CÁTEDRA DE SISTEMAS OPERATIVOS II

Departamento de Computación

FCEFyN - UNC

SOCKETS

AUTOR: Tissot Esteban

e-mail: egtissot@gmail.com

Cordoba, 14/03/2017

Indice

INTRODUCCION

En la Provincia de Córdoba existen aproximadamente 400 estaciones hidrometeorológicas automaticas (AWS), de distintas marcas, modelos y configuraciones, que pertenecen a distintas redes, instituciones y organismos, desparramadas por todo el territorio de provincial. Cada estación consta de una serie de sensores (barómetro, termómetro, sensor de radiación solar, etc.), conectado a un sistema de adquisición de datos que toma medidas de los sensores (telemetría) cada un periodo determinado de tiempo.

En este trabajo se desarrollarán e implementarán temas de interés para realizar la conexión, control y transferencia de datos telemétrico entre un cliente y un servidor .

Se utilizará una conexión segura mediante el protocolo TCP (orientado a la conexión) para las solicitudes especificas de datos, y una conexión no segura mediante el protocolo UDP (no orientado a la conexión) para cuando se requiera descargar el archivo con el total de los datos del servidor.

Para realizar estas dos conexiones se utilizará el concepto de SOCKETS. Un socket se puede definir como un método para la comunicación entre un cliente y servidor en una red. Los componentes necesarios para la correcta implementación, es un par de direcciones IP local y remota, un protocolo de transporte y un par de números de puerto local y remoto.

Se supone que las pruebas de compilación se realizarán en un equipo que cuenta con las herramientas típicas de consola.

DESCRIPCION GENERAL

Para poder realizar la transferencia de datos se implementa la arquitectura cliente-servidor que permite realizar una conexión entre una placa Raspberry Pi (servidor) y una computadora personal (cliente).

El servidor contiene datos sobre las distintas estaciones hidrometeorológicas de la provincia de Córdoba. Estos datos se guardan en un archivo .CVS con el siguiente formato:

Numero,Estacion,ID Localidad,Fecha,Temperatura [ºC],Humedad [%HR],Punto de Rocío [ºC],Precipitación [mm],Velocidad Viento [Km/h],Dirección Viento,Rafaga Maxima [Km/h],Presión [hPa],Radiación Solar [W/m2],Temperatura Suelo 1 [ºC],Temperatura Suelo 2 [ºC],Temperatura Suelo 3 [ºC],Humedad Suelo 1 [%grav],Humedad Suelo 2 [%grav],Humedad Suelo 3 [%grav],Humedad de Hoja [%],

El programa que se ejecutará en el servidor será capaz de parsear los datos y responder frente a las solicitudes del cliente luego de que el cliente inicie la conexión y sea logueado correctamente.

