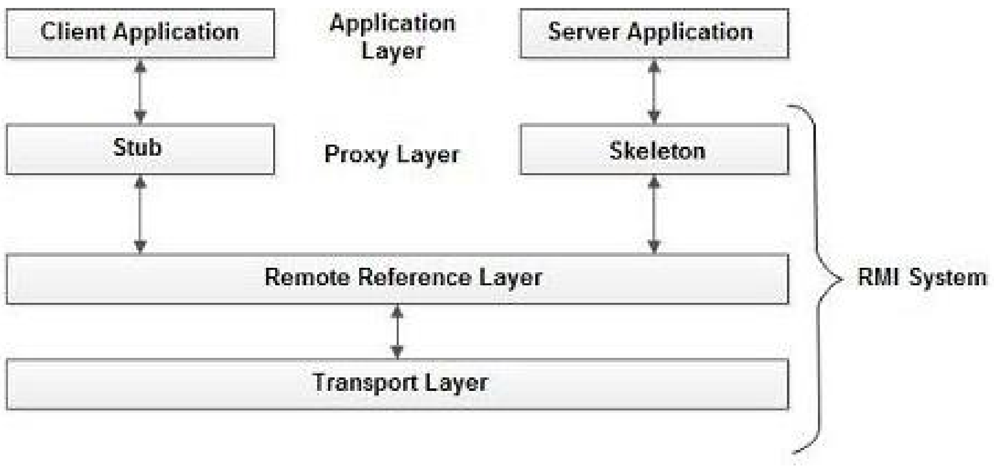
Chapter 2
Distributed Objects and File System

1. Introduction

- Distributed Objects are the objects with respect to Object Oriented Programming Paradigm, that are distributed across multiple address spaces, which may be on multiple computers within a network or multiple processes running within a system/ computers.
- A distributed object is an object that can be accessed remotely. This means that a distributed object can be used like a regular object, but from anywhere on the network.
- Distributed objects might be used :
- to share information across applications or users.
- to synchronize activity across several machines.
- to increase performance associated with a particular task.
- work together by sharing data and invoking methods.
- This often involves location transparency, where remote objects appear the same as local objects.

2. Communication between distributed objects

- The main method of distributed object communication is with *Remote Method Invocation*, generally by message-passing: one object sends a message to another object in a remote machine or process to perform some task. The results are sent back to the calling object.
- Remote Method Invocation (RMI) is an API (Application Programming Interface) that provides a mechanism to create distributed application in JAVA code.
- It provides remote communication between the applications using two intermediate objects: Stub and Skeleton.
- RMI allows an object to invoke method on an object running in another JVM.
- The method invocation between the objects in different processes within a computer or different computer is known as remote method invocation.

