Distributed Operating System

Presented by: Avichandra Singh Ningthoujam, Ph.D.

Manipal University Jaipur

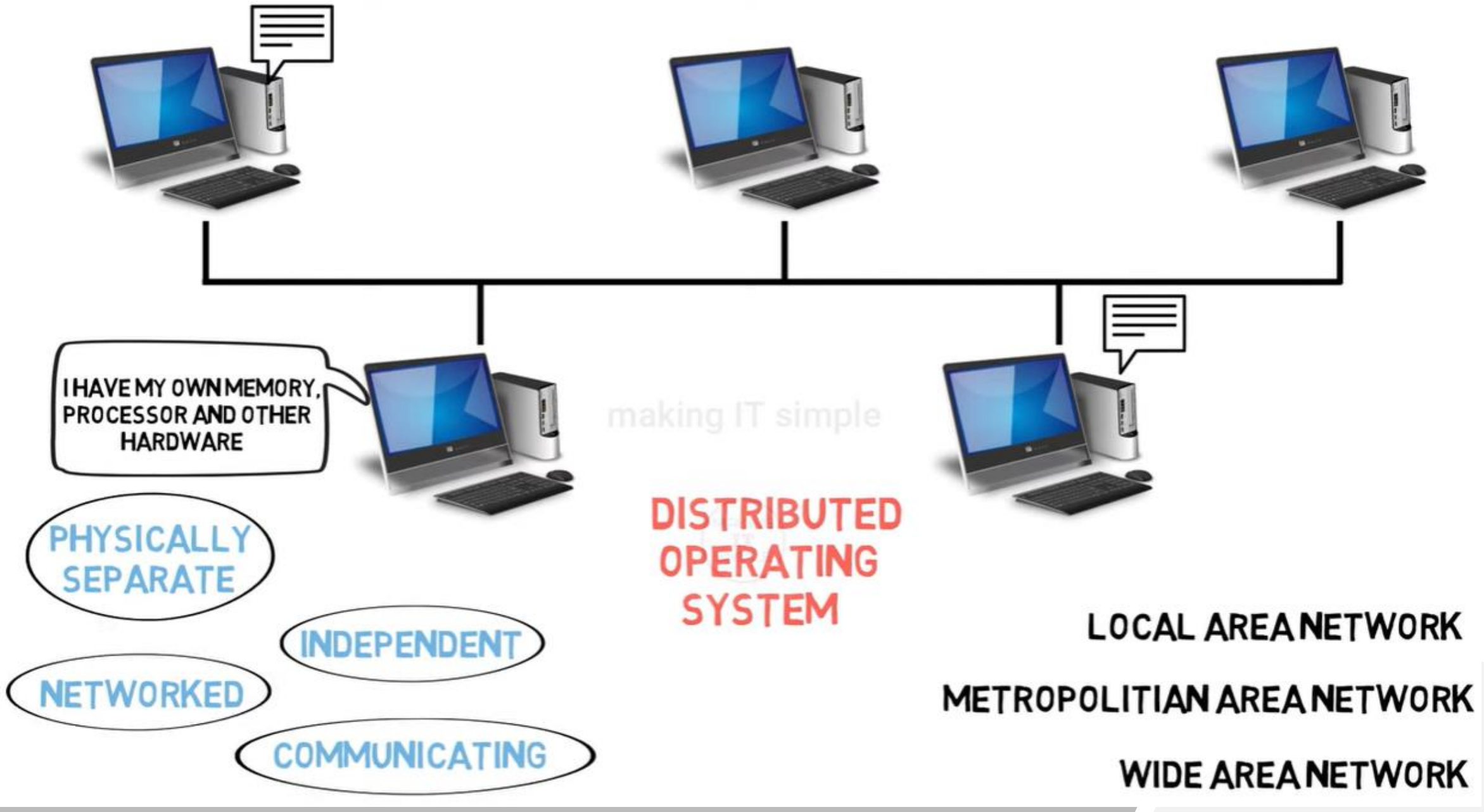
Date: 10 Feb. 2025

