User Manual

Intro

Our software functions as a dropbox where users can store files that will be shared with other users that share the same group. The software allows users to upload and download files, and all grouping will be managed by a root user. The system also features continuous authentication, which allows administrators to strictly enforce permissions that will take place immediately.

Running the Program + Setup

In order to run the program, the root user must run at least one of each: Authentication Server, Resource Server, and Client. Upon running each server, the user will be prompted for a port number that the Client will connect on. Upon running the client, the user will be prompted for a server name and port for each of the servers to connect to. The user will then have the option to login. Initially, the only user in the system will be the root, which can be logged into by using the following username-password pair: (root, root). From here, the root user can create new users and groups that can then log into the system.

Users

Each user is represented by a unique username and password that is at least eight characters in length. Each user can belong to only one group at a time, although they can be reassigned to a different group by the root user.

Groups

Groups are used to organize users and files and to limit the files that users can see. Groups are identified by a unique string. Any number of groups can exist in the system, and any number of users can exist in each group. When a user belongs to a given group, they can view, upload, and download all files that belong to that group. Users will not be able to view files that do not belong to the same group as them, nor interact with groups other than their own.

Resource Servers

The Client will have the option to connect to different RSs given they exist. There is no benefit to connecting to a specific RS over another, but users should choose one to use consistently because files are not synced across RSs.

Root User + Privileges

The root user is the only user able to create or delete other users. In order to sign in as the root user, the user should enter username: “root” and password: “root”. Once authenticated, the user will be able to run commands to manage users and groups. Attempting to run root commands while not signed in as the root user will result in a “command not found” error.

$ *create* <username> <password> <group_number>

This command is used to create a user in the system. The root must specify the username, password, and group that the new user will be assigned to. Usernames must be unique, and the root will receive an error message if they attempt to create a duplicate user. Once a user is created, they will exist in the system until they are deleted, and will belong to their assigned group until they are reassigned or deleted.

$ *delete* <username>

This command is used to delete a user in the system. The root must specify the username of the user to be deleted. The root will receive an error message if they attempt to delete a user that does not exist in the system. Deleting a user will remove all information about them and cannot be undone.

$ *collect* <group_name>

This command is used to create a group in the system. The root will receive an error message if the group already exists.

$ *release* <group_name>

This command is used to remove a group from the system. The root will receive an error message if the group does not exist or if any users currently belong to the group.

$ *assign* <username> <group_name>

This command is used to assign a user to a new group. The root must specify the username of the user to be reassigned and the number of the new group.

$ *list* <group_name>

This command is used to display all members of the given group. The output will be a list of usernames that belong to the users of the group.

$ *groups*

This command is used to display all groups in the system. The output will be a list of Strings that represent the names of each group.

$ *logout*

See *User Commands/logout*.

Sign-in/Authentication

Users will be prompted to enter a username and password by the client when starting the program. If either the username or password is incorrect, an error message will display that informs the user that the credentials they entered are invalid. Upon successful login, the user will be notified of the group they are assigned to and then given control of a command prompt to enter commands. Users can also log out of the system, which will return to the username prompt.

User Commands

Once logged into the system, users can run commands to manage the files they have access to:

$ *list*

This command is used to list all files that are visible to a user. The output will be a list of filenames that belong to the same group as the user.

$ *upload* <filename>

This command is used to upload a file to share with the user’s group on the resource server.

$ *download* <filename>

This command is used to download a file from the user’s group on the resource server.

$ *delete* <filename>

This command is used to delete a file from the user’s group on the resource server.

$ *logout*

This command will log the user out of the system.

Technician's Guide

The Authentication Server

The authentication server's purpose is to authenticate users by accepting login credentials and returning a token for access to a resource server. It accomplishes this by storing a list of users and a list of groups that it will reference each time it attempts to authenticate a user.

When a user attempts to login, the AS will consult the user list to determine if the user is present in the system and to verify the password that they provided is correct. If both fields match, the AS will create a new token to be sent back to the Client.

The AS is also responsible for handling all root commands. When any root command is sent, the AS will first confirm that the root user is logged in. Given that this is true, the root will be able to run commands to create and delete users and groups.

The Resource Server

The resource server's purpose is to store files that users will manage. It accomplishes this by manipulating the file system of the server that is currently running on. It interacts with the client and responds to file-related commands that deal with accessing the resources such as uploading, downloading, and deleting files.

