Distributed File System

This is a resource management component of a DOS. It implements a common file system that can be shared by all the autonomous computers in the system. Two important goals of DFS:

- **Network transparency:** the primary goal of DFS is to provide the same functional capabilities to access files distributed over a network as the file system of a timesharing mainframe system does to access files residing at one location. Ideally, users do not have to be aware of the location of files to access them. This property of a distributed file system is known as network transparency.

Contd.

- **High availability:** another major goal is to provide high availability. Users should have the same easy access to files, irrespective of their physical location. System failures or regularly scheduled activities such as backups or maintenance should not result in the unavailability of files.

Contd.

Architecture of Distributed File System

In distributed file system(DFS) files can be stored at any m/c(computer) and the computation can be performed at any m/c. When a m/c needs to access a file stored on a remote m/c the remote m/c performs the necessary file access operations and returns data if a read operation is performed. However for higher performance several m/c referred to as file servers are dedicated to storing files and performing storage and retrieval operations. The rest of the m/c in the system can be used solely for computational purposes. These m/c are referred to as clients and they access the files stored on servers.

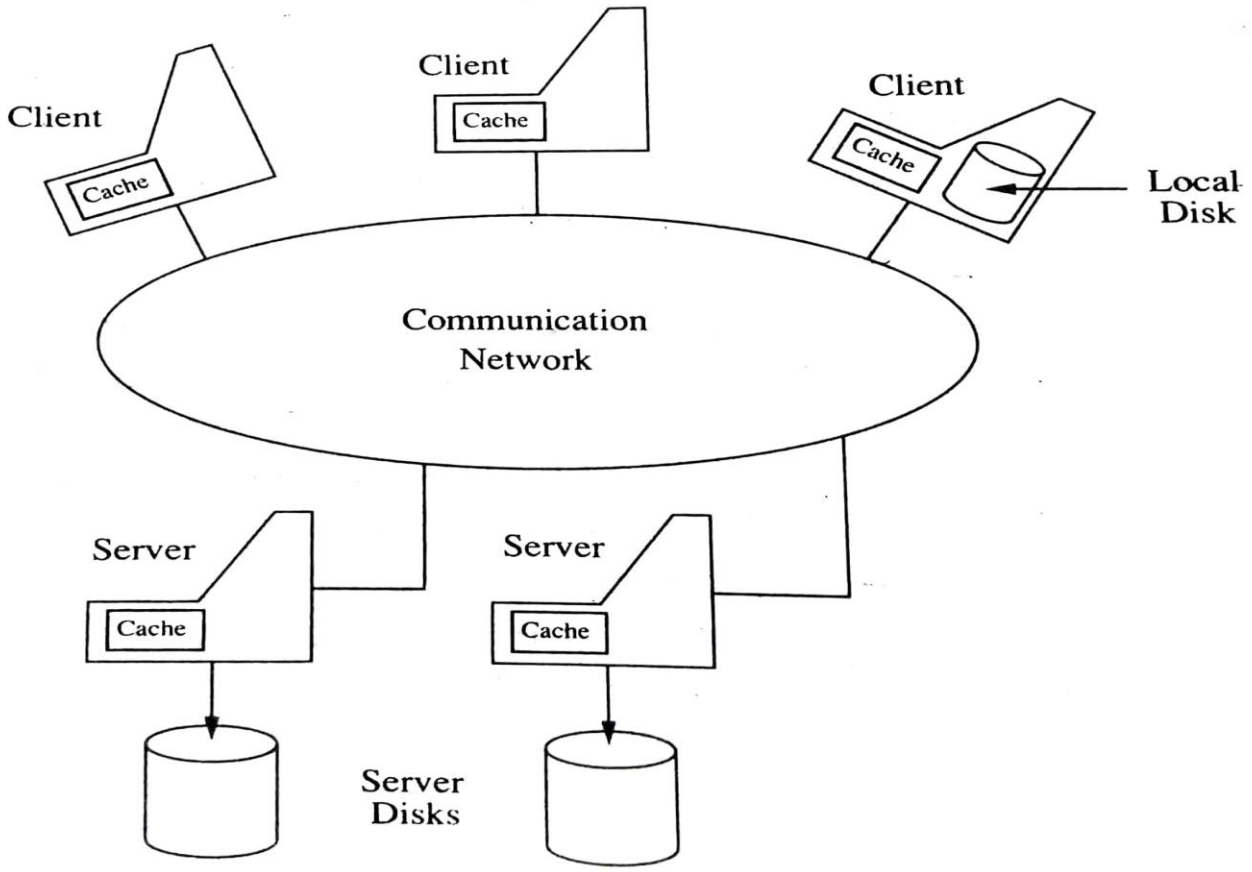

Fig1. Architecture of a Distributed File System

Contd.

The two most important services present in a DFS are the name server and cache manager. A name server is a process that maps names specified by clients to stored objects such as files and directories. The mapping(also referred to as name resolution) occurs when a process references a file or directory for the first time. A cache manager is a process that implements file caching. In file caching a copy of data stored at a remote file server is brought to the client's m/c when referenced by the client. Subsequent access to the data are performed locally at the client, thereby reducing the access delays due to network latency. Cache managers can be present at both clients and file servers. If multiple clients are allowed to cache a file and modify it the copies can become inconsistent. To avoid this inconsistency problem, cache managers at both servers and clients coordinate to perform data storage and retrieval operations. The data access in a DFS proceeds as shown in the next figure.

