



Universidad
Nacional
de Córdoba

Cátedra de Sistemas Operativos II

Trabajo Práctico N° I

Integrante: Choquevilca Gustavo
(g.choquevilca@gmail.com)

22 de Abril del 2018



Índice

Introducción	3
Propósito	3
Ámbito del Sistema	4
Definiciones, Acrónimos y Abreviaturas	4
Referencias	4
Descripción General del Documento	4
Descripción General	4
Perspectiva del Producto	5
Funciones del Producto	5
Características de los Usuarios	5
Restricciones	5
Suposiciones y Dependencias	5
Requisitos Futuros	5
Requisitos Específicos	6
Interfaces Externas	6
Requisitos Funcionales	6
Requisitos No Funcionales	7
Requisitos de Rendimiento	7
Atributos del Sistema	7
Diseño de solución	8
Implementación y Resultados	14
Conclusiones	15

Introducción

Un **Socket**^[1] designa un concepto abstracto por el cual dos [programas](#) (posiblemente situados en [computadoras](#) distintas) pueden intercambiar cualquier flujo de datos, generalmente de manera fiable y ordenada.

En una comunicación por socket, el cliente debe conocer la dirección IP del servidor para poder conectarse. Existen varios tipos de sockets, pero los más conocidos y usados son dos: Sockets de Flujo (Stream Sockets), y Sockets Datagrama (Datagram Sockets). Un socket de flujo define la comunicación en dos direcciones, de manera seguro y con conexión. La conexión segura la obtiene mediante el Protocolo de Control de Transmisión (TCP), que asegura que la información llegue en la forma que se envía y sin errores. Por otro parte, la conexión por Datagrama utiliza el Protocolo de Datagramas de Usuario (UDP) que no asegura que la información llegue a destino, y si llega, puede hacerlo de manera desordenada, pero sin errores.

En el presente trabajo práctico se pretende utilizar ambos protocolos para realizar la conexión, control y transferencia entre uno o más clientes con un servidor que posee el baash(Lab 2) de Sistemas Operativos I. El proyecto estará realizado con el protocolo TCP y solamente se utiliza el protocolo UDP, para realizar la transferencia de archivos desde el servidor al cliente.

Este documento está formado por la Especificación de Requisitos Software (ERS) y el Diseño de solución, implementación y resultados, para el Proyecto de Socket en Sistemas de Tipo Unix(SSTU). Esta especificación se ha estructurado inspirándose en las directrices dadas por el estándar “IEEE Recommended Practice for Software Requirements Specification ANSI/IEEE 830 1998”

Propósito

Este documento tiene como propósito conocer y especificar el funcionamiento general del proyecto SSTU. Planteando desde los requerimientos de sistemas hasta la implementación y Resultados. Este documento está dirigido a los usuarios que deseen utilizar el software para la comunicación entre procesos y para aquellas personas que necesiten definir nuevos requerimientos.

Ámbito del Sistema

Este proyecto será utilizado cuando se desee acceder a un servidor con Sistema Operativo(SO) Unix de manera local o remota, para así poder obtener información acerca del sistema o poder realizar transferencias de archivos al cliente. No podrá ser utilizado en SO que no se ha Unix.

Definiciones, Acrónimos y Abreviaturas

- UDP: Protocolo de datagrama de usuario
- TCP: Protocolo de Control de Transmisión
- SO: Sistema Operativo
- baash: es un programa que está implementado en el servidor cuya función es simular un interprete de linea de comando de SO Unix.
- SSTU: Socket en sistemas de tipo Unix
- GB: Giga Byte.

Referencias

[1] Socket de Internet, https://es.wikipedia.org/wiki/Socket_de_Internet, [Abril 2018]

[2] Linux file transmission program based on TCP, https://github.com/scalpelx/Transfer_File, [Abril 2018]

[3] Limpiar el buffer de teclado en Linux con ejemplos en C y C++, <https://poesiabinaria.net/2009/05/buffer-de-teclado-en-linux/>, [Abril 2018]

Descripción General del Documento

El presente Documento describe cómo se implementa la comunicación entre procesos de SO de tipo Unix a través de socket, además se realizará una breve descripción del problema, se expondrán los requerimientos utilizados, se mostrará el diseño y la documentación del proyecto, para finalmente mostrar los resultados obtenidos.

Descripción General

Se presenta una descripción de los requisitos del sistema, con el fin de conocer las principales funciones que debe realizar, perspectiva del producto, funciones del producto, características del usuario , supuestos y dependencias que afectan al desarrollo.