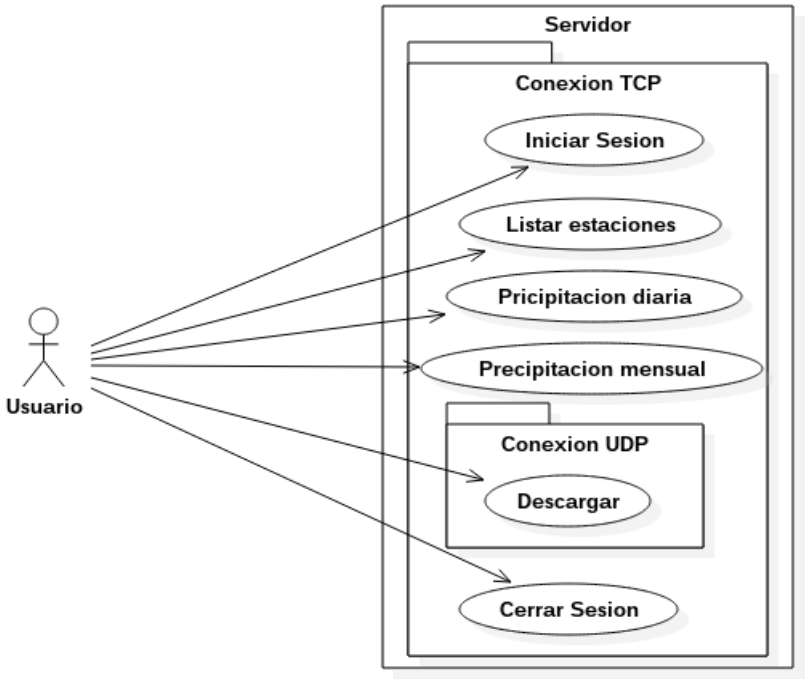
REQUISITOS ESPECÍFICOS

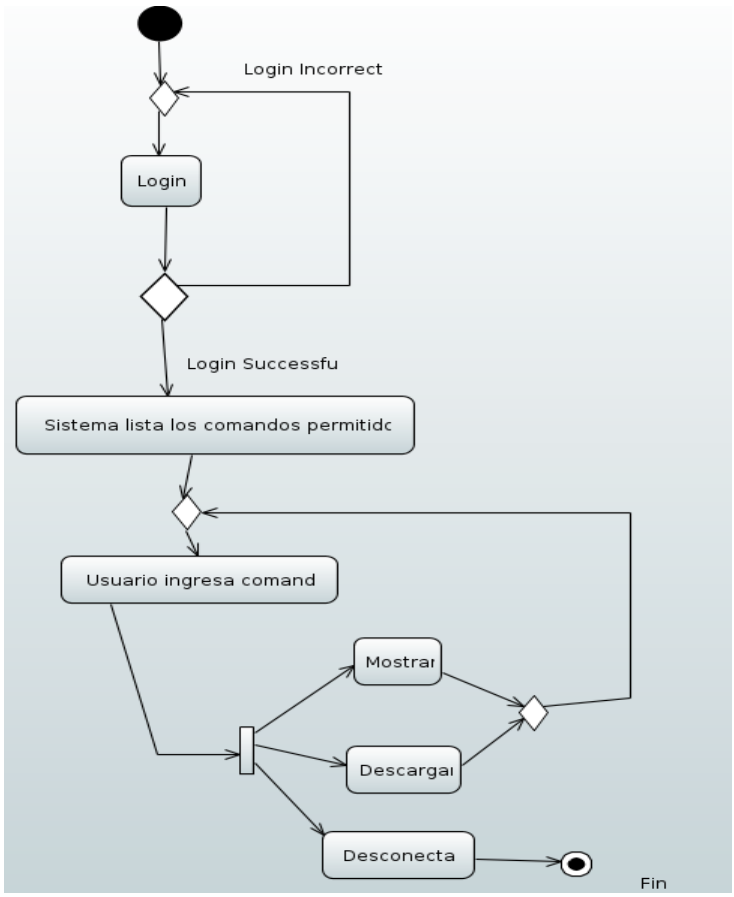
Requerimientos Funcionales

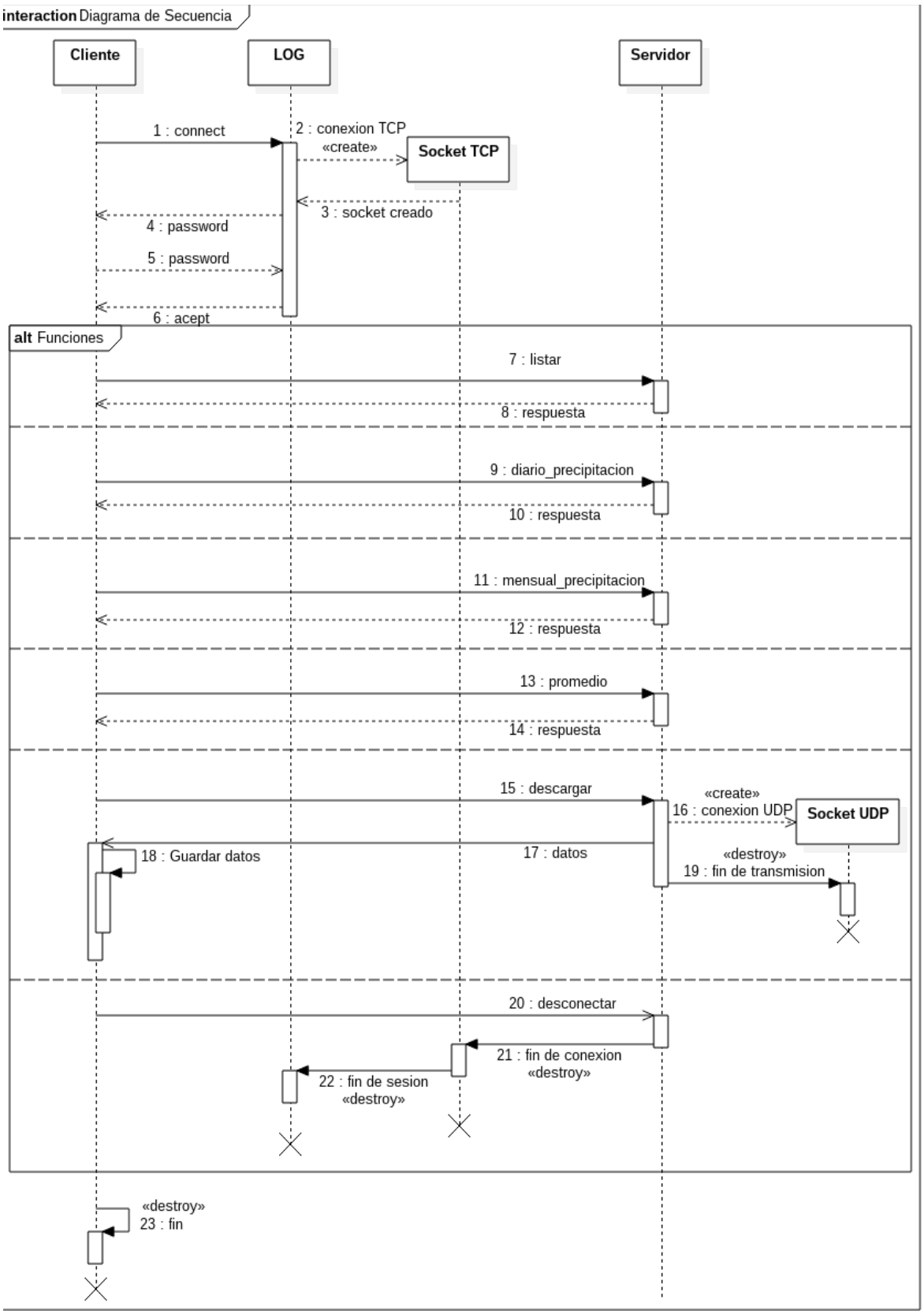
- Los programas desarrollados deberán ser en lenguaje C y en la compilación se deberá hacer uso de flags de warning (-Werror, -Wall y -pedantic).
- Ambos programas deben ser capaces de soportar conexiones TCP y en caso de descarga una conexión UDP.
- El AWS debe poder almacenar los datos censados en un archivo con formato csv.
- El AWS debe tener un puerto fijo (6020).
- Una vez iniciada la conexión se deberá chequear el usuario y pedir la correspondiente contraseña.
- El AWS debe ser capaz de responder a las solicitudes de parte del cliente y tener un control de errores.
- El programa cliente debe poder conectarse de forma segura y debe iniciar una conexión no segura para realizar la descarga del archivo cvs.
- El promp del cliente debe identificar al usuario logueado y a que servidor esta conectado.
- Una vez aceptado el usuario y contraseña se debe imprimir una lista con los posibles comandos.
- Durante la operación de transferencia del archivo se debe bloquear la interfaz de usuario.
- El cliente debe poder loguearse con multiples usuarios y a multiples servidores sin necesidad de reabrir el programa.
- Funciones a desarrollar:
    - **listar:** muestra un listado de todas las estaciones que hay en la “base de datos”, y muestra de que sensores tiene datos.
