DELHI TECHNOLOGICAL UNIVERSITY

DISTRIBUTED SYSTEM
(CO - 407)
Innovative Project Design Document

Submitted By --

1. **Aditya Sharma (2k17/co/029)**
2. **Akhil Tomar (2k17/co/037)**

INTRODUCTION

A distributed system is a collection of autonomous computing elements that appears to its users as a single coherent system. To assist their development, each of the distributed systems are often organized to have a separate layer of software that is logically placed on top of the respective operating systems of the computers that are part of the system, called the middleware.

The project aims at providing a distributed file system that is scalable, transparent and location independent. The communication over the network (between client and server) in the distributed system takes place in the form of Remote Procedure Calls. RPCs involve server and client stubs which are responsible for performing the process of marshaling (for parameter specification) and have a predefined IDL based interface.

For a secured communication to take place on a non-secure network, a computer-network authentication protocol can be used. In this project, we'll be using the Needham–Schroeder Protocol. The Needham–Schroeder Protocol is based on a symmetric encryption algorithm and forms the basis for the Kerberos protocol. This protocol aims to establish a session key between two parties on a network, typically to protect further communication.

The current project involves File server and Distributed node registration through the security protocol and authentication with servers to mount files on a shell with regards to the invoked RPC.

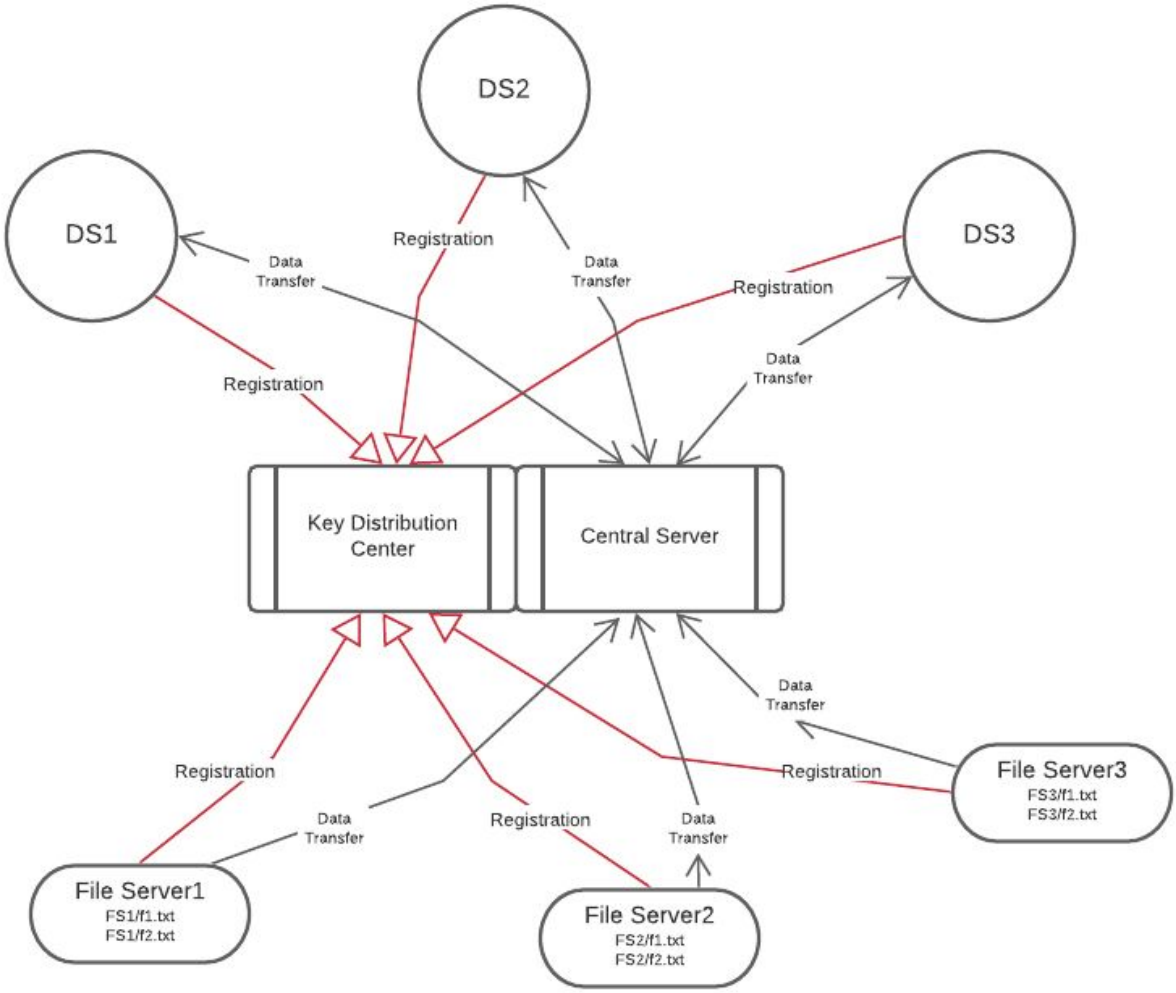
TECHNOLOGY STACK USED :