Perspectiva del Producto

Se proyecta implementar un sistema de comunicación entre servidor y cliente, en donde el servidor le proveerá y traspasará información acerca del SO Unix al cliente. Cabe aclarar que el producto resultante es dependiente, ya que tendrá en cuenta al SO Unix que es de donde se extraerá la información para enviársela al cliente.

Funciones del Producto

Las funciones que conforman el sistema son las siguientes:

- Gestión de autenticación (usuario y contraseña) de acceso al servidor.
- Sistema baash instalado en el servidor.
- Gestión de comunicación entre cliente y servidor para la recepción y envío de información.
- Gestión de transferencia de archivos entre servidor y cliente.
- Gestión control y manejo de errores por parte del servidor con comunicación al cliente.

Características de los Usuarios

Tener conocimiento medio o superior de manejo SO Unix, es decir ya sea para poder ejecutar los programas hasta reconocer los resultados que obtendrá por pantalla el cliente.

Restricciones

- En cuanto a las restricciones de software se exige que el proyecto SSTU funcione bajo SO tipo Unix.
- En cuanto a las restricciones de hardware: disponer de computadoras con memoria RAM superior a 1 GB, procesador mínimo Pentium 2

Suposiciones y Dependencias

Utilizar SO Unix, en SO Windows u otros no funciona el proyecto ya que el servidor está implementado con un baash Unix. Así por ejemplo aplicar el proyecto en un SO Windows implicaría modificar los requerimientos.

Requisitos Futuros

Una futura mejora es que el proyecto pueda funcionar en cualquier plataforma de SO

Requisitos Específicos

En este apartado se presentan los requisitos funcionales que deberán ser satisfechos por el sistema. Todos los requisitos aquí expuestos son esenciales, es decir, no sería aceptable un sistema que no satisfaga alguno de los requisitos aquí presentados.

Interfaces Externas

- **Interfaz de usuario:** es el terminal del SO, en donde se mostrará toda la información necesaria por el usuario.
- **Interfaz de Hardware:** es la pantalla del monitor ya que por ella se mostrara los resultados obtenidos. Otra interfaz es el teclado por donde se ingresara los comando a ejecutar.

Requisitos funcionales

- **RF1:** El cliente debe conectarse al servidor en modo seguro(TCP)
- **RF2:** Para realizar las conexiones TCP UDP se debe utilizar el puerto 6020.
- **RF3:** El cliente debe tomar un número de puerto libre del SO
- **RF4:** El servidor tiene un baash instalado.
- **RF5:** El servidor tiene un mecanismo de autenticación usuario y contraseña
- **RF6:** El cliente para conectarse al servidor debe ingresar el prompt: *connect usuario@numero_ip:puerto.*
- **RF7:** El cliente tendrá una contraseña asociada al usuario.
- **RF8:** Si la contraseña o usuario son incorrectos, el servidor debe denegar el acceso al cliente y notificarlo con la leyenda *"nombre de usuario y/o contraseña incorrecto"*
- **RF9:** Si la autenticación es correcta, el prompt del cliente debe cambiar: *usuario@servidor:~\$.*
- **RF10:** Si la autenticación es correcta el cliente debe enviar comandos que recibe el servidor y ejecuta el baash.
- **RF11:** Toda salida del baash debe ser transmitida por el mismo socket al cliente y ser mostrado en pantalla.
- **RF12:** Estará disponible un función que permite descargar un archivo del servidor y enviarlo en modo no conexión(UDP) al cliente. El prototipo de la función es: *descarga nombre_archivo*
- **RF13:** Durante la descarga se deberá bloquear la terminal del cliente.



Requisitos no funcionales

- **RNF1:** Para poner en funcionamiento el servidor y el cliente se utiliza SO unix
- **RNF2:** El servidor debe estar en una placa de desarrollo que puede ser INTEL Galileo o Raspberry pi.
- **RNF3:** El cliente será una computadora.
- **RNF4:** El sistema deberá ser sencillo e intuitivo, tanto en el servidor como en el cliente.
- **RNF5:** Todos los procesos deberán ser ejecutados en mono-thread y no existirá ningún sistema de base de datos.
- **RNF6:** El servidor tendrá un mecanismo de control y manejo de errores

Requisitos de Rendimiento

El sistema soporta la conexión en simultáneo de 5 personas(con usuario y contraseña). Es decir un número de 5 terminal abiertas en simultáneo realizando consultas al servidor.

