SECURE RPC FILE SYSTEM

Management of files and storage across nodes of a distributed system is achieved through a Distributed File System. But, as files are shared across the network, there is always the possibility of a third party accessing them without permission. Same goes for unprotected communication over the network.

We aim to design a secure distributed file system which will authenticate the nodes in the distributed system and encrypt the communications between node and file server which will take place using RPC.

Problem Description

Objective is to design a secure file system which will allow for nodes in a distributed system to remotely access files using RPC stored on a remote File Server. The File System should have the following functionalities:

* A common Key Distribution Centre(KDC) that provides facilities for registering and sharing symmetric keys of distributed nodes and file servers.
* File servers will register with the KDC for files they store currently.
* For distributed node to access a file on a FS, it should:
* Register itself at KDC which generates a session key and mutually authenticate with the file server using the key
* Users should be provided with a shell by the Distributed Nodes for accessing the file system. The following commands should be available:

**ls pwd cat cp**

Note: All communications will take place using RPC.

System Overview

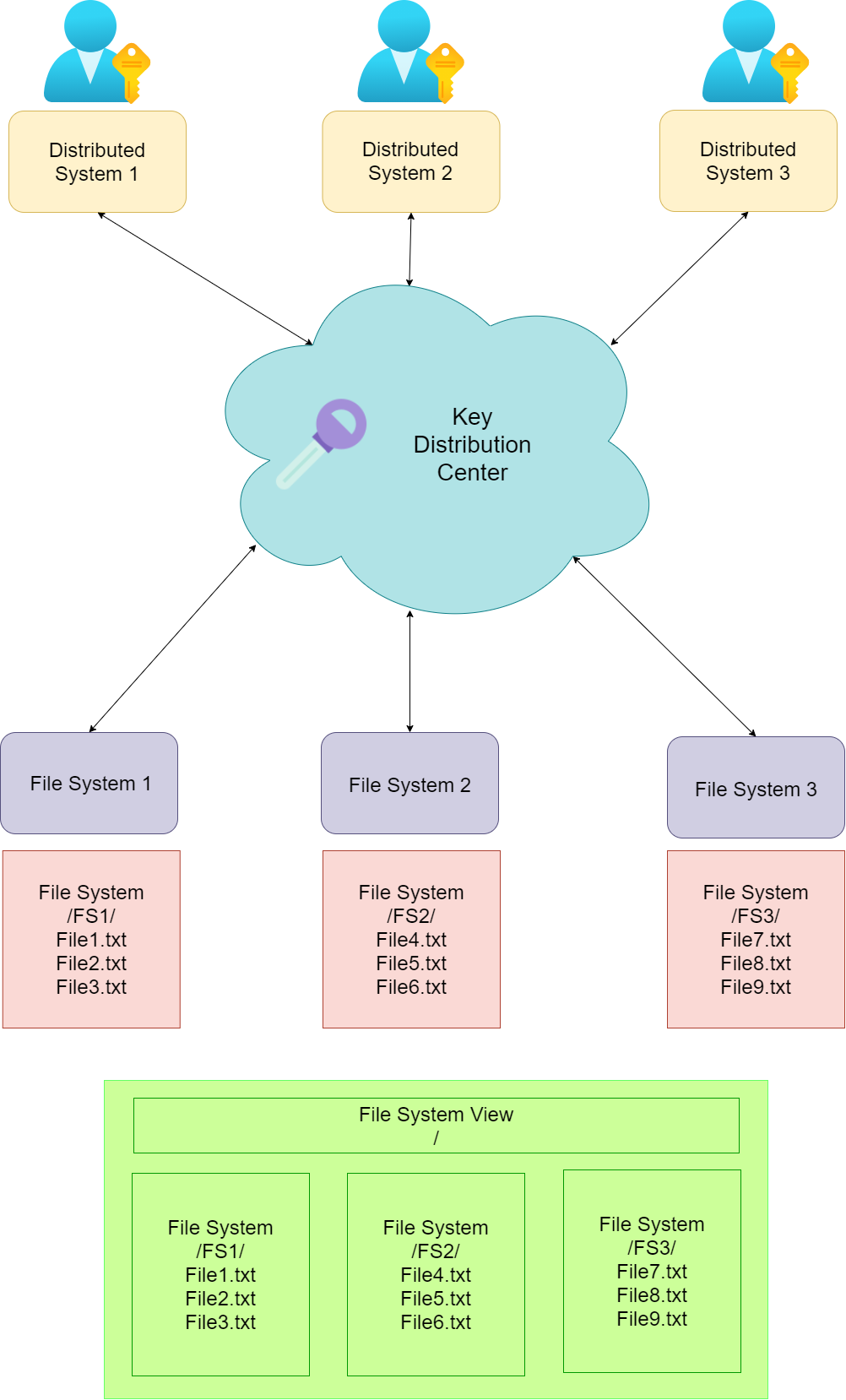
The system contains 3 key components:

File Server (FS): This server is the part of distribution that provides the storage functionality and file system. A distributed node can access this server after mutual authentication with FS.