1. Fernet Module of Cryptography Package in python for the generation of symmetric keys and encryption/ decryption of Response/ Requests.
2. gRPC module as an RPC framework for generating a synchronous communication channel on the network while handling requests/ responses of the clients, file servers and central server.

The most important difference between traditional RPC frameworks and gRPC is that gRPC uses protocol buffers as the interface definition language for serialization and communication instead of JSON/XML. Protocol buffers can describe the structure of data and the code can be generated from that description for generating or parsing a stream of bytes that represents the structured data.

ARCHITECTURE

The whole project can be divided into 3 file types, each designed to perform a specific task.



This block diagram can be extended to N Client nodes and M File Servers & the ‘Key Distribution Center’ and ‘Central Server’ functionalities have been culminated to a single python script.

1. CENTRAL SERVER
   The Central server is a python script executed as a part of this Distributed File System Project to provide various methods and functionalities. Both the FS and Clients access it over a secured communication channel using gRPC and offers the following methods over RPC:
   a. Registration : Maintain dictionaries of shared_keys between multiple clients and FS.
   b. GiveFS : Provide a list of active servers to a client with which they can establish a session.
   c. GenKey : While establishing a connection with some FS, generate a session key for the same while following the procedure involved in Needham-Schroeder for authentication purposes.
   d. NewFile : When a new file is created by some client, the FS contacts CS through the NewFile RPC method. CS maintains the updates in the form of a dictionary.
   e. GetUpdate : Client contacts the CS after every task for any updates/notifications. CS responds with data stored in the previously discussed dictionary.

2. CLIENT NODE
   Multiple clients can be connected to the central server and sessions can be mounted with different file servers. The Client Node follows a predefined sequence of activities.
   a. Initiated with a registration to the CS
   b. Asking for the FS List from the CS
   c. Selecting an FS
   d. Authentication through Needham-Schroeder Protocol
   e. Selecting a task with respect to that FS
   f. New task selection or server change
   g. Exit if done

3. FILE SERVER
   The File Server contains methods to respond to requests made through an RPC from a client node. The FS undergoes registration with CS, directory build up, authentication for connection establishment and session maintenance. It offers the following RPCs:
   a) ShareKey : It forms the part of the Needham-Schroeder authentication service.
   b) LS : List the items of the server
   c) CP : Copy contents of one file to another
   d) CAT : Display contents of one file
   e) PWD : Display the current directory of File Server
   f) New : Create a new file in the FS

REGISTRATION

As a part of the deliverables, both the File Servers and Client Nodes need to be registered to the Central Server upon their creation. The process is elucidated in the following lines:

1. For the Client Node:

   The client node uses the Central Server's public key to send an Encrypted Registration request token to the latter. The request contains three strings:
   1. Shared Key : This key will be used for future communication between the central server and the client node.
   2. Random Nonce : To ensure the authenticity of the communication process. The Client sends Ra and expects a nonce Ra-1 from the recipient.
   3. PID : Process ID of the Client Node for identification purposes.