Atributos del Sistema

Cuando un cliente intente conectarse al sistema deberá introducir su identificación (login) y clave de acceso, y el sistema deberá comprobar que se trata de un usuario autorizado. Si el identificador introducido no corresponde a un usuario autorizado o la clave no coincide con la almacenada, se notifica al cliente y debe reingresar nuevamente . El sistema de autenticación tendrá distintos de usuarios y a cada uno de ellos se le permitirá el acceso total al sistema, pudiendo ejecutar comando y realizar descargas de archivos.

Diseño de solución

Para el realizar la solución del proyecto se utilizó la arquitectura cliente-servidor donde se atiende las solicitudes de las operaciones propuestas utilizando como lenguaje de Programación C.

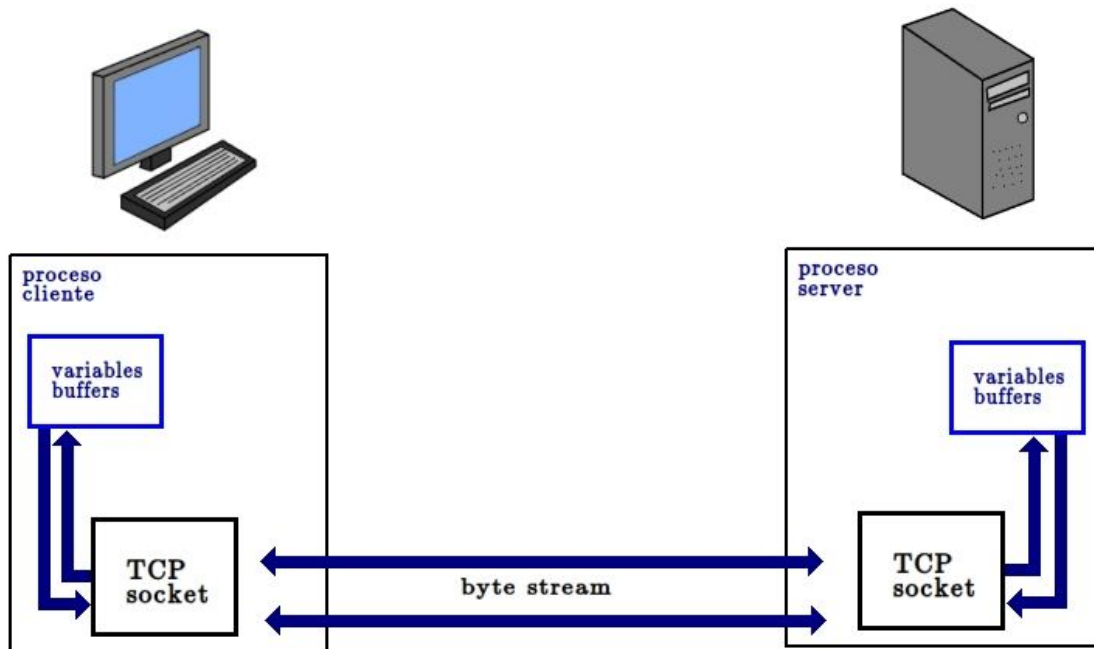


figura.1

Todo el proyecto está orientado a conexión TDP y tiene la siguiente arquitectura:

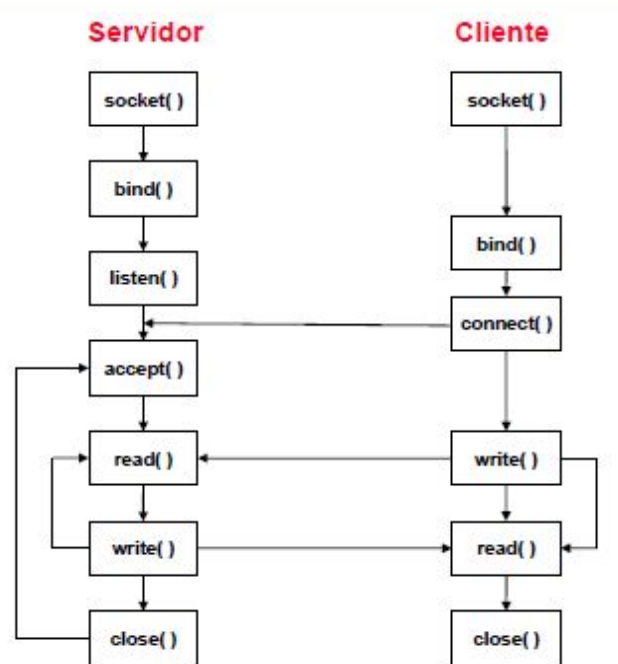


figura.2

Solamente la descarga de archivos está orientada a no conexión UDP y tiene la siguiente arquitectura.