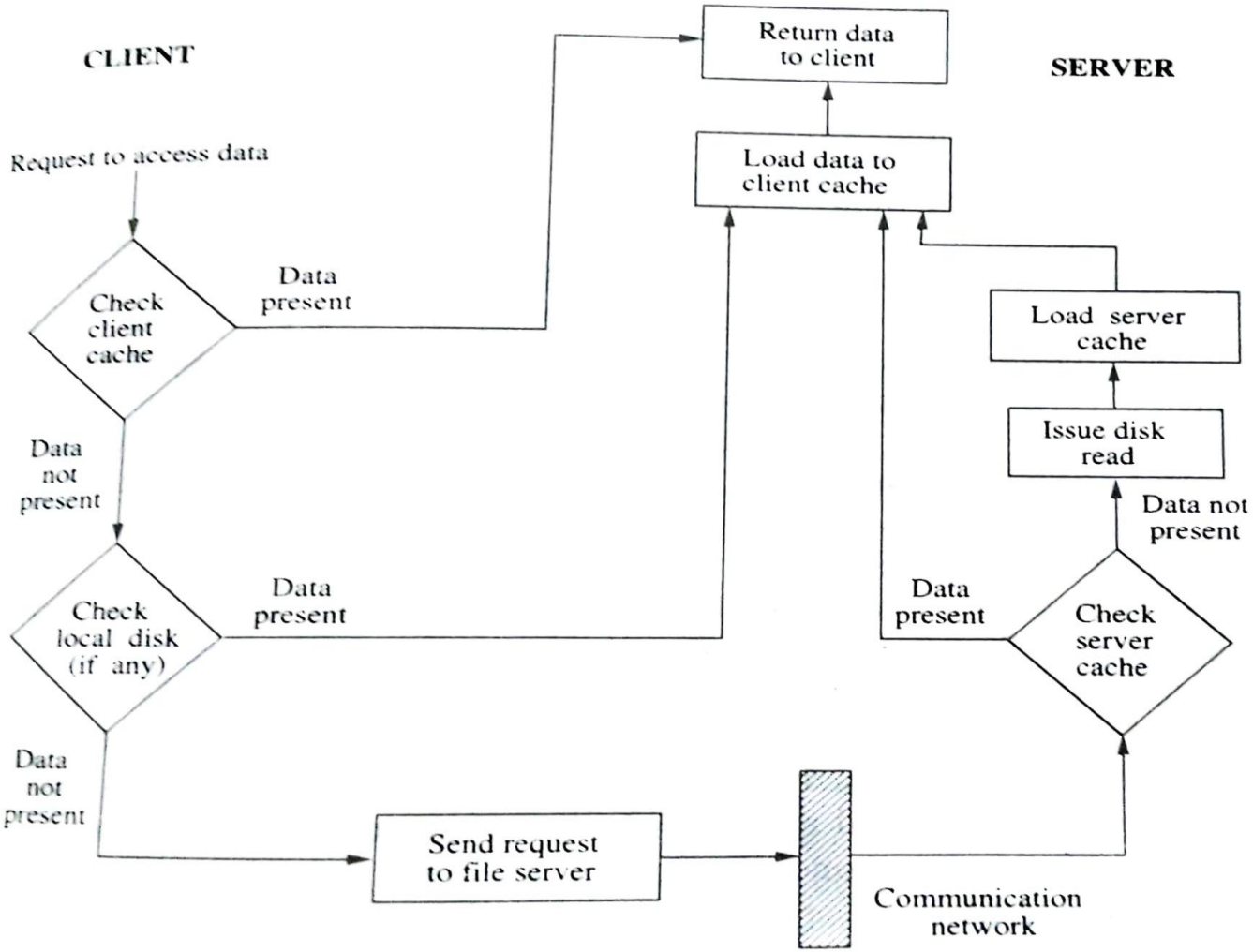

Fig2. Typical data access actions in DFS

Contd.

A request by a process to access a data block is presented to the local cache(client cache) of the m/c(client) on which the process is running. If the block is not in the cache then the local disk if present is checked for the presence of the data block. If the block is present then the request is satisfied and the block is loaded into the client cache. If the block is not stored locally then the request is passed on to the appropriate file server(as determined by the name server). The server checks its own cache for the presence of the data block before issuing a disk I/O request. The data block is transferred to the client cache in any case and loaded to the server cache if it was missing in the server cache.

Mechanisms for building DFS

- Mounting

This mechanism allows the binding together of different filename spaces to form a single hierarchically structured name space. Even though mounting is UNIX specific, it is worthwhile to study this mechanism as most of the existing DFS are based on UNIX. A name space (or collection of files) can be bounded to or mounted at an internal node or a leaf node of a name space tree.

Contd.