    - **descargar {*n.º de estación*}**: descarga un archivo con todos los datos de n.º estacion.
    - **diario_precipitación {*n.º de estación*}**: muestra el acumulado diario de la variable precipitación de n.º estación.
    - **mensual_precipitacion {*n.º de estación*}**: muestra el acumulado mensual de la variable precipitación.
    - **promedio {*variable*}**: muestra el promedio de todas las muestras de la variable de cada estación.
    - **desconectar**: termina la sesión del usuario.

DISEÑO E IMPLEMENTACIÓN

Diagramas UML




Login Incorrect
Login
Login Successfu
Sistema lista los comandos permitidc
Usuario ingresa comand
Mostrar
Descargar
Desconecta
Fin



Las imágenes anteriores corresponden a los diagramas UML de caso de uso, actividad y secuencia que se utilizaron en la planificación del proyecto. A partir de estos diseños se desarrollaron los programas con sus respectivas funcionalidades.

Programas

Cliente:

Al iniciar el programa se muestra un promp (>>). En este estado solo se aceptará el comando "***connect***", el cual recibe como parámetro el usuario, el servidor al cual se quiere conectar (dirección ip) y el numero de puerto. El comando seguira el siguiente formato:

connect user@127.0.0.1:6020 donde 127.0.0.1 debe reemplazarse por la ip correspondiente al servidor que se quiere utilizar. Luego de introducir este comando, el sistema va a pedir una contraseña que se debe corresponder con el usuario, si todo esta bien en este puto esta creada la conexión TCP con el servidor.