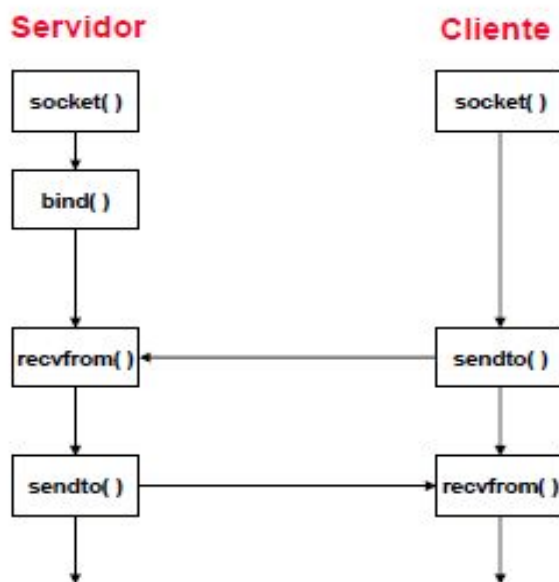


figura.3

Diagramas UML

A continuación se muestra los diagramas UML necesarios para la explicación de los requerimientos de software. Solo se realizaron los diagramas de caso de uso, actividad y secuencia.

Diagrama de caso de uso

Una vez conocidos los requerimientos del sistema se construye un modelo de requerimientos utilizando casos de uso. El sistema cuenta con dos actores y 14 caso de usos.

