Distributed File System

Distributed File System is implemented in Python using RESTful Webservices. Web.py is used to implement RESTful webservices in python. NFS File System style is taken.

Four main components are implemented in DFS. The following are them:

1. Distributed Transparent File Access
2. Directory Service
3. Security Service
4. Lock Service

Distributed Transparent File Access

The following are the tasks that is taken care by distributed file server:

1. Makes a call to the Authentication server for getting the server encryption key for decrypting the ticket which in turn be used for decrypting and encrypting the messages sent by and send to client proxy.
2. Encrypts the messages received from client proxy and decrypts the message that are send to client proxy.
3. Carry out the read and write operations

Directory Server

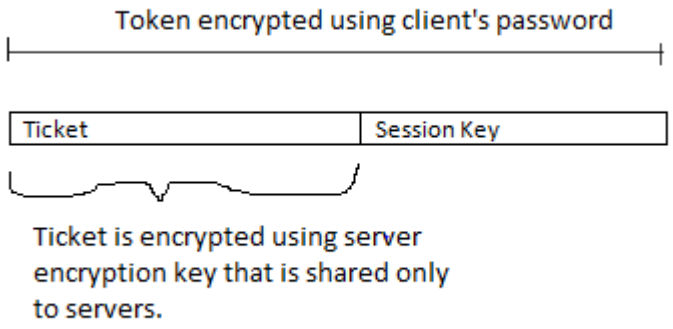
The following are the tasks that are done by the Directory Server:

1. Makes a call to the Authentication server for getting the server encryption key for decrypting the ticket which in turn be used for decrypting and encrypting the messages sent by and send to client proxy.
2. Encrypts the messages received from client proxy and decrypts the message that are send to client proxy.
3. A persistent dictionary is maintained at the Directory Server side in which filename, absolute file path and the port number in which the file server (in which that particular filename resides) runs.
4. Receives the filename from the client proxy , retrieves absolute file path and port number of the file server associated with that filename from the persistent dictionary.
5. Return the encrypted file path and port of the fileserver to the client proxy.

Security Service

As a first step, client is asked to login to the system. The username and password is authenticated by the Authentication Server. Authentication server maintains a persistent dictionary at its side in which the usernames and corresponding passwords are saved. After the authentication of the client, server sends back the token that is encrypted using the client's password.

Token contains two parts ticket and a session key that is encrypted using the server encryption key. This server encryption key is only shared to other server.



Ticket + session key is <u>token</u> that is encrypted using client's password.

After receiving this token, client decrypts it with its password to get the session key. This session key is used by client to encrypt the messages that it sends to servers and also to decrypt the messages it receives from the servers.

Client sends this encrypted message along with the ticket to the servers.

At the server side, server approaches authentication server for server authentication key. Ths key is used to decrypt ticket. The decrypted ticket is used to decrypt the message received from client and encrypt the message to be send to client.

Lock Server

The following are the tasks of the lock server:

1. Receives the encrypted filename from the client proxy.
2. Makes a call to the Authentication server for getting the server encryption key for decrypting the ticket which in turn be used for decrypting and encrypting the messages sent by and send to client proxy.
3. Encrypt the messages to be send to client and decrypt the messages received from client.
4. Maintains a persistent dictionary that stores filenames and the corresponding lock state (lock (1)/unlock(0))
5. Checks whether the requested file is locked or not.
6. Does the locking of file (whenever a write method) when triggered.
7. Unlock the file when triggered (after write operation is completed).

Client Proxy

A client program has been created that includes a client proxy that calls each of servers via rest calls.

The client takes the input from the user and passes the received data to the client proxy. The user is not aware about the client proxy that runs in the background.

Client proxy is responsible for making the rest call to each of the servers which are up in the order.

The following are the steps that are taken care by the client proxy:

1. Client proxy receives the username and password and make a rest call to the Authentication Server for the authentication. Client proxy also receives the security keys from the authentication server. (Process is explained in the Security Service)
2. Client proxy encrypts all the messages that it sends.
3. Client proxy also decrypts all the messages received from servers.
4. Makes a call to the Directory Server by passing the filename and receives the absolute file path along with the fileserver address (port number of the fileserver).
5. Makes appropriate calls to the lock server for locking and unlocking the files.
6. Makes calls to the file server for read and write operations.

The following is the flow of the read and write operations:

If the operation requested by user is read:

Login by calling Authentication server -> receives security keys from Authentication server -> Accepts filename and the method as “read” -> encrypts the filename and call the directory server for the absolute filepath and file server port in which the file resides -> Receives port in which the file server is running and the absolute file path as encrypted message -> Decrypt the message for getting port and absolute file path -> Encrypt the file path and makes a call to the file server by passing this encrypted file path to read the content in the file -> Receives the encrypted file content from file server -> decrypts the content and displays it.

If the operation requested by user is write:

Login by calling Authentication server -> receives security keys from Authentication server -> Accepts filename, the method as “write” and the content to be written into the file -> encrypts the filename and call the directory server for the absolute filepath and file server port in which the file resides -> Receives port in which the file server is running and the absolute file path as encrypted message -> Decrypt the message for getting port and absolute file path -> Calls the lock server by passing the encrypted filename -> Receives the encrypted message from the lock server-> Decrypt the message from lock server -> Executes case1 or case 2 based on the response from the lock server.

Case 1 – file is not locked

If the file is not locked, make another call to the lock server to lock the file by passing the encrypted filename and receive the response from lock server -> encrypts the file path and content to be written and makes a call to the file server by passing this encrypted message to write the content in

the file -> Receives the encrypted success message from file server -> decrypts the success message and displays it -> Unlock the file by calling lock service.

Case 2 – file is locked

If the file is locked, display that the write operation is not possible and exits.