El promp en esta etapa cambia, esta formado por el nombre de usuario y la ip del servidor y el sistema ya es capaz de aceptar las funciones descriptas en los requerimientos de usuario. Si se ejecuta el comando descargar, el sistema creara una conexión UDP para realizar la descarga a la computadora. Se creará un archivo llamado datos.txt con la información provista por el servidor. Tener en cuenta que una conexión de este tipo puede perder datos.

Si se ejecuta el comando desconectar el programa finalizara la sesión y la conexión habra terminado, sin embargo se regresa al estado inicial, donde el promp es ">>", para finalizar el programa se debe introducir el comando ***"fin"***.

Servidor:

Este programa se ejecuta sobre una placa Raspberry Pi (modelo B+), y al momento de inicio crea un socket TCP y se queda escuchando el puerto 6020 hasta que algún cliente solicite una conexión.

Cuando un cliente solicita la conexión, este responde y pide una contraseña. Luego de chequear que se hayan introducido correctamente el nombre de usuario y contraseña queda establecida la conexión TCP con el usuario logueado. En caso de que el nombre de usuario o contraseña sean incorrectos se devuelve un mensaje de error.

En este punto el servidor esta en condiciones de responder a las solicitudes por parte del cliente descriptas anteriormente.

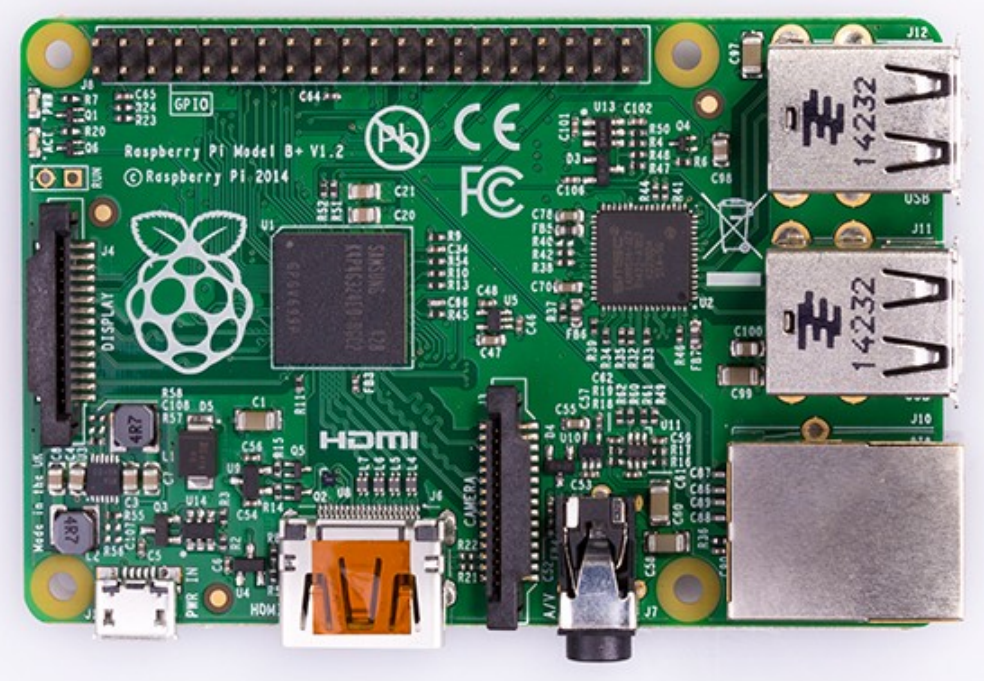
La conexión y sesión finalizan luego de recibir el comando "***desconectar***". Sin embargo el programa no finaliza, sigue escuchando por el puerto 6020 esperando nuevos clientes.

CONCLUSIÓN

Este trabajo permite una conexión, mediante socket, tanto TCP como UDP entre una Raspberry Pi y una computadora personal. Si bien es una primera versión de los programas, ambos son estables, modulares y escalables. Para una próxima revisión se aconseja mejorar el control de errores y notificación de los mismos.

APÉNDICE

RASPBERRY PI (Model B+)



Especificaciones Básicas:

- Procesador Broadcom Soc de 700MHz
- 512 MB de RAM
- Consumo de 0,5 W
- 40 pines GPIO
- Conector microUSB
- 4 USBs 2,0
- 1 puerto ETHERNET
- 1 puerto HDMI
- Soporte para microSD
- Salida de 3,3V y 1,8V

https://www.raspberrypi.org/products/model-b-plus/