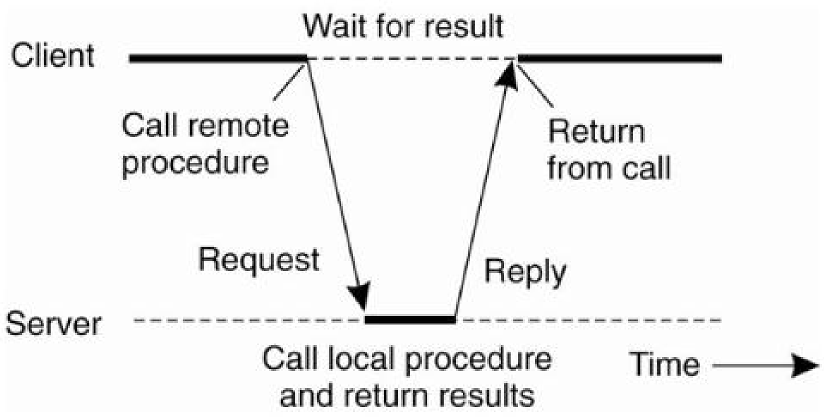


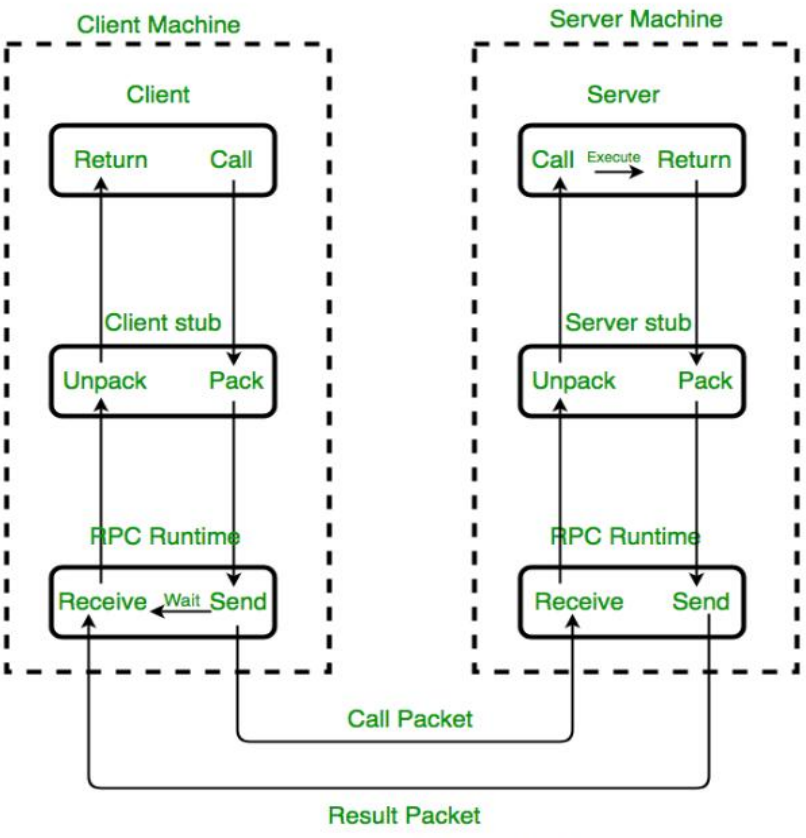
Architecture of RMI

- ✧ Stub: It creates an information block and sends it to the server. The information block consists of ID of remote object, method to invoke on object and parameters to be passed. It receives the response from the server and returns the value.
- ✧ Skeleton: It opens up the information block, calls derived methods, forwards parameters and returns the value to the client stub object.

3. Remote Procedure Call

- ➢ RPC is a remote communication medium in which a client program calls a procedure in another program running in a server process. The role of server and client is designated based on the work which is temporarily assigned. It means the server may also be clients of other servers creating a chain of RPC.
- ➢ It is a interprocess communication technique.