   CS:

```
adishar@LAPTOP-S5SH03UC:/mnt/c/Users/Asus/Documents/SEM VII/DIS/New folder$ python3 centralserver.py
Received a request for Registration by FS with Port 4000.
Sent a Response to the FS: `86298647`

Received a request for Registration by client with PID 1659.
Sent a response to the client: `6447267`

Received a request for FS list by client with PID 1659.
Received a request for Registration by client with PID 1666.
Sent a response to the client: `71658851`
```

   Client:

```
adishar@LAPTOP-S5SH03UC:/mnt/c/Users/Asus/Documents/SEM VII/DIS/New folder$ python3 client.py
The Client is functioning now.
Registration request sent to central server by client: `71658852`
Registration complete successfully.
```

2. For the File Server:

   The FS node uses the Central Server's public key to send an Encrypted Registration request token to the latter. The request contains three strings:
   1. Shared Key : This key will be used for future communication between the central server and the FS node.
   2. Random Nonce : To ensure the authenticity of the communication process. The FS sends Ra and expects a nonce Ra-1 from the recipient.
   3. PID : Process ID of the FS Node for identification purposes. (has the form FS###)

   CS:

```
adishar@LAPTOP-S5SH03UC:/mnt/c/Users/Asus/Documents/SEM VII/DIS/New folder$ python3 centralserver.py
Received a request for Registration by FS with Port 4000.
Sent a Response to the FS: `86298647`
```

FS:

```
adishar@LAPTOP-S5SH03UC:/mnt/c/Users/Asus/Documents/SEM VII/DIS/New folder$ python3 fileserver.py 4000 1
Directory updated !!
Server started at port  4000 .
Sent a Registration request to Central Server.`86298648`
Registration complete successfully.
File Server is now active:
```

Encryption is performed using the Cryptography Library's Fernet Module and is sent over a synchronous communication channel developed through the gRPC module similar to all other RPC calls made through this project.

The Central Server has a Registration method defined to handle all such related RPC requests within the Class Central. The Shared Key is stored in the form of a dictionary having the pid of client/FS as the 'key' and shared_key as the item. The method encrypts the response (nonce - 1) with this shared key and sends it over the synchronous channel to complete the RPC procedure.

AUTHENTICATION

The Authentication of File Server and Client Nodes prior to the session establishment follows the procedure as described in the Needham-Schroeder Symmetric Protocol.

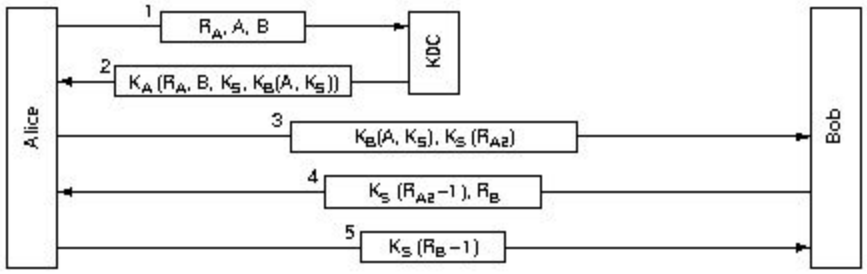


Fig. 7-18. The Needham-Schroeder authentication protocol.

The Needham–Schroeder Protocol is based on a symmetric encryption algorithm and forms the basis for the Kerberos protocol. This protocol aims to establish a session key between two parties on a network, typically to protect further communication.

