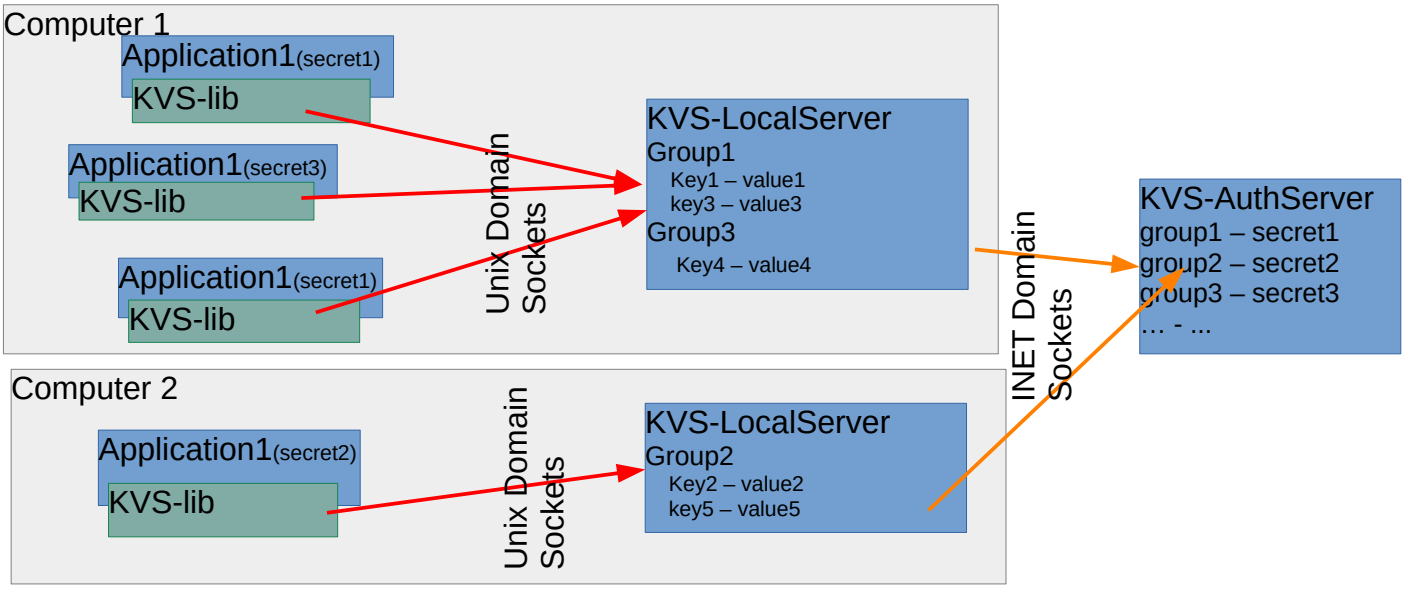
System Programming (MEEC/MEAer)

Project Assignment 2020/2021

In this project the students will implement key value store.

Using this system, application can store values (associated to keys) and share inside a group. Applications belong to a group if they share a secret and use it when connecting to the system.

The components of the KVS (Key-Value System) are as follows:



The applications will use an API implemented by a library to contact and interact with the local server. This server have an user interface for management, receives requests from applications and stores in memory the values. The **KVS-AuthServer** stores information about groups and it secrets.

1 Communication Channels

The applications interact with the library calling C functions declared in the **KVS-lib.h** file and implemented in the **KVS-lib.c** file.

The **KVS-lib** interacts with the **KVS-LocalServer** using UNIX domain stream socket. Before making requests to the **KVS-LocalServer**, each application should call an initialization function from the API that establishes the socket connection and authenticates the application.

The **KVS-LocalServer** interacts with the **KVS-AuthServer** using Internet Domain sockets datagram sockets. The **KVS-AuthServer** can be located in a different computer from the **KVS-LocalServer** and will be accessed by multiple **KVS-LocalServers**.

2 Application interactions

Application use the KVS system to store values associated with key (like an hash-table or dictionary). The data is organized into groups, and only applications belonging to a group can access data of such group.

Groups are created by the administrator (on the **KVS-LocalServer** console) and have a secret associated.