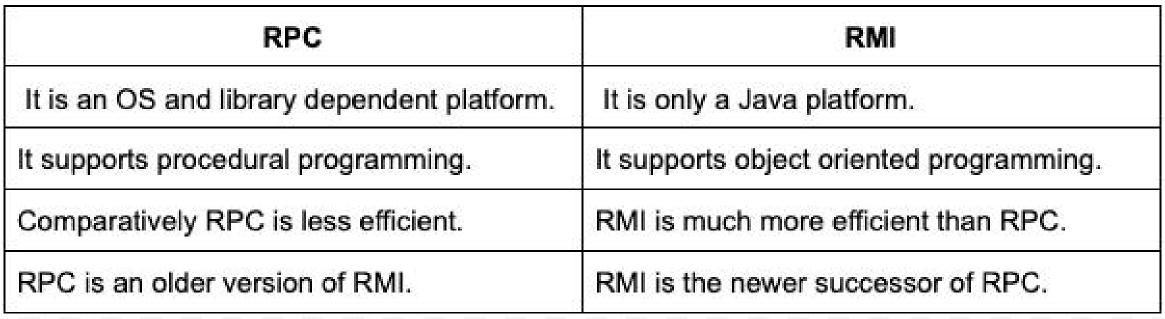


Implementation of RPC mechanism

- Working of RPC
  - Client procedure calls client stub in normal way.
  - Client stub builds message (organized message package consisting of parameters and procedures into a single unit is created), call local OS.
  - Client's OS sends message to remote OS.
  - Remote OS gives message to server stub.
  - Server stub unpacks parameters, call server.
  - Server does work, returns result to the stub.
  - Server stub packs it in message, call local OS.
  - Server's OS sends message to client's OS.
  - Client's OS gives message to client stub.
  - Stub unpacks result, returns to client.

- There are 5 elements of RPC as:
  1. Client
  2. Server
  3. Client Stub
  4. Server Stub
  5. RPC Runtime

- RPC can communicate between processes on the same or different machines.
- RPC method helps clients to communicate with servers by the conventional use of procedure calls in high-level languages.
- Process-oriented and thread-oriented models are supported by RPC.



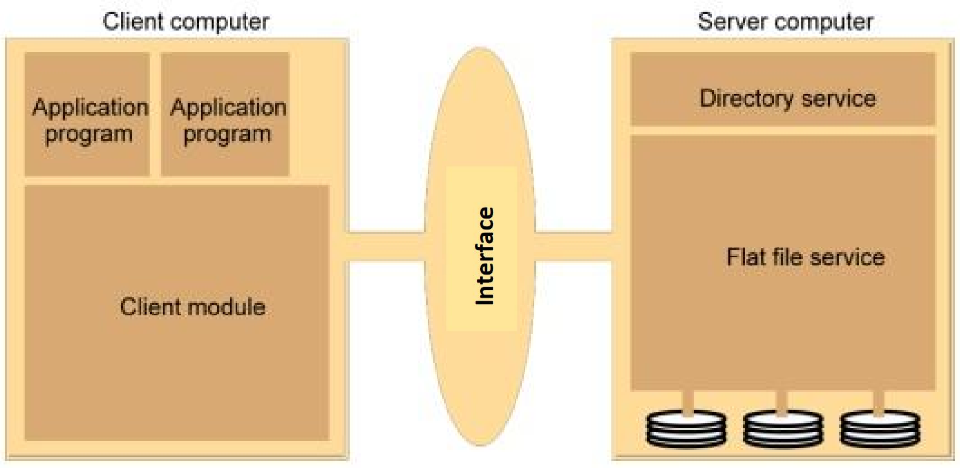
RPC	RMI
It is an OS and library dependent platform.	It is only a Java platform.
It supports procedural programming.	It supports object oriented programming.
Comparatively RPC is less efficient.	RMI is much more efficient than RPC.
RPC is an older version of RMI.	RMI is the newer successor of RPC.

Assignment

4. **Events And Notifications**
5. **Java RMI Case Study**

6. Introduction to DFS

- ➢ Distributed File System (DFS) a file system with data stored on a server. The data is accessed and processed as if it was stored on the local client machine.
- ➢ DFS makes it convenient to share information and files among users on a network in a controlled and authorized way. The server allows the client users to share files and store data just like they are storing the information locally.
- ➢ However, the servers have full control over the data and give access control to the clients. One process involved in implementing the DFS is giving access control and storage management controls to the client system in a centralized way, managed by the servers.
- ➢ Transparency is one of the core processes in DFS, so files are accessed, stored, and managed on the local client machines while the process itself is actually held on the servers. This transparency brings convenience to the end user on a client machine because the network file system efficiently manages all the processes.
- ➢ Requirements of DFS:
  - ❖ The design of DFS must support transparency of access, location, migration, performance, etc.
  - ❖ Concurrent file updates should be controlled.
  - ❖ It should support file replication to enhance scalability and fault tolerance.
  - ❖ The service interfaces should be defined such that they can be implemented in heterogeneous systems.
  - ❖ The service should continue to operate even in case of client or server failures.
  - ❖ It should maintain consistent states for files.
  - ❖ It should provide access control mechanisms to provide security.