STEP I : Sent a Request containing a random nonce, pid of client and pid of FS to CS.
STEP II : CS sends a response encrypted by the shared key of client and CS. The Response contains the original nonce, pid of FS, session key generated by CS and a ticket. The Tickets is a string encrypted by the shared key of FS and CS consisting of session key and pid of client.
STEP III : The Client sends a request to FS containing two messages -- the ticket and a Random nonce encrypted by the session key.
STEP IV : The FS uses the ticket to obtain the session key, decrypts the random nonce and Sends a response nonce-1 encrypted by the session key.
STEP V : The Client authenticates its identity by responding with the nonce sent by the FS.

Command Line Interface for the same -

Client :

```
adishar@LAPTOP-S5SH03UC:/mnt/c/Users/Asus/Documents/SEM VII/DIS/New folder$ python3 client.py
The Client is functioning now.
Registration request sent to central server by client: `24221464`
Registration complete successfully.
The Number of File Servers active are 1.
Select the File Server to establish communication:

1 ) FS4000
Index of selected FS:   1

Needham-Schroeder Authentication:


STEP1: Sent a request to the KDC for a ticket to establish session between the client with pid 1866 and File Server FS4000.
STEP2: Session Key and Ticket received.
STEP3: Sent RPC request to FS with an encrypted nonce `65271349` and Ticket.
STEP4: Received an Ecrypted nonce from FS `65271348`.
STEP5: Sent the Encrypted nonce to FS.
Session has been established between client with pid 1866 and file server FS4000.
```

CS :

```
adishar@LAPTOP-S5SH03UC:/mnt/c/Users/Asus/Documents/SEM VII/DIS/New folder$ python3 centralserver.py
Received a request for Registration by FS with Port 4000.
Sent a Response to the FS: `44858309`

Received a request for Registration by client with PID 1866.
Sent a response to the client: `24221463`

Received a request for FS list by client with PID 1866.

Received a Session Establishment Request from client with pid 1866 for FileServer FS4000.
Sent a Ticket and generated a Session-Key for the same.
```

FS :

```
adishar@LAPTOP-S5SH03UC:/mnt/c/Users/Asus/Documents/SEM VII/DIS/New folder$ python3 fileserver.py 4000 1
Directory updated !!
Server started at port  4000 .
Sent a Registration request to Central Server.`44858310`
Registration complete successfully.
File Server is now active:

1 )Received a session key from client pid 1866 with a nonce.
   Sent a nonce to the client for authentication.
   Connection has been established!
```

SERVICES

As stated earlier, the project encompasses a multitude of services. The client selects a service as per his/her whims and after establishing a session with an FS, a request encrypted by the session key is sent to the latter. The FS works on the request and provides a response encrypted by the same session-key, all happening through a synchronous RPC channel.

Most of these services are described as follows:

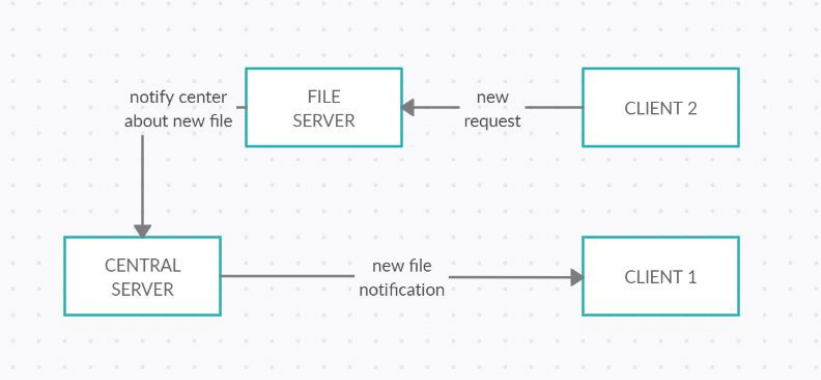
1. Server List:
   Request: The pid of the client sent to the CS.
   Response: A list of File Servers active with their respective ports to establish connection.
2. Ls
   Request: The pid of the client .
   Response: A list of files present in the FS.

3. Cat
   Request: The pid of the client and the filename whose content needs to be displayed.
   Response: The content of the file as read from the latter.