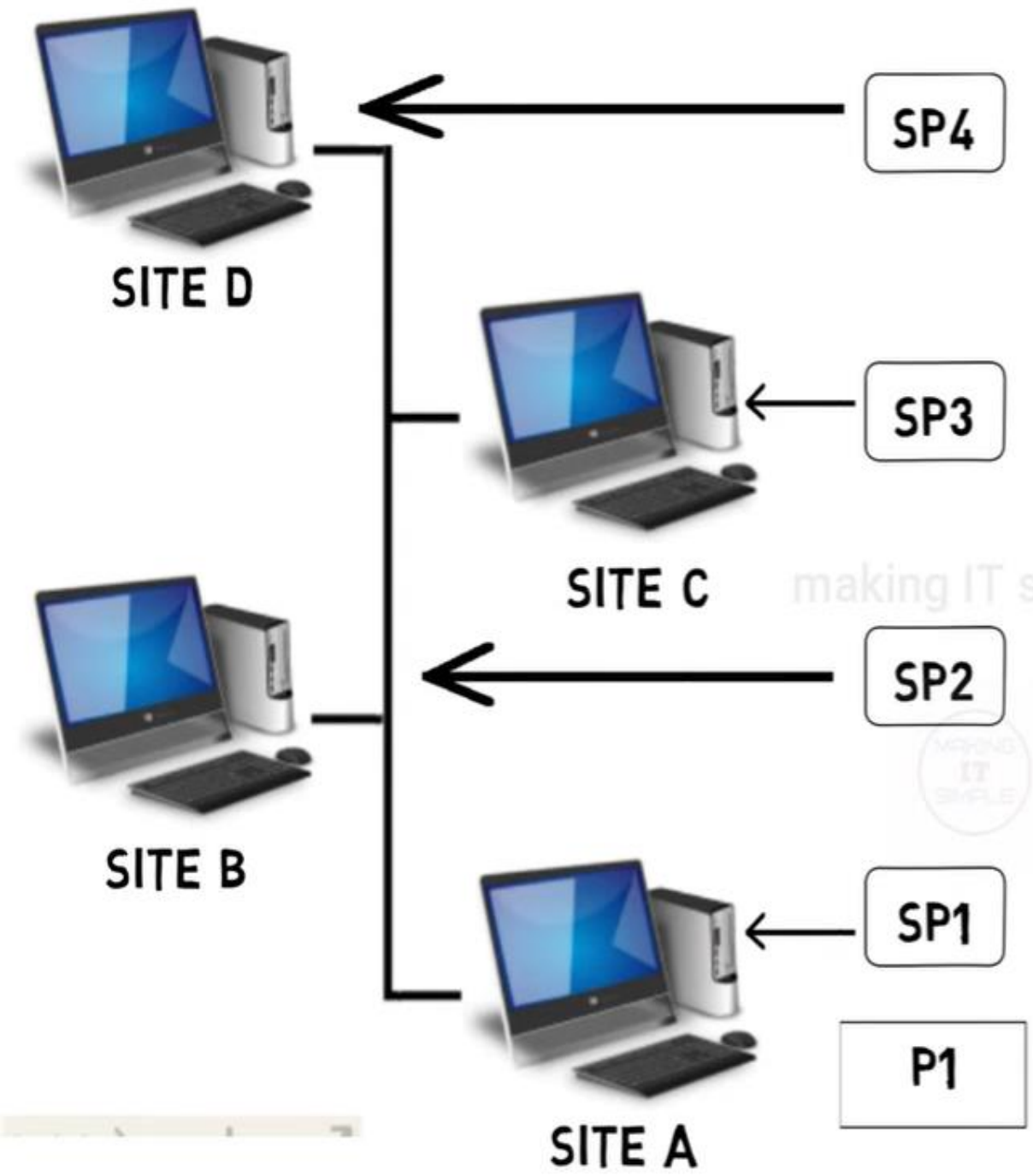
What is Distributed Operating System

- Collection of **independent** computer **interconnected** via **network** and capable of **collaborating** task.