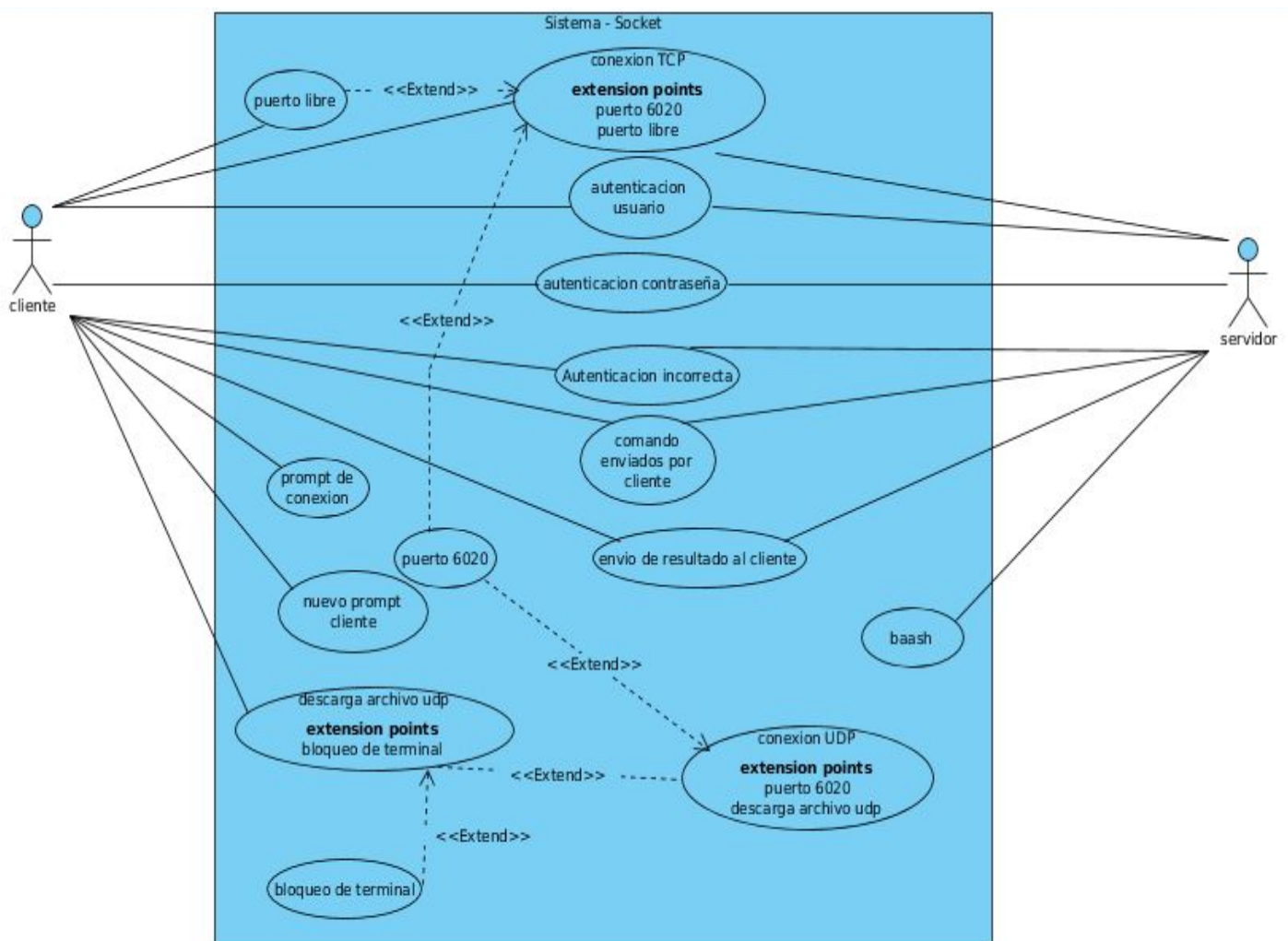


figura.4

Diagrama de Actividades

Otro diagrama que muestra el comportamiento del sistema es el diagrama de actividad. A continuación mostramos dicho diagrama:

