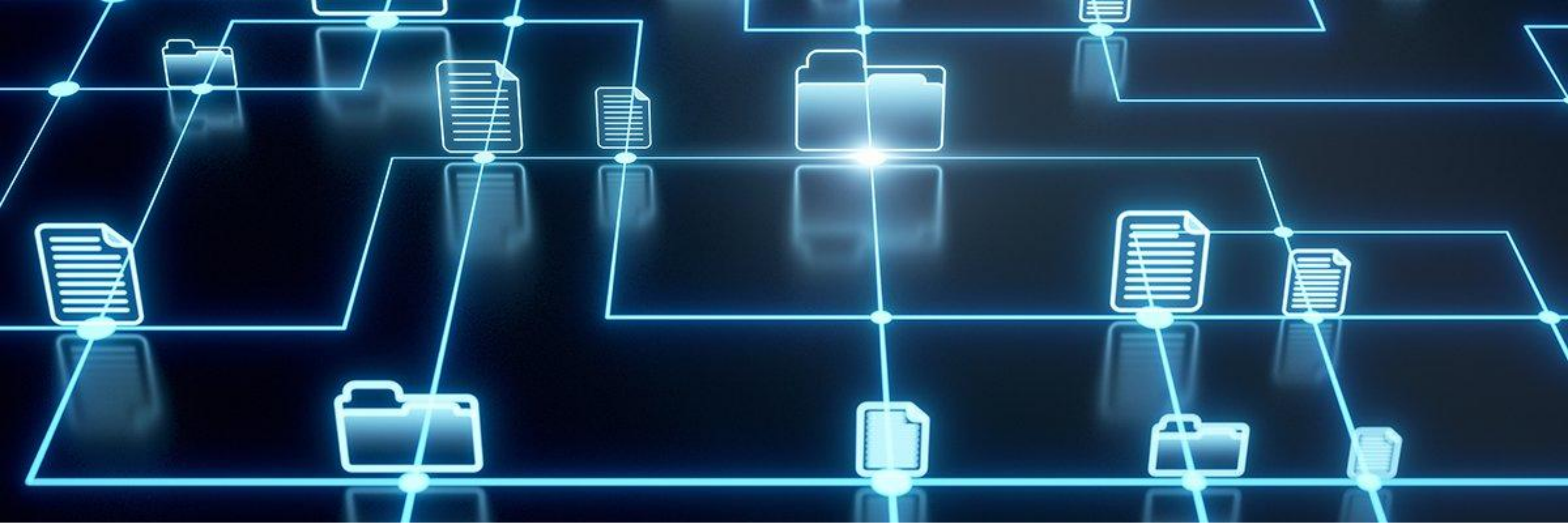


FILE SERVICE ARCHITECTURE

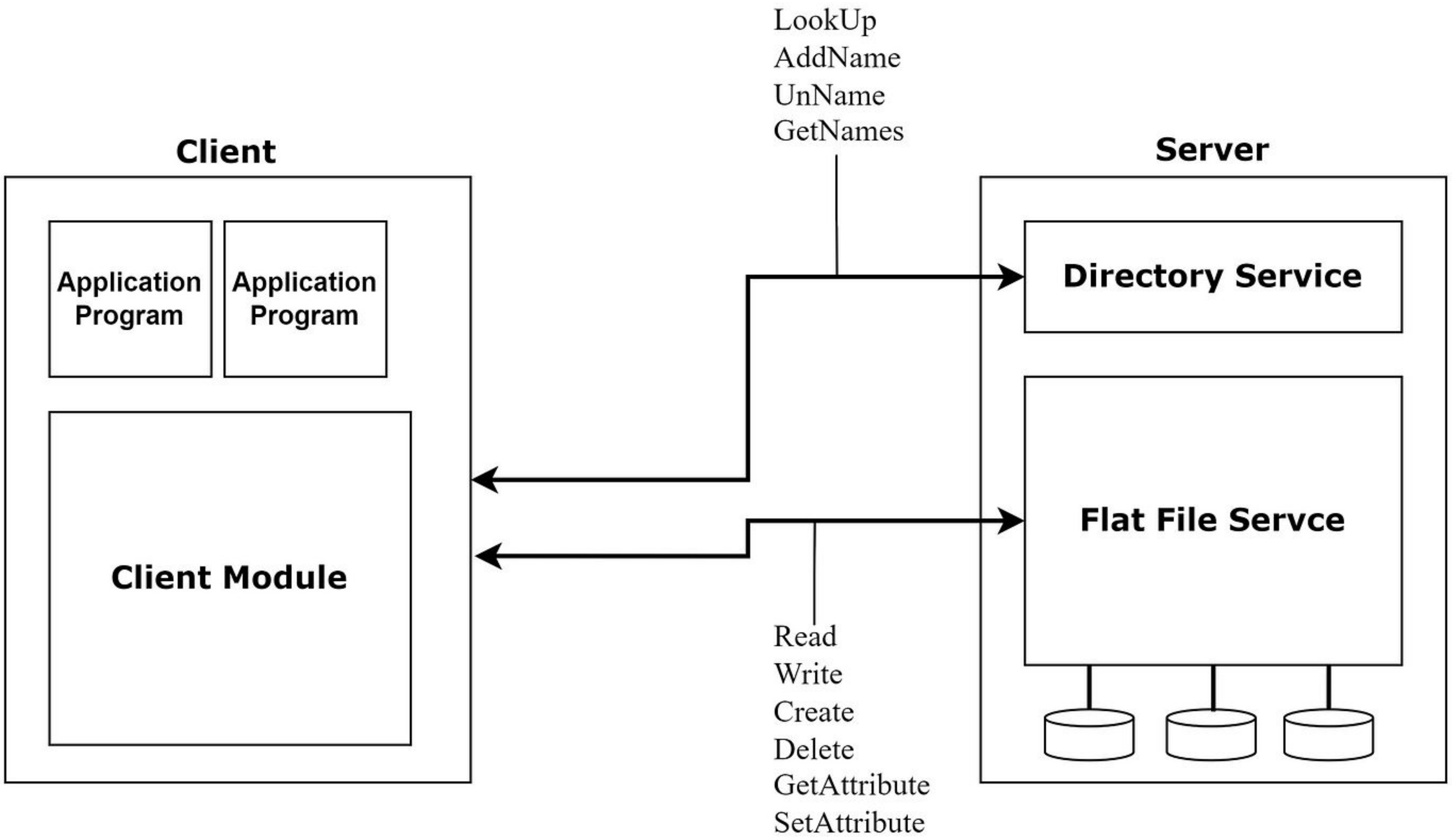
FILE SERVICE ARCHITECTURE

File Service Architecture is an architecture that provides the facility of file accessing by designing the file service as the following three components:

A client module

A flat file service

A directory service


LookUp
AddName
UnName
GetNames
Client
Server
Application Program
Application Program
Directory Service
Client Module
Flat File Servce
Read
Write
Create
Delete
GetAttribute
SetAttribute

1. **Client Module:**

   The client module executes on each computer and **delivers an integrated service (flat file and directory services) to application programs** with the help of a **single API.**

   It stores information about the network locations of flat files and directory server processes.

2. Flat file service:

- A flat file service is used to perform operations on the contents of a file.
- The Unique File Identifiers (UFIDs) are associated with each file in this service.
- For that long sequence of bits is used to uniquely identify each file among all of the available files in the distributed system.
- When a request is received by the Flat file service for the creation of a new file then it generates a new UFID and returns it to the requester.

Flat File Service Model Operations:

Read(FileId, i, n) -> Data: Reads up to n items from a file starting at item ‘i’ and returns it in Data.

Write(FileId, i, Data): Write a sequence of Data to a file, starting at item I and extending the file if necessary.

Create() -> FileId: Creates a new file with length 0 and assigns it a UFID.

Delete(FileId): The file is removed from the file store.

GetAttributes(FileId) -> Attr: Returns the attribute of file.

SetAttributes(FileId, Attr): Sets the attributes of the file.

Advantage of File System Architecture

Data Management and Organization:

- Hierarchical structure of directories and subdirectories.
- Naming conventions for unique file identification.

Data Access and Retrieval:

- Indexing methods for quick file location.
- Caching strategies to speed up data access.

Data Integrity and Reliability:

- Journaling for tracking uncommitted changes.
- Error detection and correction mechanisms.

Security and Access Control:

- Permissions for controlling read, write, and execute access.
- Support for encryption to protect data.

Space Management:

- Efficient allocation and deallocation of storage space.
- Enforcement of quotas to limit space usage by users/groups.