- This operating system consists of numerous **computers**, **nodes**, and **sites** joined together via **LAN/WAN** lines.

- Distributed operating systems can share their **computing** resources and **I/O** files while providing users with virtual machine abstraction.
- Example:
  - Telephone network & cellular network
  - Computer network such as internet
  - ATM


I HAVE MY OWN MEMORY, PROCESSOR AND OTHER HARDWARE
PHYSICALLY SEPARATE
INDEPENDENT
NETWORKED
COMMUNICATING
DISTRIBUTED OPERATING SYSTEM
LOCAL AREA NETWORK
METROPOLITIAN AREA NETWORK
WIDE AREA NETWORK



1. Load Balancing
   - ✓Work is evenly distributed
2. Computational Speedup
   - ✓Computational time will be less
3. Hardware Preference
   - ✓Need for specialized hardware at different site
4. Data Access
   - ✓Instead of importing file, process is executed at where the file are present.

Design Issues of Distributed System

- **Heterogeneity**
  - The components of distributed systems may differ and vary in **operating systems**, **networks**, **programming languages**, **computer hardware**, and implementations by different developers
  - Different computers in the network may have different **architectures**, **operating systems**, or **configurations**, requiring the system to handle diverse components seamlessly to function effectively; essentially, it's the ability of a distributed system to **operate** on **various hardware** and **software** components **without issue**.
- **Transparency**
  - The primary purpose of a distributed operating system is to **hide** the fact that **resources** are shared.
  - Transparency also implies that the user should be **unaware** that the **resources** he is accessing are **shared**.
    - Types: **Access, Location, Migration, Replication, Concurrency, Failure transparency**.

Design Issues of Distributed System

- **Scalability**
  - The ability of a system to **accommodate** a **growing load** or **demand** without **compromising** performance is referred to as scalability.
  - A system must be scalable in order to accommodate **growing user traffic**, **data volumes**, or **computing** demands without suffering a major performance hit or necessitating a total redesign.
- **Reliability**
  - Reliability is crucial in system design, ensuring **consistent performance** and **minimal failures**.
  - The reliability of a system is defined as the **probability of performing the intended function over a given period under specified operating conditions**.
- **Availability**
  - A system or service's readiness and **accessibility** to users at any given moment is referred to as availability.
  - Availability is measured as the **percentage of time** a system or service is operational and **accessible** to **users over** a specific period.

Design Issues of Distributed System

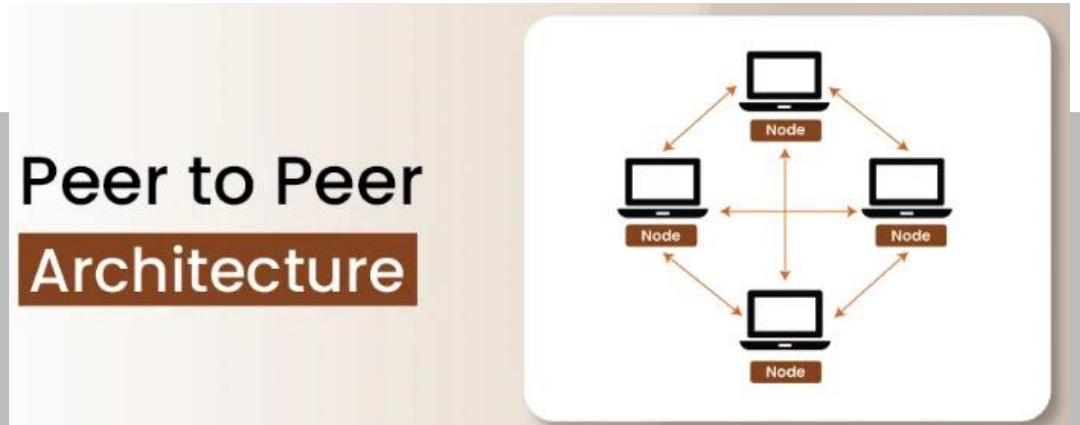
- **Consistency**
  - Consistency in system design refers to the property of **ensuring** that all nodes in a distributed system have the **same view** of the data at any given point in time, despite possible **concurrent operations** and **network delays**.
- **Latency**
  - Latency is the time it takes for data or a signal to travel between two points of a system.
  - It combines a number of delays – Response times, transmission, and processing time.
- **Load Balancing**
  - A load balancer is a crucial component in system design that distributes incoming network traffic across multiple servers.
  - Its main purpose is to ensure that no single server is overburdened with too many requests, which helps improve the performance, reliability, and availability of applications.