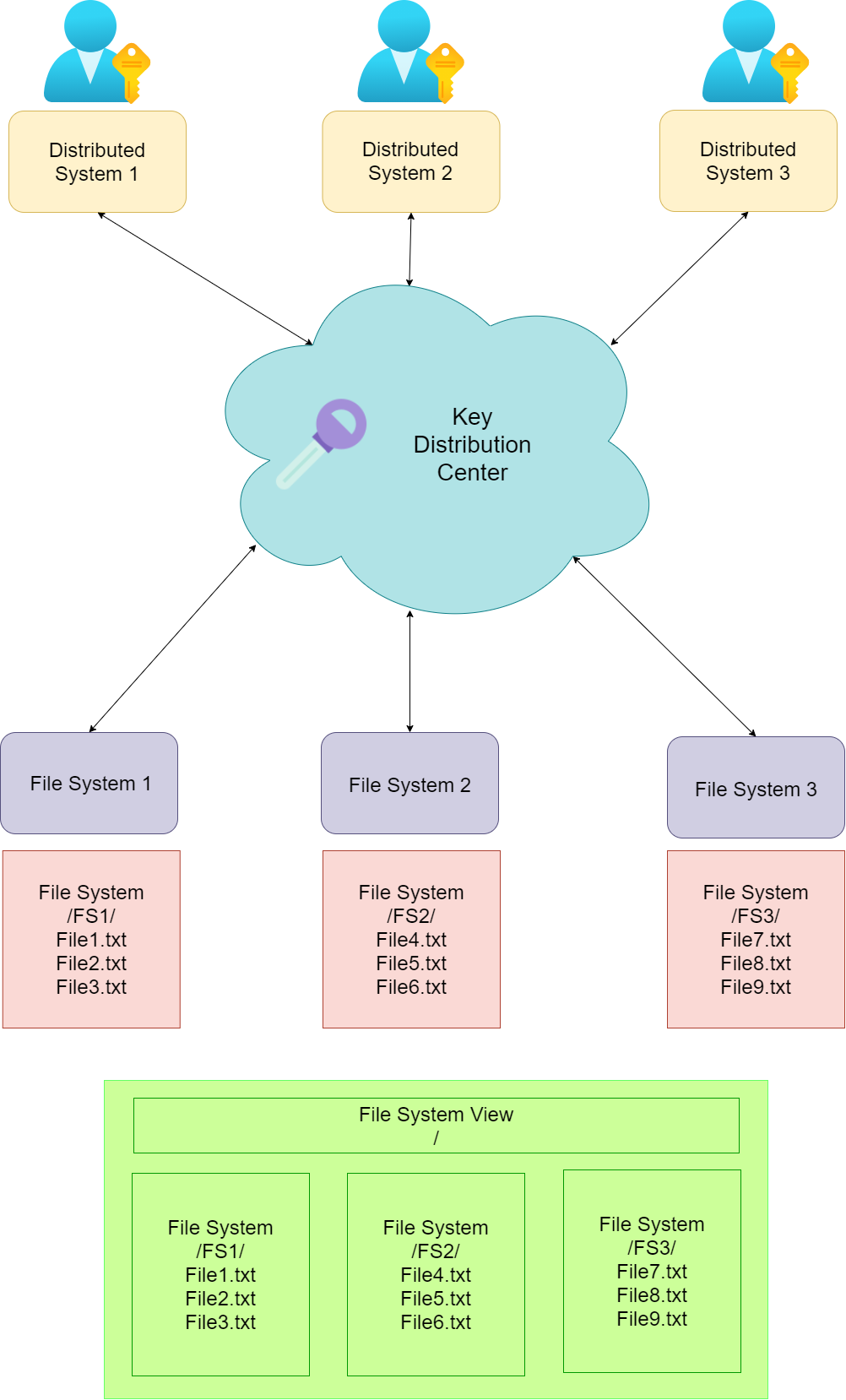
Distributed Node (DN): This is a node in the distributed system which gives the user access to the file system using various shell commands

Key DIstribution Center (KDC): This part of the system is responsible for registering the FS and Distributed node and distributing the symmetric key for mutual authentication among FS and Distributed node.