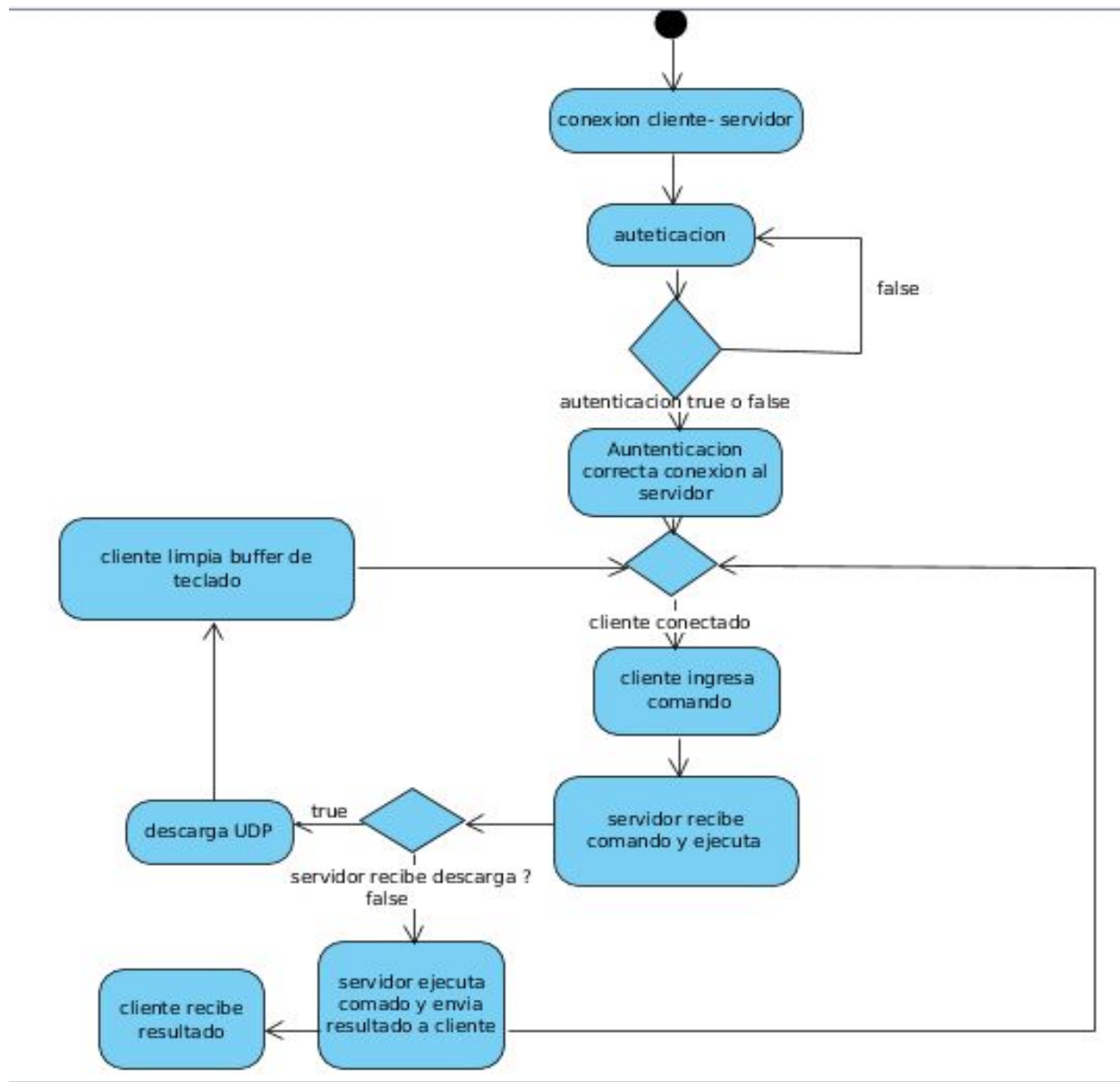
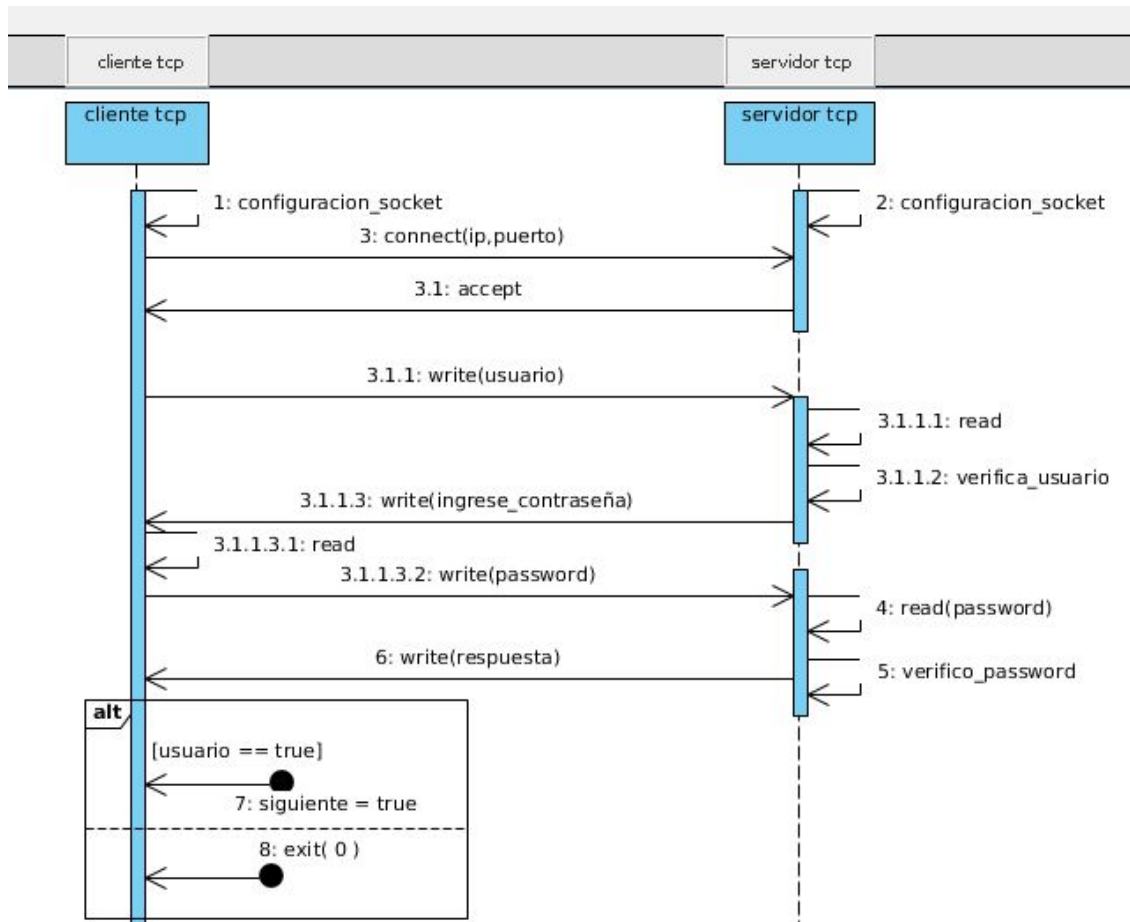


figura.5

Diagrama de Secuencia

Por último se muestra el diagrama de secuencia que sigue el proyecto.