Types of Distributed Operating Systems Based on Architecture

- **Client-Server System**
  - This type of system needs the client to request some resource, which is then provided by the server. When any client connects to the server, the server may simultaneously serve many clients.
  - Clients are devices or programs that make requests for services or resources, while the server is a powerful machine or software that fulfills these requests.
  - Communication between clients and the server follows a request-response protocol, such as HTTP/HTTPS for web services or SQL for database queries.

Peer-to-Peer Architecture

- **Peer-to-Peer System**
  - Peer-to-peer (P2P) architecture is a **decentralized** computing model where network participants share **resources directly** with each other without the need for a centralized server.
  - In P2P each node acts as both a client and a server, enabling distributed sharing of files, data, and computing resources.
  - These nodes can also share resources and information as required. They require some network to connect once more.



Distributed Operating Systems Based on Architecture

- **Middleware**
  - In distributed systems, middleware is a **software component** that provides **services** between two or more applications and can be used by them.
  - Middleware can be thought of as an application that sits between two separate applications and provides service to both..
- **Three-tier**
  - The Three-Tier Client-Server Architecture divides systems into presentation, application, and data layers, increasing scalability, maintainability, and efficiency.
  - This model optimizes resource management and allows for independent scaling and updates, making it a popular choice for complex distributed systems.
- **N-tier**
  - N-tier systems are utilised when a server or application has to send requests to other enterprise services over the network.

Applications of Distributed Operating System

- **Network Applications**
  - DOS is used by many network applications, including the Web, peer-to-peer networks, multiplayer web-based games, and virtual communities.
- **Telecommunication Networks**
  - DOS is useful in phones and cellular networks. A DOS can be found in networks like the Internet, wireless sensor networks, and routing algorithms.
- **Parallel Computation**
  - DOS is the basis of systematic computing, which includes cluster computing and grid computing, and a variety of volunteer computing projects.
- **Real-Time Process Control**
  - The real-time process control system operates with a deadline, and such examples include aircraft control systems.

Pros and Cons of DOS

- **Pros**
  - Share all resources between sites, enhancing data availability throughout the system.
  - Since all information is replicated across all sites, the risk of data corruption is reduced
  - Because the entire system is independent of one another, if one site crashes, the complete system does not come to a halt.
  - It improves the speed with which data is sent from one site to another.
  - It aids in the speeding up of data processing.
- **Cons**
  - It is difficult to establish appropriate security in DOS.
  - The database attached to a DOS is complicated and difficult to operate.
  - Communication delay will increase when a system becomes more broadly spread.
  - High cost.

What is File System

- A file system is a way to organize and manage files on a storage device like a hard drive, USB flash drive, or solid state drive (SSD). It also defines how data is accessed and stored on the device.

- What we can do in File System?

What is Distributed File System

- A Distributed File System (DFS) is a file system that is **distributed** on **multiple file servers** or **multiple** locations.