The block diagram below shows how the components interact with each other.

Fig. 1 Interaction of various components in the file system

Here, file servers FS1 , FS2 and FS3 register themselves with the KDC. A directory is created with ‘/’ at the root and FS1, FS2, FS3 as the children to root. The various files present in the file servers are at the leaf of the directory.

Fig. 2 Directory structure of File System

Functionalities of the System

Registration of a File Server: KDC registers file server using the unique id of FS and the symmetric key of FS is shared with KDC.

Registration of Distributed Node: KDC registers distributed node using the unique id of the node.

Accessing a FS by a Distributed Node: Node contacts KDC using its symmetric key and id of File Server. KDC generates the session key and sends it to Node. Node then mutually authenticates with FS using the session key.

Addition of a new fIle to the FS: When a new file is introduced to the FS, it sends updated directory to all the Distributed nodes.

Command Shell provided to the user: After the distributed node is authenticated with the FS, it is allowed to run terminal commands at the file system and view the results to the user.

Five commands will be available:

**ls:** list all the files present in the current directory.

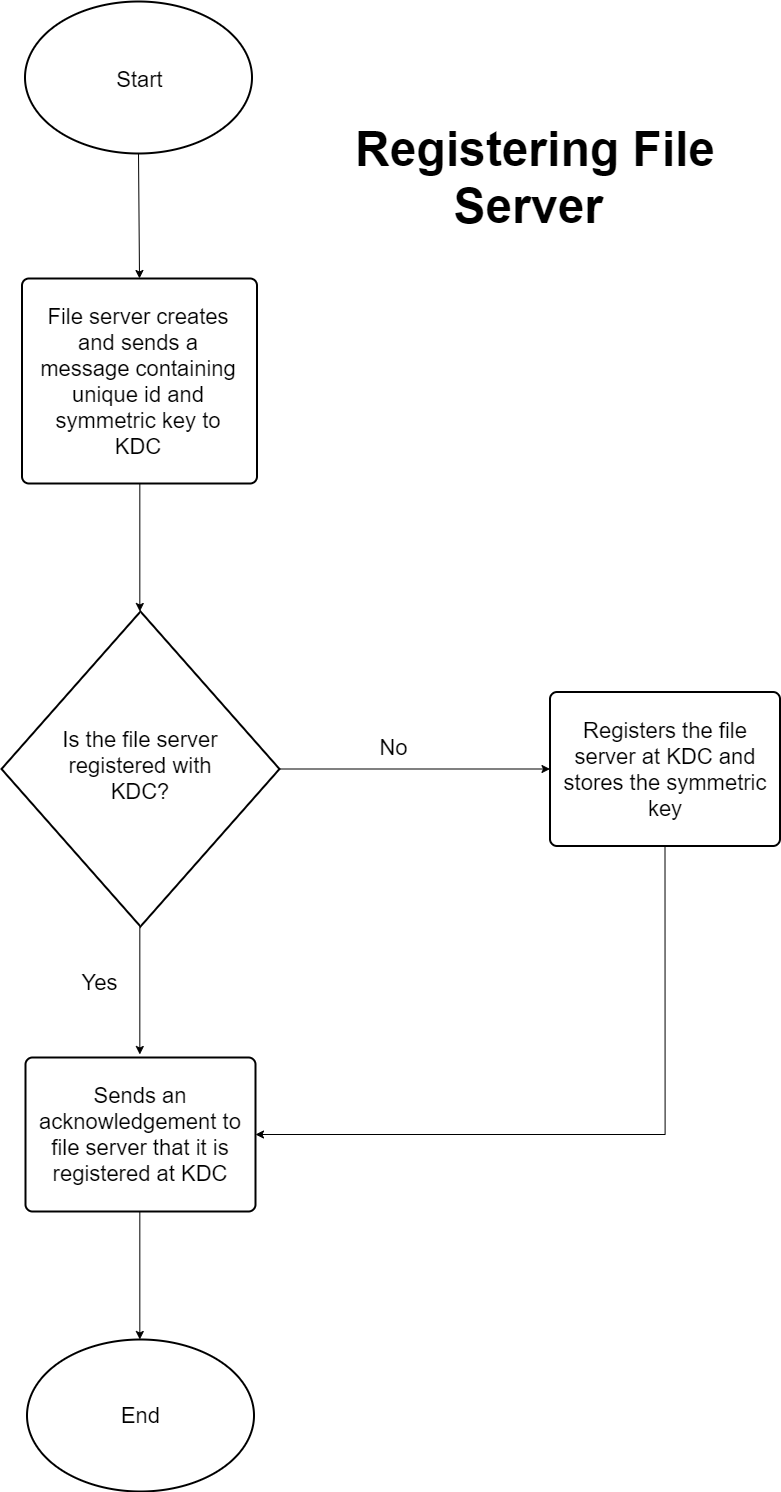
**pwd:** get the current working directory

