Client – Server Architecture

Client / Server and Clusters
Basic of RPC

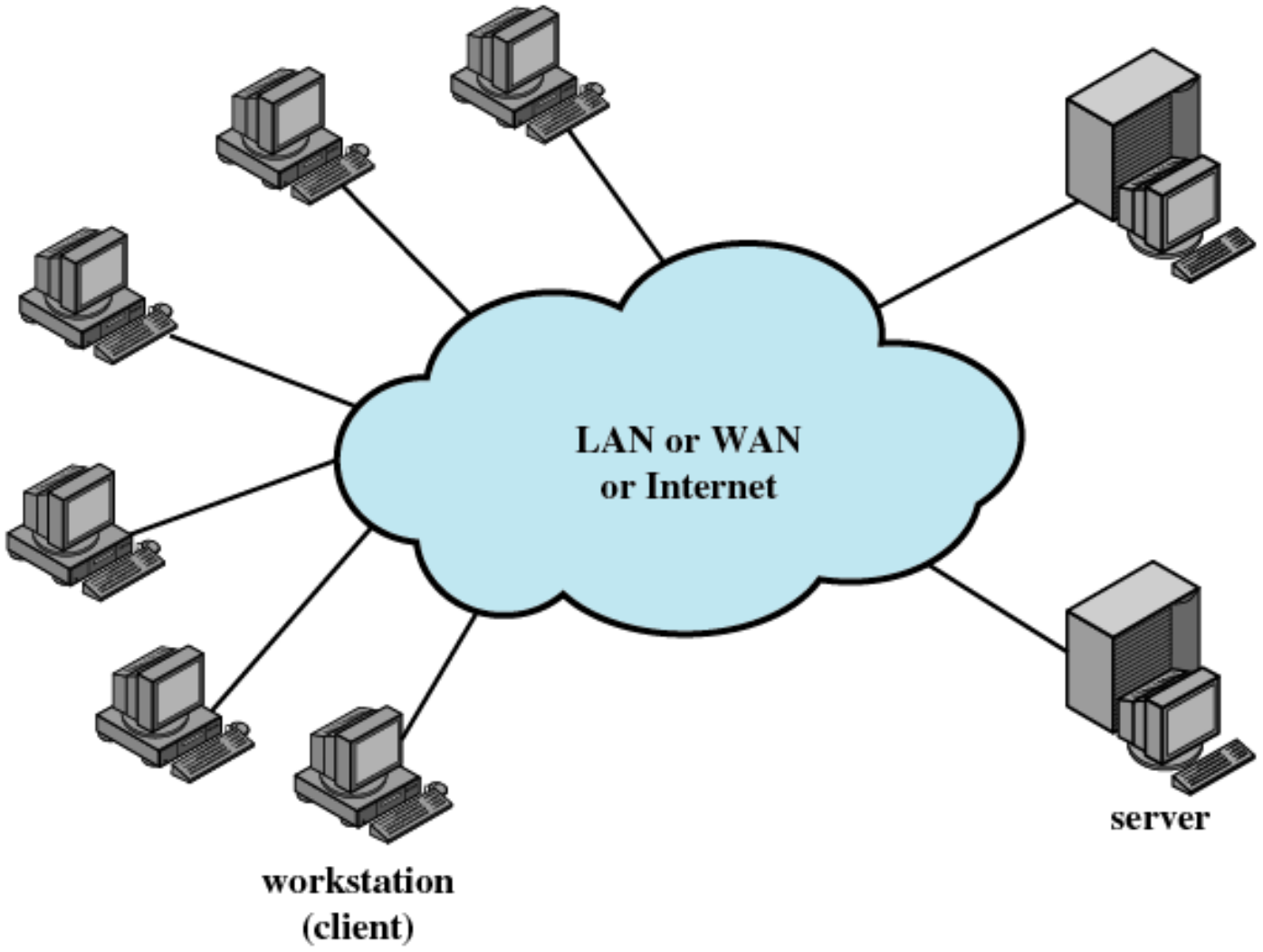
ARASH HABIBI LASHKARI
(April - 2011)

Client / server computing

- Client machines are usually single-user PCs or workstations that provide a user-friendly interface to the end user
- Each server provides a set of shared services to the clients
- The server enables many clients to share access to the same database and enables the use of a high-performance computer system to manage the database