- It allows programs to **access** or **store** isolated files as they do with the **local ones**, allowing programmers to **access files** from any network or computer.

- This setup not only **improves performance** by enabling **parallel access** but also **simplifies data sharing** and **collaboration** among users.

- **DFS's primary goal is to enable users of physically distributed systems to share resources and information through the Common File System (CFS**

Distributed File System Supports

- Remote Information Sharing
- User Mobility
- Diskless Workstation
- Availability

- **Services Provide**
  - Storage service
    - In secondary storages how, where, and management(how much space and for)
  - Truefile Service
    - It support individual file
    - Creating, Deleting, Accessing, Modifying a file, File are store in log file
  - Name Service

Issues and Goal of DFS

- **Location Transparency:**
  - The ability to access a file without needing to know its physical location on the network.
  - The user can retrieve a file using a name or identifier, regardless of which server it's stored on.
  - **Example: Web browsing:** When you access a website using a domain name (like "google.com"), the Domain Name System (DNS) translates that name to an IP address, effectively hiding the physical location of the web server.
- **Redundancy:**
  - The practice of storing multiple copies of the same file across different nodes or servers within the system.
  - Ensuring that even if one node fails, the file remains accessible from other copies, thus maintaining data availability and fault tolerance.
  - It's a backup mechanism for files in a distributed environment.

Features of DFS

- **Transparency:**
  - ***Structure Transparency***.
    - The client does not need to be aware of the number or location of file servers and storage devices
  - ***Naming Transparency.***
    - There should be no hint of the file's location in the file's name. When the file is transferred form one node to other, the file name should not be changed
  - ***Access Transparency.***
    - Local and remote files must be accessible in the same method. The file system must automatically locate the accessed file and deliver it to the client.
  - ***Replication Transparency.***
    - When a file is copied across various nodes, the copies files and their locations must be hidden from one node to the next

Features of DFS

- **Scalability**
  - The distributed system will inevitably increase over time when more machines are added to the network, or two networks are linked together.
- **Data Integrity**
  - Many users usually share a file system. The file system needs to secure the integrity of data saved in a transferred file.
- **High Reliability**
  - The risk of data loss must be limited as much as feasible in an effective DFS. Users must not feel compelled to make backups of their files due to the system's unreliability.
- **High Availability**
  - A DFS should be able to function in the case of a partial failure, like a node failure, a storage device crash, and a link failure.
- **Ease of Use**
  - The UI of a file system in multiprogramming must be simple, and the commands in the file must be minimal