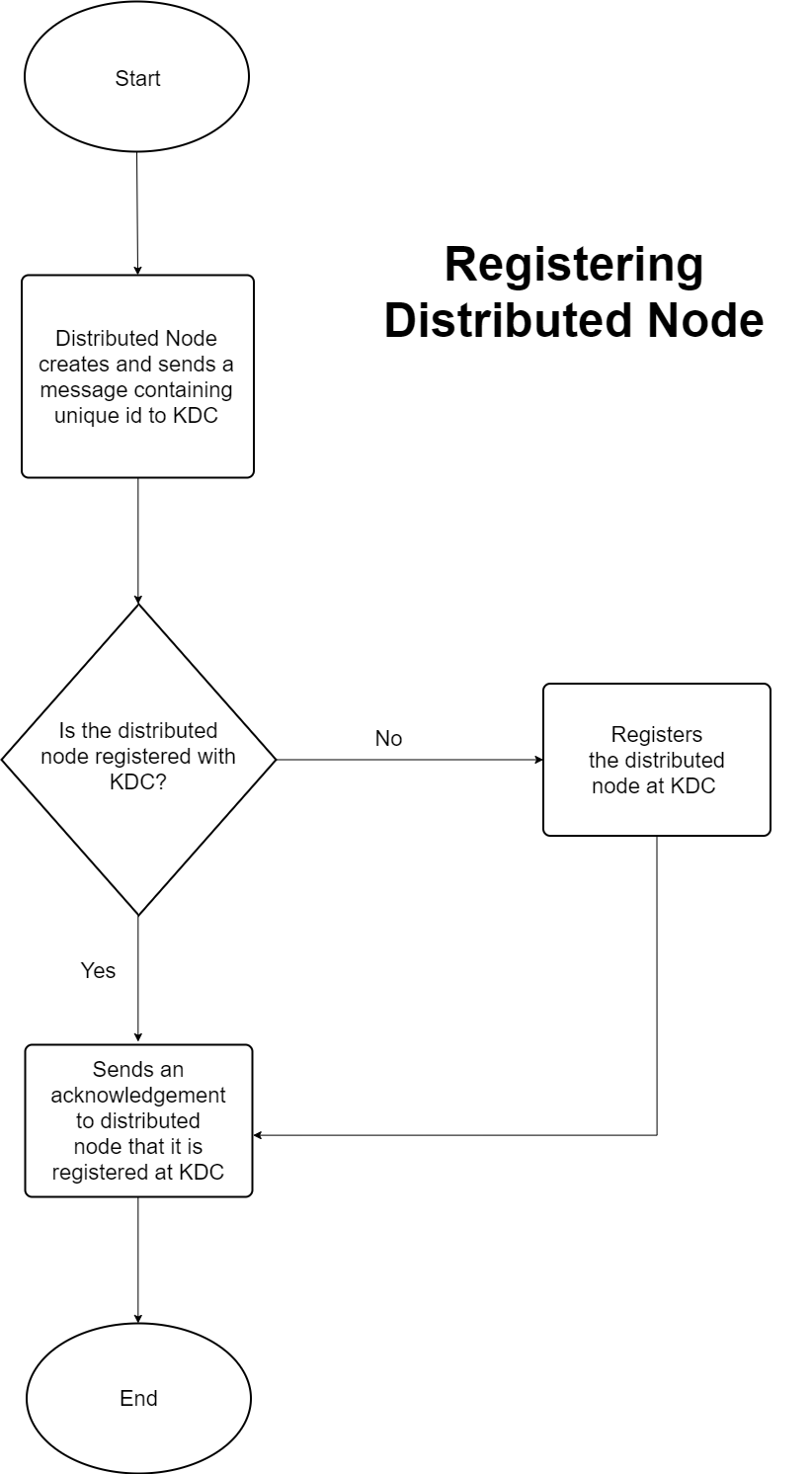
**cat file.xyz:** view the contents of the file named file.xyz