- ❖ DFS Consists of 3 components:
  - ✧ <u>Flat File Service</u>- It is a component responsible for implementing operations on the file contents. During requests for file service operations, Unique File Identifier (UFID) is used to indicate the files. The various flat file service operations are: Read, write, create, delete, getAttributes and setAttributes.
  - ✧ <u>Directory Service</u>-It is a component that provides mapping between text names for files and their UFID. The various directory service operations are: lookup, addName, unName and getNames.
  - ✧ <u>Client Module</u>- It is a component running in client computer, integrating the file service and directory service operations under a single API. With the use of cache, client module helps to achieve high performance.

7. File Service Architecture

- File system provides an abstract view of secondary storage and is responsible for global naming, file access, and overall file organization. These functions are handled by the name service, the file service, and the directory service.
- File system is necessary for organization, storage, retrieval, naming, sharing and protection of files.
- Files contain data and attributes.
- DFS provides services to the clients of distributed system.

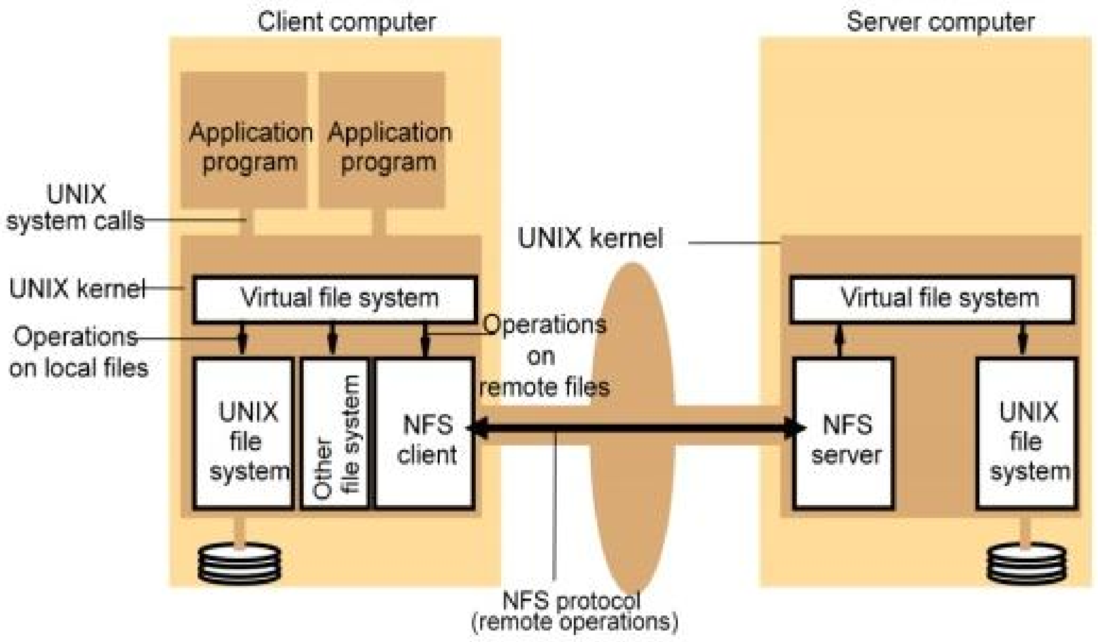
- File service is the specification of what the file system offers to its clients. It is a process that runs on some machine and helps implement the file service.

8. Sun Network File System (SUN NFS)

- SUN NFS is a first successful network file system developed by SUN microsystems that focuses on transparency.

- Properties:
  - It is both an implementation and a specification of how to access remote files.
  - It focuses on sharing a file system in a transparent way.
  - It uses client server model. A node can act both as client and server.
  - It uses mount to make server file system visible from a client.
  - It is stateless (all client requests must be self contained).
  - It is machine and operating system independent.