Pros and Cons of DFS

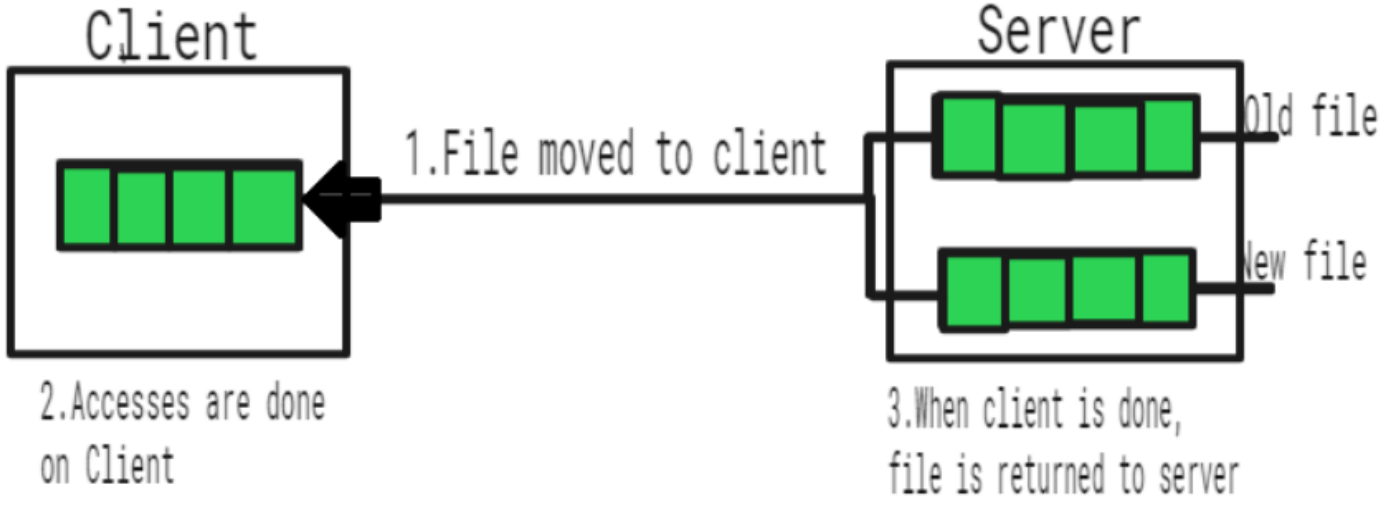
- **Pros**
  - It allows the users to access and store the data.
  - It helps to improve the access time, network efficiency, and availability of files.
  - It permits the data to be shared remotely.
  - It helps to enhance the ability to change the amount of data and exchange data.
- **Cons**
  - The database connection is complicated.
  - Database handling is also more complex than in a single-user system.
  - If all nodes try to transfer data simultaneously, there is a chance that overloading will happen.
  - There is a possibility that messages and data would be missed in the network while moving from one node to another.

Remote File Access

- **Remote file sharing (RFS)** is a type of **distributed file system technology**.
- It enables file and/or data access to multiple remote users over the Internet or a network connection.
- It is also known as a general process of providing remote user access to locally stored files and/or data.
- Files can be shared across the network via variety of methods:
  - Using FTP i.e., file transfer protocol is used to transfer file from one computer to other.
  - Using distributed file system (DFS) in which remote directories are visible from local machine.
  - Using Remote File System (RFS) in which the arrival of networks has allowed communication between remote computer.

Remote File Access

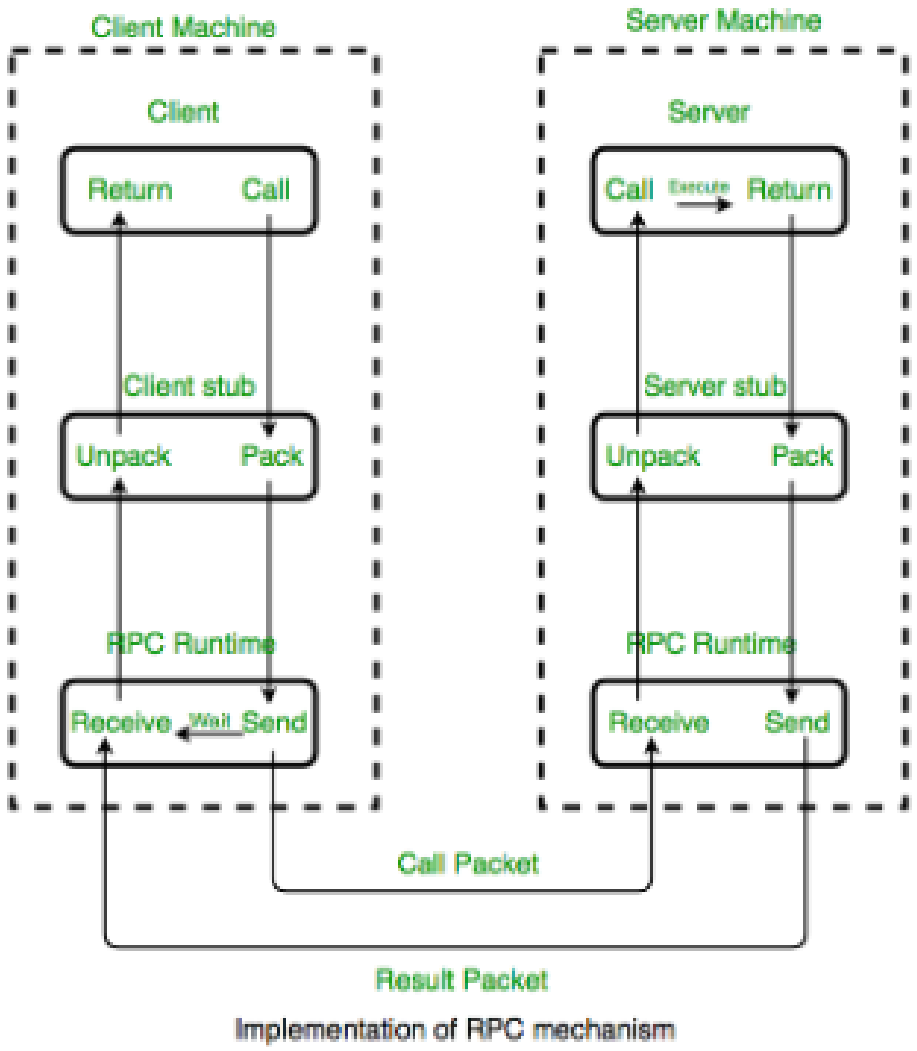
- To implement remote file system we use **client-server model.** It was one of the basic application of Remote File System.
  - In this case, the machine containing the files is server and the machine wanting access to the files is the client. The server specifies which file can be accessed by a particular client(s).
  - A server can serve **multiple clients**, and a client can access **multiple servers**, depending on the implementation details of a given client-server facility. Once it is mounted, file operation requests are sent on the behalf of the user to the server, via network.