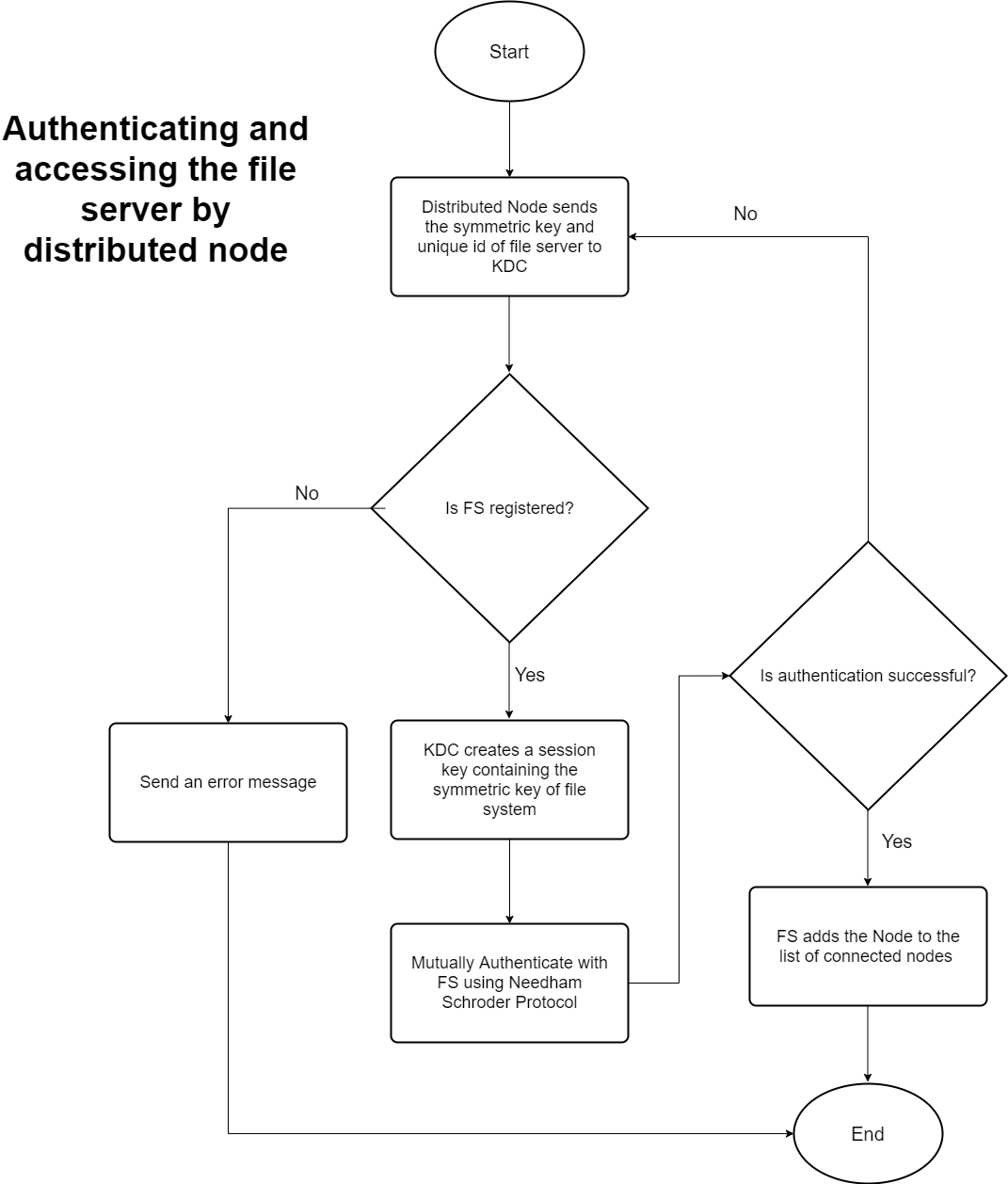
**cp file1.xyz file2.xyz:** Copy the contents of file1.xyz to file2.xyz