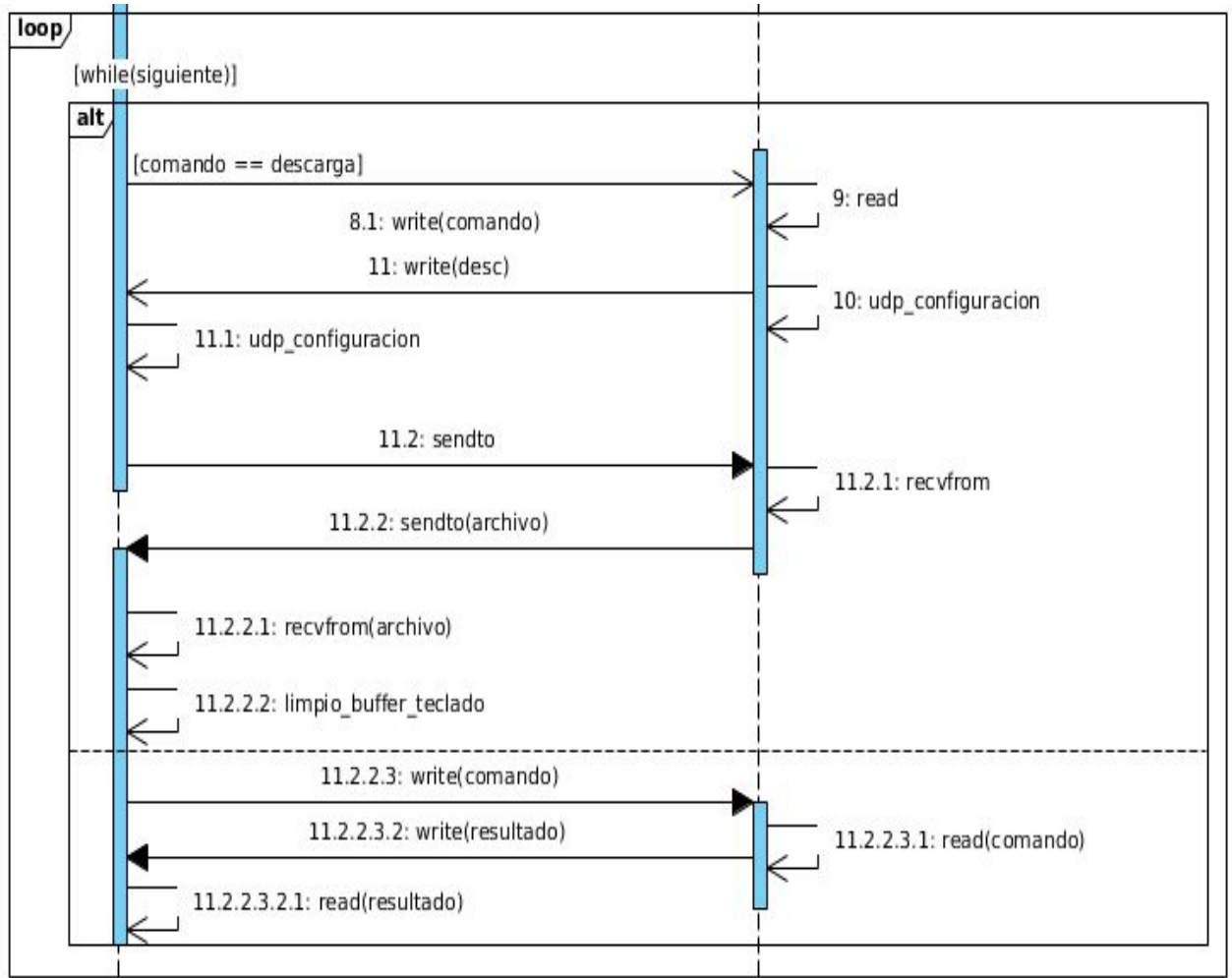


figura.6

Implementación y Resultados

La implementación y resultado se puede observar en las siguientes imágenes:

```
ggg@ggg:~/Documentos/SISTEMAS_OPERATIVOS_II/TRABAJOS_PRACTICOS/TP1/1
ENTE$ ./cliente
cliente: connect gustavo@127.0.0.1:6020
Usuario es: gustavo
Ip del servidor es: 127.0.0.1
Puerto es: 6020

-----Autenticacion de Usuario-----
Usuario conectado: gustavo
Servidor: Usuario y Contraseña correcto

-----BIENVENIDO AL BAASH-----

/////////////////////////////////SISTEMA OPERATIVO II/////////////////////////////////
TRABAJO PRACTICO N°1: SOCKET-TCP-UDP
ALUMNO: CHOQUEVILCA ZOTAR GUSTAVO BRAIAN
/////////////////////////////////

/////////////////////////////////LISTA DE DIRECTORIOS EN EL PATH/////////////////////////////////
PATH[0] = /usr/local/sbin
PATH[1] = /usr/local/bin
PATH[2] = /usr/sbin
PATH[3] = /usr/bin
PATH[4] = /sbin
PATH[5] = /bin
PATH[6] = /usr/games
PATH[7] = /usr/local/games
PATH[8] = /snap/bin
/////////////////////////////////

---Usuario y Contraseña correcto---
Usted se conecto al servido baash, puede ejecutar comandos

gustavo@servidor-baash:~$: 
```