Remote Procedure Call (RPC)

- Remote Procedure Call (RPC) is a type of technology used in computing to enable a program to request a service from software located on another computer in a network without needing to understand the network's details.
- RPC works by allowing one program (a client) to directly call procedures (functions) on another machine (the server).
- The client makes a procedure call that appears to be local but is run on a remote machine.
- When an RPC is made, the calling arguments are packaged and transmitted across the network to the server.
- The server unpacks the arguments, performs the desired procedure, and sends the results back to the client.

Working of RPC



Implementation of RPC mechanism

- A client invokes a client stub procedure (It is an intermediary between the client application and the network, responsible for packaging function arguments into a format suitable for transmission to the server (marshalling) and then receiving and unpacking the server's response (unmarshalling).
- The client stub marshalls (pack) the parameters into a message. Marshalling includes converting the representation of the parameters into a standard format, and copying each parameter into the message.
- The client stub passes the message to the transport layer, which sends it to the remote server machine. On the server, the transport layer passes the message to a server stub, which demarshalls(unpack) the parameters.
- When the server procedure completes, it returns to the server stub (e.g., via a normal procedure call return), which marshalls the return values into a message.
- The server stub then hands the message to the transport layer. The transport layer sends the result message back to the client transport layer, which hands the message back to the client stub.
- The client stub demarshalls the return parameters and execution returns to the caller.

Types of RPC

- **Callback RPC:** Callback RPC allows processes to act as both clients and servers. It helps with remote processing of interactive applications. The server gets a handle to the client, and the client waits during the callback. This type of RPC manages callback deadlocks and enables peer-to-peer communication between processes.

- **Broadcast RPC:** In Broadcast RPC, a client's request is sent to all servers on the network that can handle it.

- **Batch-mode RPC:** Batch-mode RPC collects multiple RPC requests on the client side and sends them to the server in one batch. This reduces the overhead of sending many separate requests.

Distributed Shared Memory

- It is a mechanism that manages memory across multiple nodes and makes inter-process communications transparent to end-users.

- DSM is a mechanism of allowing user processes to access shared data without using inter-process communications.

- In DSM every node has its own memory and provides memory read and write services and it provides consistency protocols.