Client/Server Terminology

- **Applications Programming Interface (API)**
  - A set of functions and call programs that allow clients and servers to intercommunicate
- **Client**
  - A networked information requester, usually a PC or workstation, that can query database and/or other information from a server
- **Middleware**
  - A set of drivers, APIs, or other software that improves connectivity between a client application and a server
- **Relational Database**
  - A database in which information access is limited to the selection of rows that satisfy all search criteria
- **Server**
  - A computer, usually a high-powered workstation, a minicomputer, or a mainframe, that houses information for manipulation by networked clients
- **Structured Query Language (SQL)**
  - A language developed by IBM and standardized by ANSI for addressing, creating, updating, or querying relational databases



Generic Client/Server Environment

Server types

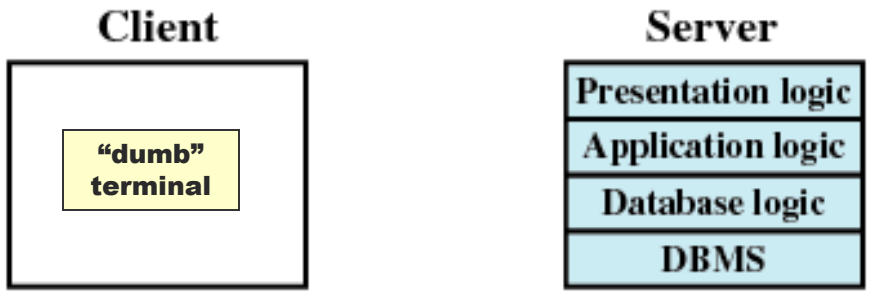
- Disk server
  - Provides shared drive like Z:
- File server
  - Provides logical structure, protection, concurrent access, etc.
- Print server
  - Provides SPOOLing
    - Simultaneous Peripheral Operation On-Line

Server types

- Database server
  - Searches for and returns a small number of records
  - Server software is typically SQL
  - Client does data analysis with application software
  - Server can do more work if analysis involves a large number of records
    - It is not efficient to send 1,000,000 records over a network
    - This requires some application-specific software to reside on the server

Classes of Client/Server Applications

- Host-based processing
  - Not true client/server computing
  - Traditional mainframe environment



(a) Host-based processing

Classes of Client/Server Applications

- Server-based processing
  - Server does all the processing
  - Client provides a graphical user interface

(b) Server-based processing

Classes of Client/Server Applications

- Cooperative processing
  - Application processing is performed in an optimized fashion
  - Complex to set up and maintain