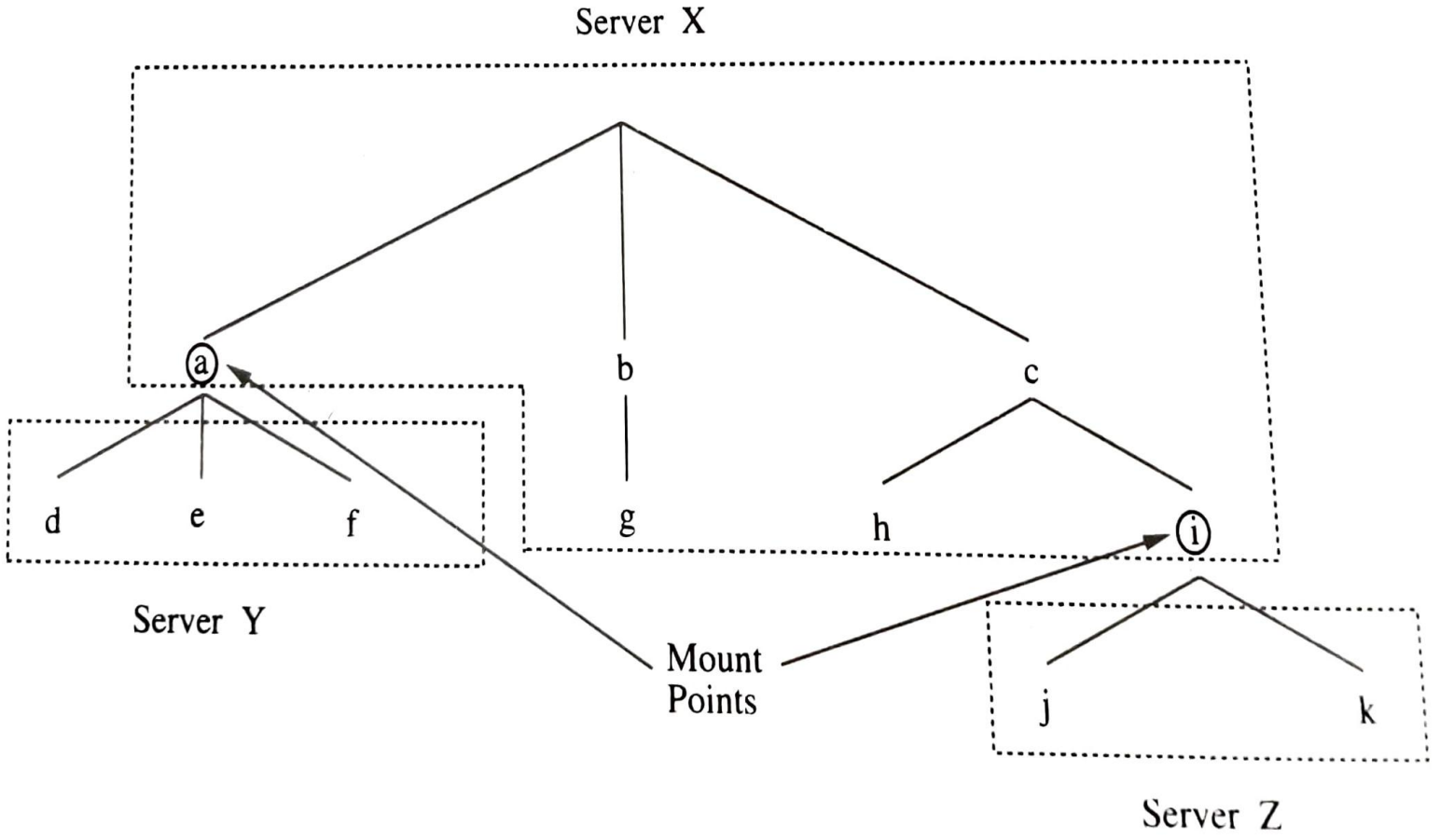

Fig3. Name Space Hierarchy

Contd.

A node onto which a name space is mounted is known as a mount point. In previous fig. nodes a and i are mount points at which directories stored at server Y and server Z are mounted respectively. a and i are internal nodes in the name space tree. The kernel maintains a structure called the mount table which maps mount points to appropriate storage devices.

In DFS file systems maintained by remote servers are mounted at the clients. There are two approaches to maintain the mount information

i) Mount information can be maintained at clients, in which case each client has to individually mount every required file system. This approach is employed in the Sun network file system.

Contd.

ii) Mount information can be maintained at servers in which case it is possible that every client sees an identical filename space. If files are moved to a different server then mount information need only be updated at the servers. In the first approach every client needs to update its mount table.

- **Caching**

It is used in DFS to reduce delays in the accessing of data. In file caching a copy of data stored at a remote file server is brought to the client when referenced by the client. Subsequent access to the data is performed locally at the client thereby reducing access delays due to network latency. Caching exploits the temporal locality of reference exhibited by programs. The temporal locality of reference refers to the fact that a file recently accessed is likely to be accessed again in the near future.

In addition caching reduces the frequency of access to the file servers and the communication network, thereby improving the scalability of a file system.

Contd.

- **Hints**
- **Bulk Data Transfer**
- **Encryption**

Design Issues

- **Naming and Name Resolution**
- **Caches on Disk or Main Memory**
- **Writing Policy**
- **Cache Consistency**
- **Availability**
- **Scalability**