For application to access data, they first need to establishing a connection to the **KVS-LocalServer** providing the groups identifier and the corresponding secret. If they match, then the application can access, manipulate and add data of such group. The application should explicitly close the connection

with the **KVS-LocalServer**.

It is possible for applications to register a callback function that is executed whenever a key-value pair is modified. When the register key-value pair is modified the server **KVS-LocalServer** should send a message to the application and the application runs concurrently with the main the call back function. The call back function is written by the programmer of the application and have the following definition:

- `void f(char * key)`

The registered callback function is invoked by the **KVS-lib** receives as argument the key that was modified and should run in a thread different from the main.

3 KVS-LocalServer

The local server runs on the same computer as applications and allows the sharing of data between applications on the same machine.

This data is stored in a key-value format (like an hash-table or dictionary).

Applications that belong to the same group (establish connection with the correct secret) can access a common set of key-value pairs.

3.1 UI

The **KVS-LocalServer** reads commands from the keyboard in order to configure groups and show status information.

The commands are as follows:

- Create group – receives the identifier of a group (a string) and prints a secret (a string)
- Delete group – removes the group (by deleting the secret and removing all associated data.
- Show group info – prints the following information about a a group:
  - secret
  - number of key-value pairs
- Show application status – lists all currently and past connected application, printing following information:
  - client PID
  - connection establishing time
  - connection close time (if not currently connected).

1.1 Data storage

Each **KVS-LocalServer** stores in memory the key-value pairs. If the server is restarted the information is reset.

1.2 Application authentication

In order to access the data stored in the **KVS-LocalServer**, the application must first connect and provide the group name and the corresponding secrete. If these values match the application can now start to access data associated to such group.

When accepting connection the **KVS-LocalServer** should contact the **KVS-AuthServer** to verify if the provided secret is the one associated with the provided group.

4 KVS-lib

The **KVS-lib** is composed of a set of .c files that should be compiled together with each applications. Students can also generate a **.so** file (as described in week2 laboratory). The corresponding .h file will be provided by the teaching staff and students can not change it.

The .c files should implement the set of functions to be called by applications that will interact with the **KVS-LocalServer** as described next:

- **`int establish_connection (char * group_id, char * secret)`**
  - This function receives as arguments the strings containing the group name and corresponding secret, and tries to open the connection with the **KVS-LocalServer**. If successful, all following operations on key-value pairs are done in then context of the provided group_id.
  - **Return Values** If the secret is correct this function returns 0. If any error occurs the function should return a negative number. For each possible error students should define and return a different return value.
- **`int put_value(char * key, char * value)`**
  - This function tries to assign the provided value to the provided key. If the provided key-pair does not exist in the server, it is created. If the key-value pair already exists in the server it is updated with the provided value.
  - **Return values:** In case of success the function return 1. If any error occurs the function should return a negative number. For each possible error students should define and return a different return value.
- **`int get_value(key, char ** value)`**
  - This function tries to retrieve the value associated to the provided key. If the provided key-pair exists in the server the corresponding value is “returned” through the value argument. This function shold do a malloc that will store the calue associated with the key.
  - **Return values:** In case of success the function return 1. If any error occurs the function should return a negative number. For each possible error students should define and return a different return value.
- **`int delete_value(key)`**
  - This function tries to retrieve delete the pair key-value associated to the provided key.
  - **Return values:** In case of success the function return 1. If any error occurs the function should return a negative number. For each possible error students should define and return a different return value.
- **`int register_callback(char * key, void (*callback_function)(char *))`**
  - This function tries to register a callback the will later be called. The function receives the key that will be monitored and the pointer to the function that will be executed in the applications. When th value associated with the key is changed.
  - **Return values:** In case of success the function return 1. If any error occurs the function should return a negative number. For each possible error students should define and return a different return value.
- int close_connection()

- This function closes the connection previously opened.
- **Return values:** In case of success the function return 1. If any error occurs the function should return a negative number. For each possible error students should define and return a different return value.

5 KVS-AuthServer

The **KVS-AuthServer** runs independently of all the processes and can be located in a different computer. Multiple **KVS-LocalServer** can use the **KVS-AuthServer** to store common group/secret information. The manipulation of the data on this server is done in any **KVS-LocalServer** console.