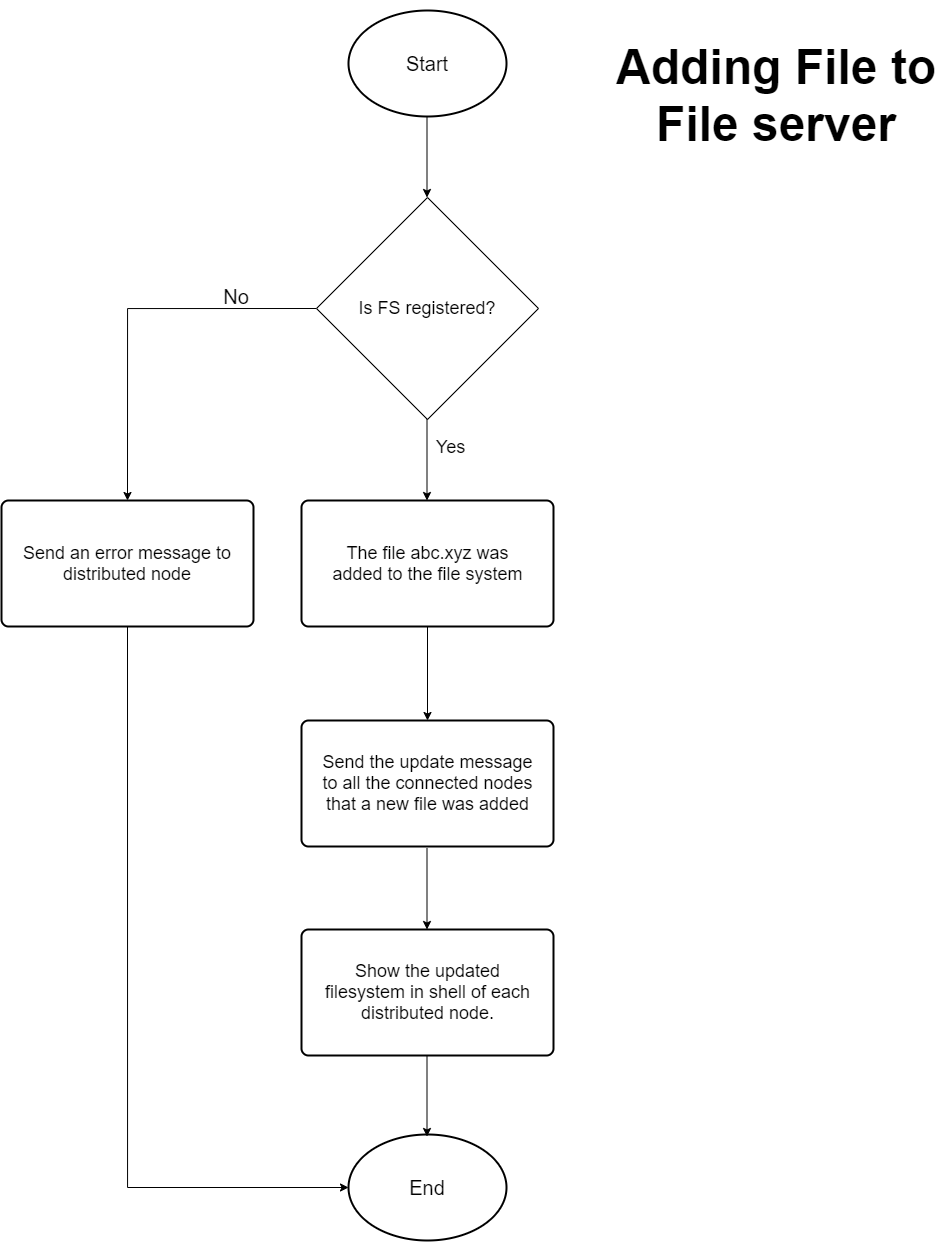
**end:** This is an extra command that we provide in case the user wants to end the current session.

The flow of various functionalities are shown below.