- Architecture:



1. *Protocol*: It uses SUN RPC mechanism and SUN external data representation (XDR) standard. The protocol is stateless. It enhances crash recovery. Each procedure call must contain all the information necessary to complete the call.

2. *Server Side*: It provides file handle consisting of:
   a) Filesystem id (identify disk partition)- stored in super block.
   b) I-node number (identify file)
   c) Generation number- stored in I-node.

3. *Client Side*: It provides transparent interface to NFS. Mapping between remote file name and remote file address is done at server boot time through remote mount.

❖ Operations:
- ✧ Search for file within directory
- ✧ Read a set of directory entries
- ✧ Manipulate links and directories
- ✧ Read/write file attributes
- ✧ Read/write file data

❖ *SUN RPC*
- ✧ SUN RPC was designed for client-server communication in SUN NFS.
- ✧ It can be used either over TCP or UDP.
- ✧ It provides an Interface Definition Language (IDL) called XDR (External Data Representation) and a compiler called “rpcgen” used with C language.
- ✧ In DS, it is important that new objects are able to be sent to any platform and discover how to run in that environment. IDL allows program written in one language to communicate with program written in another language.
- ✧ XDR is used to define the service interface by specifying a set of procedure definitions and supporting type definitions. It provides a program number and version number rather than interface name. The program number varies for each program and the version number changes if the procedure changes.
- ✧ The program number and the version number are passed in request message to ensure both the client and the server are using the same version.

9. Introduction to Name Services
10. Name Services and DNS

❖ Name Services
- ✧ Names are used because they are human readable.
- ✧ Names are used to refer various resources. Resources are accessed using identifier or references.
- ✧ A name is a value that can be resolved to an identifier or an address.
- ✧ Name services are those services that are used to resolve resource names.
- ✧ Name service returns the information about the resource when the name of a resource is given.
- ✧ It is responsible to store a collection of one or more naming contexts.

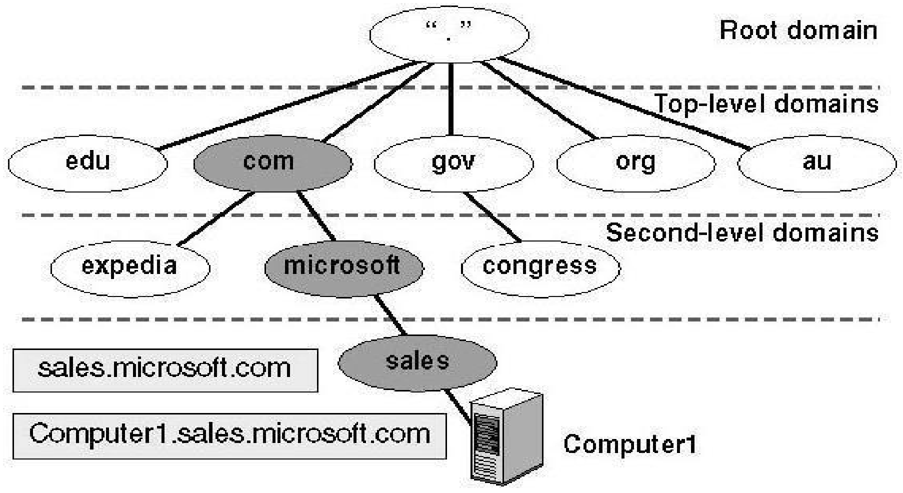
❖ Domain Name System (DNS)

A hierarchical, distributed database that contains mappings of DNS domain names to various types of data, such as IP addresses is called DNS. DNS enables the location of computers and services by user-friendly names, and it also enables the discovery of other information stored in the database.

DNS is application layer (layer 7) protocol that listens on UDP port number 53. DNS names are user-friendly so easy to remember than IP addresses. DNS names remain more constant than IP addresses as IP addresses of servers may change but server name never changes.