4. Cp
   Request: Two files, first files content is copied into the second file.
   Response: The updated content of the second file as read from the latter.

5. Pwd
   Request: The pid of the client .
   Response: The current directory of the FS.

6. new
   Request: The pid of the client and the name of the new file in the FS.
   Response:
   A. The content of the new file in the FS.
   B. An RPC request to CS notifying about the update.
   C. The CS is responsible to forward this notification to all the clients which have been registered with the same through another RPC communication Request.



In the case of Cp and Cat tasks, if the file does not exist then an error message would be displayed on the terminal. Similarly, if the entered task is not a valid task, the user will continue to be prompted unless added.

Demo for Services :

1. Client:
   We take two clients in this demo, C1 and C2.
   C1 & C2 follow the following sequence of events.

    a. Registration with CS
    b. RPC to get Active File Servers
    c. Selection of File Server
    d. Authentication of FS using NS protocol through the CS.
    e. Selection of task

```
adishar@LAPTOP-S5SH03UC:/mnt/c/Users/Asus/Documents/SEM VII/DIS/New folder$ python3 client.py
The Client is functioning now.
Registration request sent to central server by client: `94541996`
Registration complete successfully.
The Number of File Servers active are 2.
Select the File Server to establish communication:

1 ) FS4000
2 ) FS4001
Index of selected FS:   2

Sent a request to the KDC for a ticket to establish session between the client with pid 1774 and File Server FS4001.
Session Key and Ticket received.
Sent RPC request to FS with an encrypted nonce and Ticket
Session has been established between client with pid 1774 and file server FS4001.


The File Server selected is FS4001. Please select the RPC service for the same !

1. ls - list files in given FS
2. cp - copy content of one file to another
3. cat - Display contents of given file
4. pwd - show current directory
5. new - add a new file
```

Task that were selected:

a. Ls :

```
ls

The Files present in FS4000 are as follows:
1 ) 0.txt
2 ) 1.txt
3 ) 2.txt
4 ) 3.txt
5 ) 4.txt
6 ) 5.txt
7 ) 6.txt
```

b. Pwd :

```
Select another service. (N for none)  :  pwd

The directory is as follows:  /mnt/c/Users/Asus/Documents/SEM VII/DIS/New folder/FS4000
```

c. Cat :

```
Select another service. (N for none)  :  cat
Enter the filename:  1.txt

The contents of the given file are:

This is the first line in the text file - /mnt/c/Users/Asus/Documents/SEM VII/DIS/New folder/FS4000/1.txt
```

d. Cp :

```
Select another service. (N for none)  :  cp
Enter the two file names:
First File : 2.txt
Second File : 3.txt

The Contents of file 2.txt are copied into the file 3.txt.

The Request has been processed. The new content of 3.txt is :


This is the first line in the text file - /mnt/c/Users/Asus/Documents/SEM VII/DIS/New folder/FS4000/1.txt
This is the first line in the text file - /mnt/c/Users/Asus/Documents/SEM VII/DIS/New folder/FS4000/2.txt
This is the first line in the text file - /mnt/c/Users/Asus/Documents/SEM VII/DIS/New folder/FS4000/2.txt
```

e. New :

At C2 -

```
new
Please Enter the File name: 3.txt

The Contents of the new file are:

 File has been created by client with PID 1774
```

At C1 -

```
The following files have been added : FileName[FileServer]

   1 )  3.txt[FS4001]

Select another service. (N for none)  :
```

2. FS:
   We take two Servers (FS4000 and FS4001) in this demo, connected to C1 and C2 respectively.

   FS4000:

```
adishar@LAPTOP-S5SH03UC:/mnt/c/Users/Asus/Documents/SEM VII/DIS/New folder$ python3 fileserver.py 4000 1
Directory updated !!
Server started at port  4000 .
Sent a Registration request to Central Server.`29156739`
Registration complete successfully.
File Server is now active:

1 )Received a session key from client pid 1766 with a nonce.
   Sent a nonce to the client for authentication.
   Connection has been established!

   1) LS request from Client with PID 1766.
   2) PWD request from Client with PID 1766.
   3) CAT request from Client with PID 1766.
   4) CP request from Client with PID 1766.
   5) PWD request from Client with PID 1766.
```