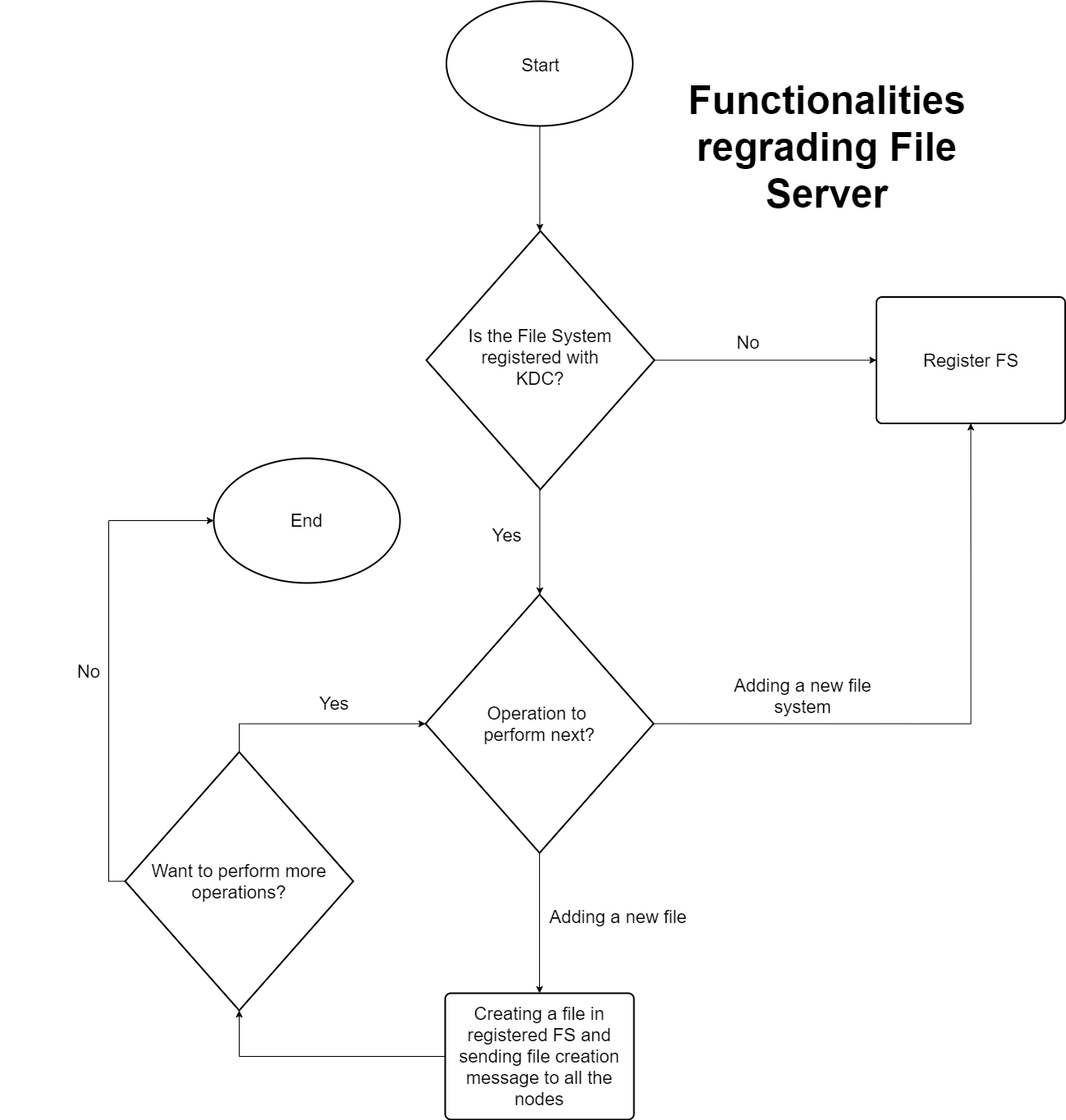


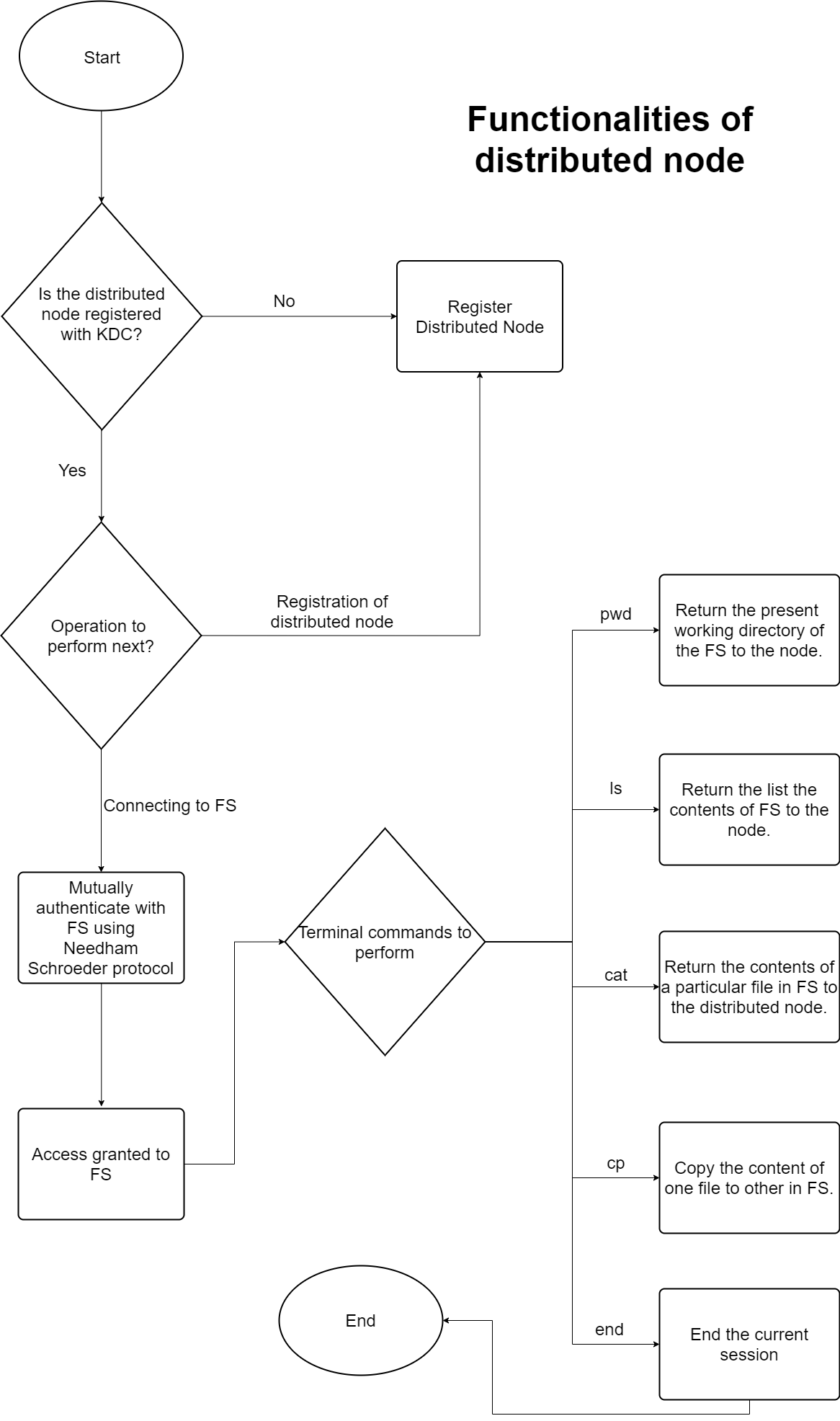




****

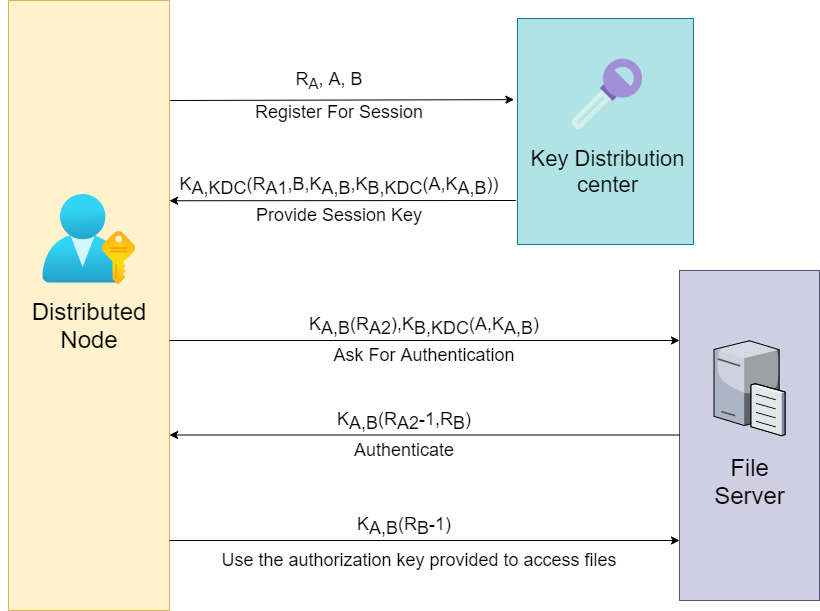
The next two flowcharts shows the overall flow of system from point of view of File Server and Distributed Nodes respectively





Protocols

For mutual authentication with FS : Needham Schroeder Protocol

Fig 3: Flow of Needham Schroeder Protocol

For Communications:Remote Procedure Calls

PLATFORM FOR IMPLEMENTATION

Operating System For Client and Server: Linux

Programming language: Python 3

Libraries:

For RPC: rpcpy

For File and console access: os

For deploying Server: Django

Submitted By**:**

Adarsh Jha (2K17/CO/22)

Aditya Choudhary(2K17/CO/24)