The Client

The purpose of the Client is to provide a command-line interface to users to interact with the RS directly and AS indirectly to access their files.

The Client will connect to one AS and one RS upon starting, which it will stay connected to until exiting. The server name and port of each server to connect to can be inputted by the user when first running the Client.

Tokens

Tokens are used for authentication in our system. A token contains two attributes of a user: their username and the group they belong to. This allows any server receiving the token to determine which user sent the message, and which permissions they have (which group they are part of). In the case of the root user, the username will be root and the group will be null.

Messages

Message objects are used for all communication between servers. Each method object can store a command, a token, and an ArrayList of objects. The command is used to determine what function is being performed and does not always directly correlate to a command entered by the user. The token holds one token, which currently is always the user's token, or null if no user has been logged in yet or the Message's action does not require a token (e.g exiting the program). The ArrayList of objects holds any other objects that need to be sent between servers, such as a user object or file.

Messages are sent using Object Output/Input Streams in each class. The Message class implements Serializable, which allows messages to easily be sent over the network.

Client-Authentication Server Communication

The client talks to the authentication server whenever it wants permission to handle a given command. The authentication server continuously listens for client commands. A client will give a command, and the authentication server makes sure they have the abilities to do that given task, and generates the user a token to do those abilities.

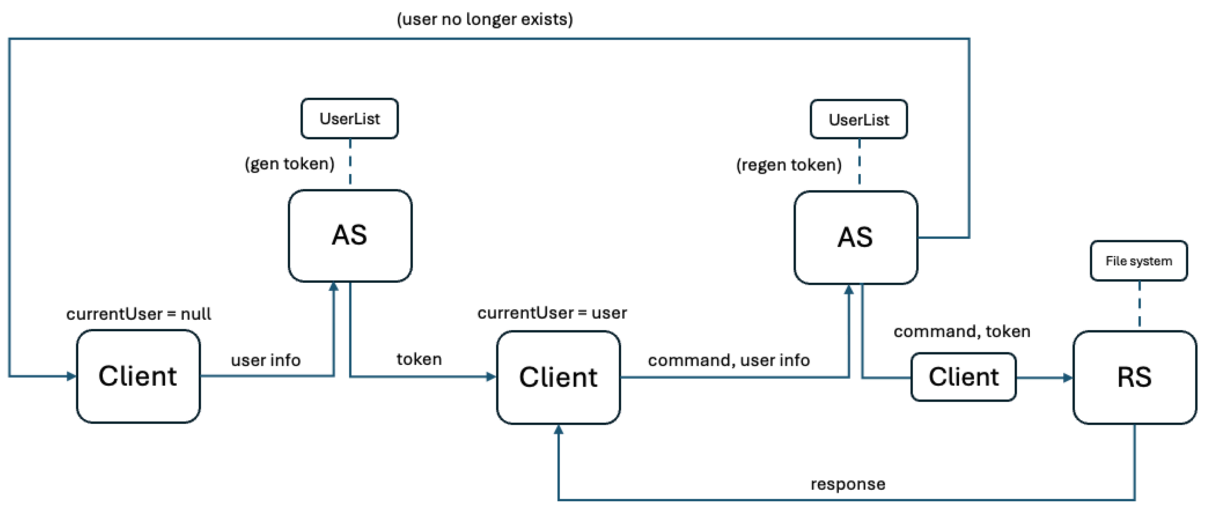
Client-Resource Server Communication

The client talks to the resource server whenever it wants to access the resources they need per given command. The resource server waits for a client to give them a token so they can give them the resources. In our case, those "resources" are the files that a user has access to because they are in the user's group.

Authentication-Resource Server Communication

All communication between the RS and AS will go through the Client, so no direct communication will occur between the two.

Example Flow Using Generic Command



Re-Authentication Strategy

In order to enforce permissions as rigorously as possible, the Resource server will require a token with each request made. This means that each request made by a user will first request a token from the Authentication server before providing that token to the Resource server along with the requested action. This will ensure that any time a user's permissions are changed, it will take effect as soon as another request is made.

File Management

Files uploaded by users are stored in directories named after the group they belong to. Each group's directory contains all files accessible to its members. The RS keeps track of these files and ensures that only users within a group can access the files in that group. When a file is uploaded, it is placed in the appropriate group directory, and when deleted, it is removed from that directory.