(c) Cooperative processing

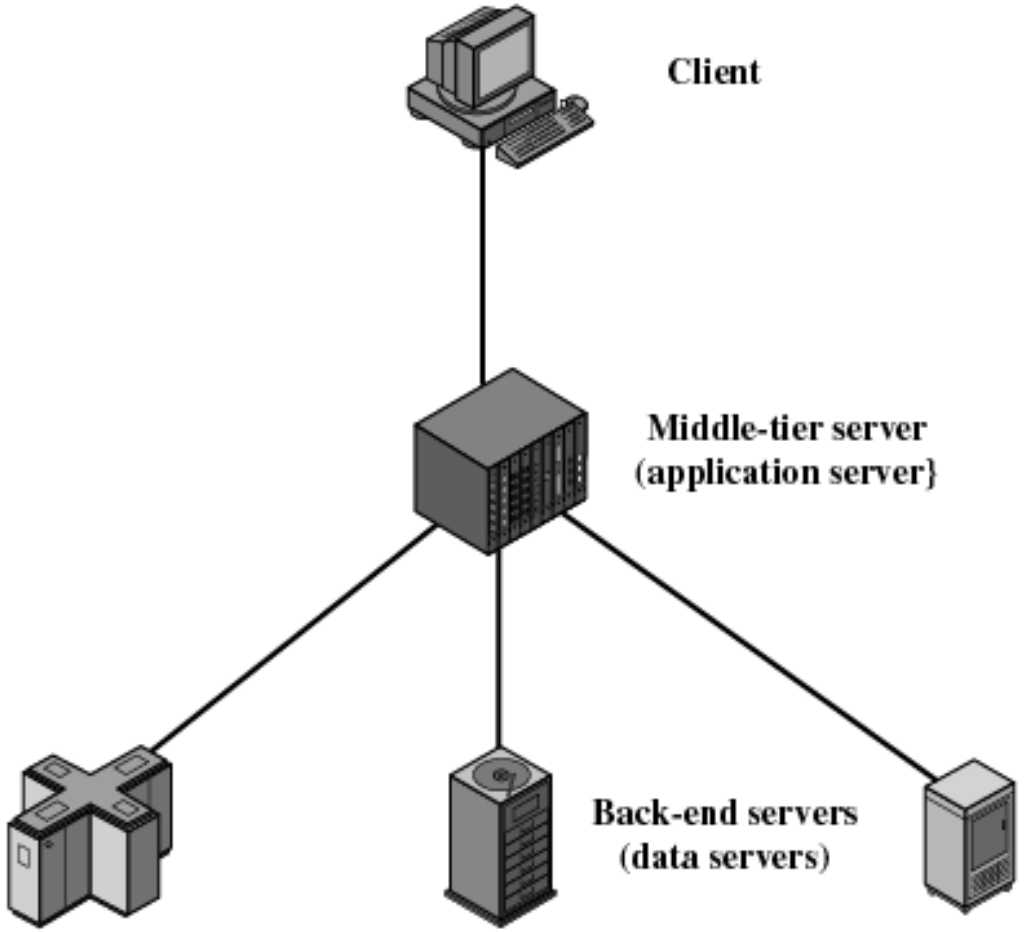
Classes of Client/Server Applications

- Client-based processing
  - All application processing done at the client
  - Data validation routines and other database logic functions are done at the server

(d) Client-based processing

Three-Tier Client/Server Architecture

- Increasingly, this architecture is used
- Application software distributed among three types of machines
  - User machine
    - Thin client
  - Middle-tier server
    - Gateway
    - Conversion protocols
    - Merge/integrate results from different data sources
  - Backend server



Three-tier Client/Server Architecture

File Cache Consistency

- File caches hold recently accessed file records on servers and each client
- Various caches must be kept consistent
- More complicated than readers/writers problem
  - Writer must write back to disk and server must notify all readers to not use cache copy

Distributed Message Passing

- Message passing must be used for interprocess communication and synchronization
  - This is because no shared memory exists
  - Therefore, no semaphores
- Two models
  - Client-server
  - Remote Procedure Call (RPC)

Client-server model

- Involves sending and receiving messages using primitives as in a single system
  - Send(destination, message)
  - Receive(from, messageBuffer)
- Note: the “from” parameter may include “all”
- The primitives rely on a “communications” distributed system architecture like TCP / IP

Reliable v. unreliable message passing

- Reliable
  - Guarantees delivery if possible
  - Not necessary to let the sending process know that the message was delivered
  - High overhead
- Unreliable
  - Sends the message out to the network without reporting success or failure
  - Reduces complexity and overhead
  - Destination may need/want to send acknowledgement