figura.7

En la figura.7 se puede observar como el cliente (lado izquierdo) se conecta por primera vez al servidor (lado derecho) estableciendo su usuario ip y puerto para luego después ingresar su contraseña.

```
gustavo@servidor-baash:~$: ls
a.out
baash.txt
boot.pdf
INFORME_FINAL_ejbca.pdf
Makefile
mjtema2.mp3
password.txt
servidor
servidor.c
tema3.mp3
tema.mp3
usuario.txt
```

figura.8

En la figura.8 se puede observar cómo ejecutando el comando ls se obtiene en el cliente todos los directorios del servidor

```

ggg@ggg: ~/Documentos/SISTEMAS_OPERATIVOS_II/TRABAJOS_PRACTICOS/TP1/
Archivo Editar Ver Buscar Terminal Ayuda
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
gustavo@servidor-baash:~$ ls
a.out
baash.txt
boot.pdf
INFORME_FINAL_ejbca.pdf
Makefile
mjtema2.mp3
password.txt
servidor
servidor.c
tema3.mp3
tema.mp3
usuario.txt
gustavo@servidor-baash:~$ descarga boot.pdf
Nombre de fichero: boot.pdf
Tamaño de fichero real: 128586 bytes
Tamaño de fichero descargado: 128586 bytes
Archivo recibido correctamente...
gustavo@servidor-baash:~$

ggg@ggg: ~/Documentos/SISTEMAS_OPERATIVOS_II/TRABAJOS_PRACTICOS/TP1/TP1V
Archivo Editar Ver Buscar Terminal Ayuda
////////////////LISTA DE DIRECTORIOS EN EL PATH////////////////
PATH[0] = /usr/local/sbin
PATH[1] = /usr/local/bin
PATH[2] = /usr/sbin
PATH[3] = /usr/bin
PATH[4] = /sbin
PATH[5] = /bin
PATH[6] = /usr/games
PATH[7] = /usr/local/games
PATH[8] = /snap/bin
////////////////
ggg@ggg:/home/ggg/Documentos/SISTEMAS_OPERATIVOS_II/TRABAJOS_PRACTICOS/TP1/TP1V$
ggg@ggg:/home/ggg/Documentos/SISTEMAS_OPERATIVOS_II/TRABAJOS_PRACTICOS/TP1/TP1V$
Socket disponible en UDP: 6020
Descarga archivo
Nombre de fichero: boot.pdf
Tamaño de fichero: 128586 bytes
Tamaño de fichero enviado: 128586 bytes
Se envió el archivo...
Envío terminado de archivo...

```


figura.9

En la figura.9 se puede observar cómo se realiza la descarga en udp del fichero boot.pdf, el cliente puede reconocer el nombre del fichero, el tamaño del fichero y el tamaño de fichero descargado, por otra parte el servidor puede reconocer también el nombre del fichero el tamaño real del fichero, el tamaño enviado de fichero y cuando termina de enviar todo el contenido se muestra un mensaje de Envío terminado de archivo.

Conclusiones

En general, podemos concluir que es posible realizar aplicaciones que nos permiten la comunicación entre dos sistemas en este caso servidor y cliente, la forma de realizarlo es mediante Sockets.

Se comprendió el funcionamiento básico de esta herramienta, la cual es de gran utilidad a la hora de conectar procesos que pueden estar en una misma máquina o en distintas computadoras. Los sockets solo pueden conectarse con sockets de su mismo tipo y familia, por lo que hay que tener muy claro esto a la hora de realizar alguna aplicación.



Se puede mencionar que una de los inconveniente a la hora de realizar el proyecto fue cuando realice la implementación de la descarga en udp ya que en un principio funcionó pero luego pude darme cuenta que no estaba realizando la descarga correcta, con el correr de los días y analizando la función descarga corregí el error y finalmente pude realizar la descarga correcta del fichero.

