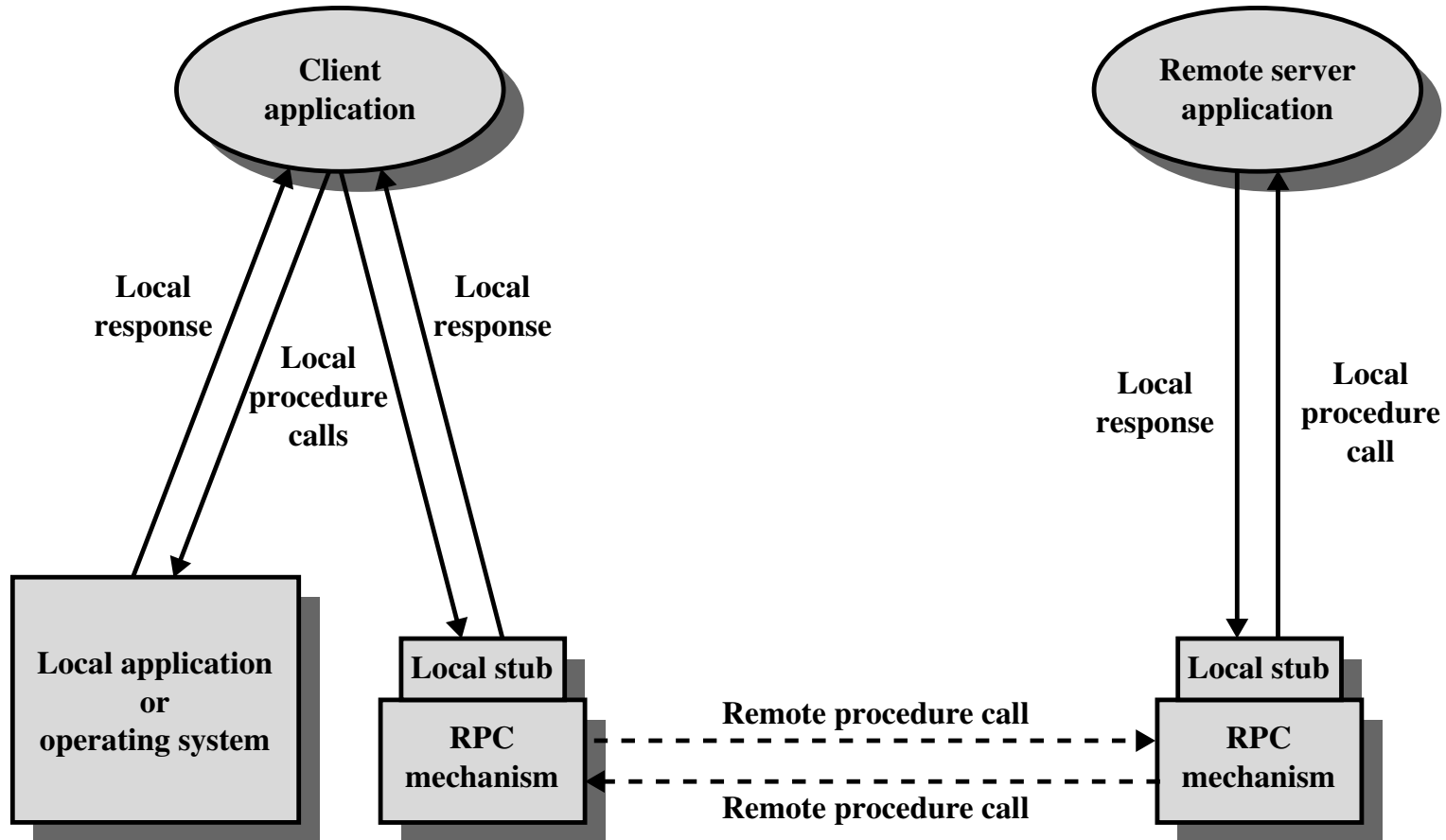
# Remote Procedure Calls

- A more sophisticated middleware package
- Makes client server applications look like a normal procedure call
- Client application does not need to actually know if a function is handled by a remote procedure call or not
- Usually are synchronous, but can be asynchronous

# Remote Procedure Calls



# Remote Procedure Calls



# Clusters

- In the client/server model applications are not really distributed
  - Most of the work is typically done on one machine
- A workstation *cluster* is a set of workstations (called *nodes*) that are all able to work on a problem together
- Advantages of clusters
  - highly scalable
  - incremental scalability (able to add machines easily)
  - high availability
  - cost

# Clusters

- There is a wide range of cluster architectures
  - The individual work stations can be very independent or closely coupled
- *Single-system image* architecture
  - Single entry point: user logs onto the cluster, not an individual computer
  - Single file hierarchy: user sees a single file hierarchy
  - Single virtual networking: any node can access any other node, independent of how nodes are physically connected
  - Single process space: a process on any node can create or communicate with any other process on a remote node

# Beowulf

- A cluster technology for PC's running Linux
- Uses mass market commodity components, making it inexpensive
- The PC's are connected by a dedicated, private network
- Each node can operate as an autonomous Linux system
- Cluster can also have a single-system image

# Process Migration

- In an advanced distributed OS's, a process should be able to migrate from one process to another
  - Load balancing
  - Fairness
  - Failure management
- True process migration is rare