DNS Servers- DNS Servers are the servers that work together to provide IP addresses of the requested website to the web browser. There are 4 types of DNS Servers as:

1. Root Name Server
2. TLD(Top Level Domain)Name Server
3. Authoritative Name Server
4. DNS Resolver



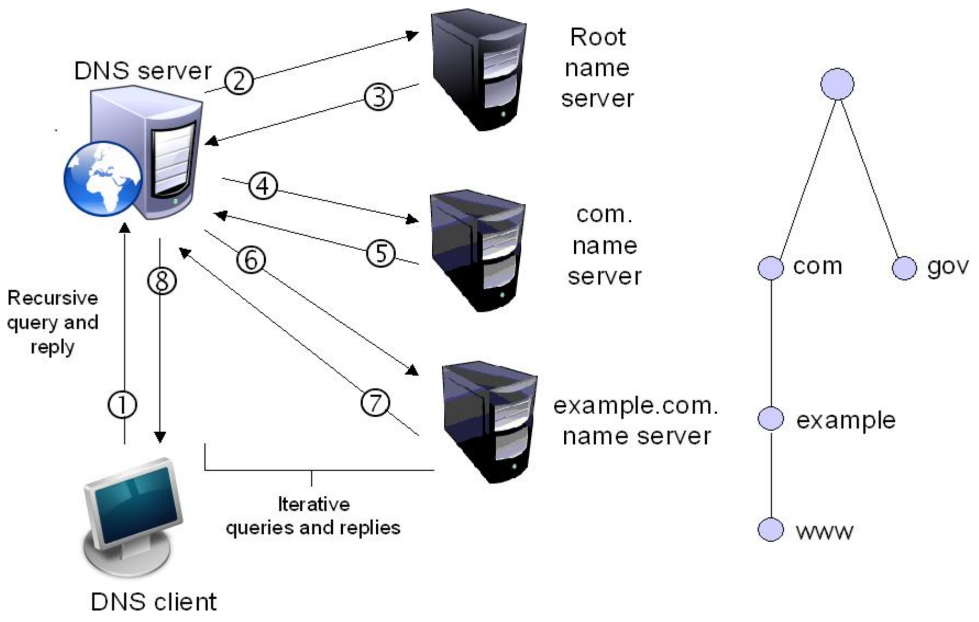
Hierarchical Structure of Domain Namespace

Root Domain - The root domain is at the top of the hierarchy and is represented as a period (.). The Internet root domain is managed by several organizations, including Network Solutions, Inc. There are a total of 13 Root Name Servers.

Top Level Domain - 2 to 3 characters name-code grouped by organizational type or geographic locations.

Top-level domain	Description
gov	Organizations Government
com	Commercial organizations
edu	Educational institutions
org	Noncommercial organizations
au	Country code of Australia

Authoritative Level Domain - The domain concerned with specific websites address is called authoritative level domain.



DNS Name Resolution

Working: When user enters URL on DNS Client, computer's OS contacts DNS Resolver (Server). DNS Server checks its cache for the respective IP address of URL. If not found, forwards DNS query to root name server.

Root name server checks the extension and provides the IP address of TLD name server to the DNS Server. DNS Server contacts the TLD name server which then provides IP address of Authoritative name server. DNS Server again contacts the Authoritative name server which then provides the exact IP address.

DNS Server caches the information and provides IP address to the user's web browser.

Root Zone Database- Root Zone Database represents the delegation details of top-level domains, including TLDs such as .com, and country-code TLDs such as .uk. As the manager of the DNS root zone, IANA is responsible for coordinating these delegations in accordance with its policies and procedures.

Extra:https://www.iana.org/domains/root/db

Reasons to not have a centralized DNS

1. Single point of failure
2. High Traffic Volume
3. Distant Centralized Database
4. Difficult Maintenance

Forward Lookup-Resolves name to an IP address

Reverse Lookup- Resolves IP address to a name

11. Directory and Discovery Services (Self)

12. Comparison of Different Distributed File Systems (Self)