   FS4001:

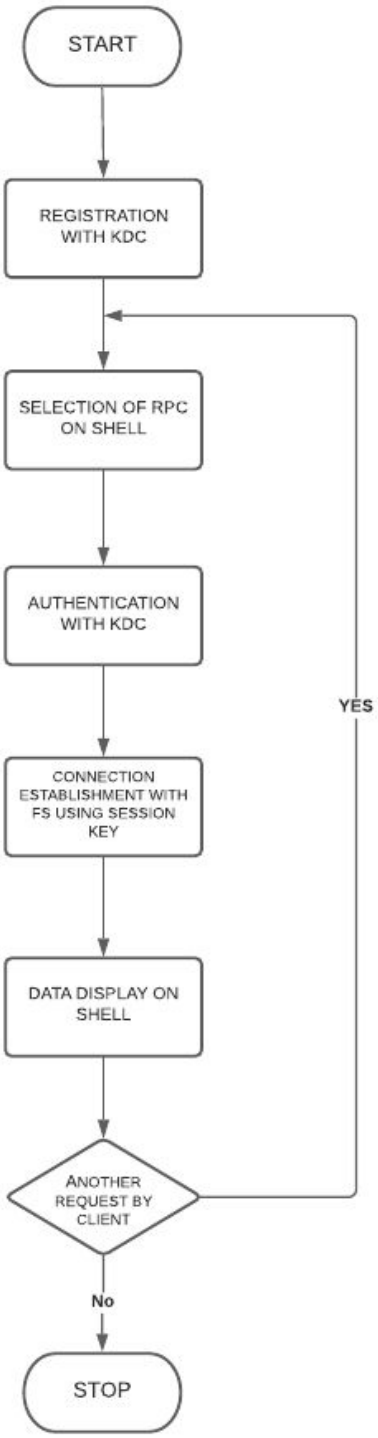
```
adishar@LAPTOP-S5SH03UC:/mnt/c/Users/Asus/Documents/SEM VII/DIS/New folder$ python3 fileserver.py 4001 1
Directory updated !!
Server started at port  4001 .
Sent a Registration request to Central Server.`00668249`
Registration complete successfully.
File Server is now active:

1 )Received a session key from client pid 1774 with a nonce.
   Sent a nonce to the client for authentication.
   Connection has been established!

   1) NEW request from Client with PID 1774. Submitted the update to Central Server.
```

3. CS:
   Single central server script is executed. The Output is as follows:

```
adishar@LAPTOP-S5SH03UC:/mnt/c/Users/Asus/Documents/SEM VII/DIS/New folder$ python3 centralserver.py
Received a request for Registration by FS with Port 4000.
Sent a Response to the FS: `29156738`

Received a request for Registration by FS with Port 4001.
Sent a Response to the FS: `668248`

Received a request for Registration by client with PID 1766.
Sent a response to the client: `44737629`

Received a request for FS list by client with PID 1766.
Received a request for Registration by client with PID 1774.
Sent a response to the client: `94541995`

Received a request for FS list by client with PID 1774.

Received a Session Establishment Request from client with pid 1766 for FileServer FS4000.
Sent a Ticket and generated a Session-Key for the same.


Received a Session Establishment Request from client with pid 1774 for FileServer FS4001.
Sent a Ticket and generated a Session-Key for the same.

New File 3.txt has been created in FileServer FS4001 by client with pid 1774.
Sent a Notification to 1766 about new files created.
```

FLOW DIAGRAM

1. DISTRIBUTED NODE FLOW



2. FILE SERVER FLOW