Blocking v. nonblocking

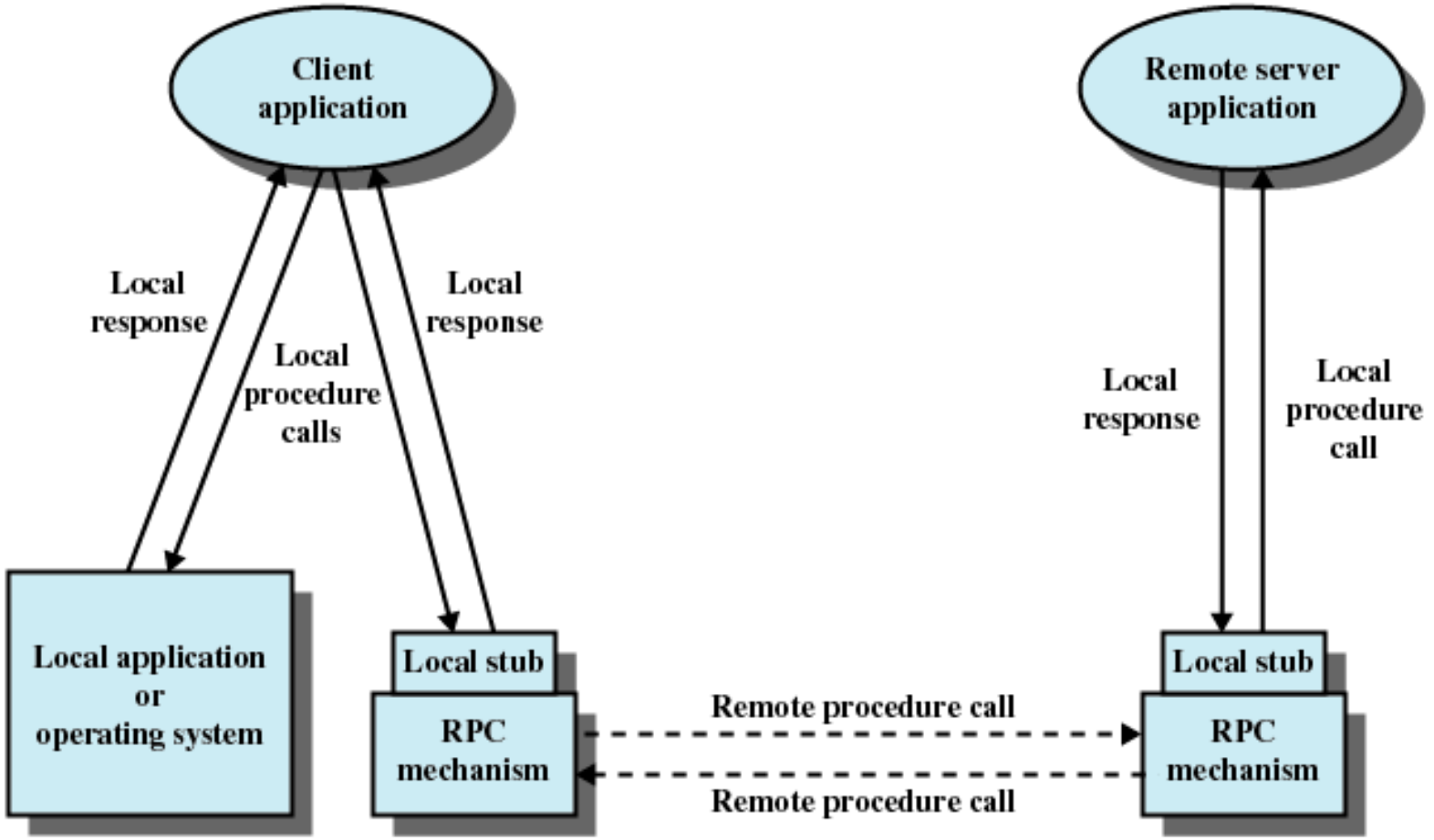
- Nonblocking
  - Process is not suspended as a result of issuing a Send or Receive
  - Sender released after a copy of the message is made
  - Efficient and flexible
  - Difficult to debug
- Blocking
  - Send does not return control to the sending process until the message has been transmitted (reliable case)
  - OR does not return control until an acknowledgment is received (unreliable case)
  - Receive does not return until a message has been placed in the allocated buffer

Remote procedure call

- Allows programs on different machines to interact using simple procedure call and return semantics
- Widely accepted
- Standardized
  - Client and server modules can be moved among computers and operating systems easily

Remote procedure call

- Sender has a dummy procedure that sends parameters to the target as a message
- The target reacts to the message by calling the actual procedure
- Results are then sent back as a message and returned by the dummy procedure



Remote Procedure Call Mechanism

Two models of RPC

- Synchronous RPC
  - Special case of reliable, blocking message passing
  - Behaves much like a subroutine call
- Asynchronous RPC
  - Does not block the caller
  - Enables client execution to proceed locally in parallel with server invocation
  - Exploits possibility of parallelism
  - Example of client server synchronization
    - The client can send a sequence of asynchronous RPCs followed by a final synchronous RPC
    - The server responds to the synchronous RPC only after all preceding work has been completed

Object-oriented mechanisms

- Clients and servers ship messages back and forth between objects
- Common Object Request Broker Architecture (CORBA)
  - A client sends a request to an object broker
  - The broker calls the appropriate object and passes along any relevant data

Clusters

- Alternative to symmetric multiprocessing (SMP)
- Group of networked computers, called “nodes”
- Computers work together as a unified computing resource
- Illusion of being one computer
- Provides load balancing among all nodes

Advantages of clusters over SMP

- Scalability
  - Both incremental and absolute
  - Easy to add a new node
  - Very large clusters of multiprocessor systems possible
- Availability
  - Failure of one node does bring system down
  - Easy to make components redundant
  - “Failover” is a switch to an alternate node in case of failure
- Price / performance
  - Building blocks are commodity components

Cluster configurations

- “Separate server”
  - Each node has private (non-shared) disk
  - Failover capability requires constant copying of data to disks on other nodes
- “Shared nothing”
  - Reduces communication overhead
  - Common RAID disk, partitioned into volumes
  - Each volume is owned by a separate node
  - Failure of node only requires transfer of ownership of its volume
- “Shared disk”
  - Each node may access all volumes
  - Mutual exclusion must be enforced

Advantages of SMP over clusters

- SMP is easier to manage and configure
- SMP takes up less space and draws less power
- SMP products are well established and stable

Questions