



Cátedra de Sistemas Operativos II

Trabajo Práctico N° 1

Arce Giacobbe, Alejandro
11 de abril del 2019

Índice

Introducción	3
Propósito	3
Ámbito del Sistema	3
Descripción General	3
Perspectiva del Producto	3
Funciones del Producto	3
Restricciones	5
Requisitos Específicos	5
Restricciones de Diseño	5
Requisitos de Rendimiento	6
Diseño de solución	6
Implementación y Resultados	6
Conclusion	8
Referencias	9

Introducción

Propósito

El siguiente trabajo práctico tiene como objetivo la implementación de una aplicación cliente/servidor que se conectan a través de Sockets, tanto TCP como UDP, permitiendo el traspaso de información entre sí.

Ámbito del Sistema

El trabajo se llevó a cabo en C, en su totalidad. Se programó utilizando Sublime Text en la PC del alumno, y también se hizo uso de una placa Raspberry Pi.

Se pueden diferenciar dos etapas: la primera, es una implementación mediante sockets 'UNIX' en una computadora de uso general; la segunda, es una implementación mediante sockets de Internet y haciendo uso de un sistema embebido para el cliente.

Descripción General

Perspectiva del Producto

Se trata de un software que permite realizar la conexión, control y transferencia de un programa servidor con n (en este caso $n = 1$) programas clientes, que simulan un satélite geoestacionario con su correspondiente estación terrena.

Funciones del Producto

El servidor (la estación terrena) proporciona un prompt, al cual se accede mediante validación de usuario y contraseña que se le solicita al usuario cuando inicia el programa. De ingresar credenciales erróneas, el sistema notifica al usuario el usuario y le pide que las ingrese nuevamente. De ocurrir esto tres veces, el programa se cerrará automáticamente.

De ingresar satisfactoriamente al sistema, se esperará por la conexión con el cliente (el satélite), una vez establecida, el usuario puede ingresar los siguientes comandos:

- update firmware.bin

Edita el número de versión del código del cliente, lo compila y envía el archivo binario al satélite con una actualización del cliente, subido el archivo, se reinicia el cliente con la nueva actualización.

- start scanning

Se realiza el envío de una imagen satelital desde el cliente hacia el servidor. La imagen tiene un tamaño considerable, por lo que se encuentra previamente fragmentada y codificada en base64 para facilitar el envío, por lo que el cliente envía estos fragmentos uno por uno en el orden correspondiente y el servidor se encarga de almacenarlos, unirlos y decodificarlos. El tamaño de cada fragmento está determinado por el tamaño máximo de un datagrama.

- obtener telemetría

El satélite se encarga de obtener y enviar a la estación terrena la siguiente información:

- Id del satélite; Uptime del satélite; Versión del software; Consumo de memoria y CPU

A diferencia de los otros comandos, este envío se realiza a través de un socket no persistente y no orientado a la conexión (udp).

- exit

Finaliza la ejecución tanto del servidor como del cliente.

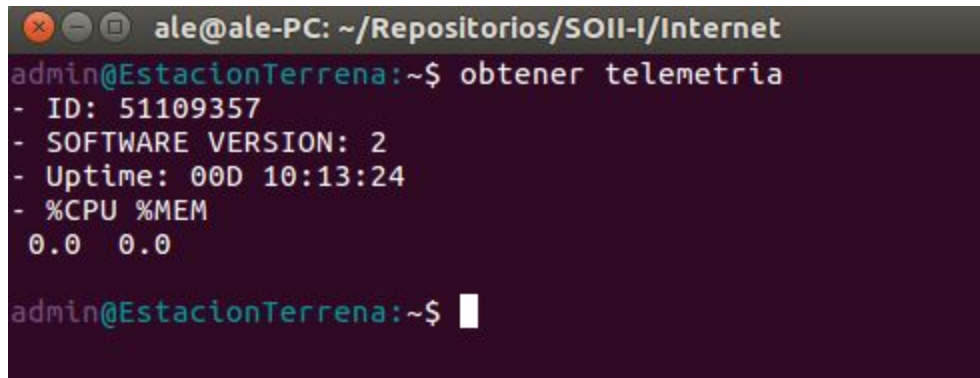


```
ale@ale-PC: ~/Repositorios/SOII-I/Internet
ale@ale-PC:~/Repositorios/SOII-I/Internet$ ./server
Uso: ./server <puerto>
ale@ale-PC:~/Repositorios/SOII-I/Internet$ make runserver
./server 27415

USER: ale
PASS: 123
Nombre de usuario y/o contraseña incorrecto, quedan 2 intentos
USER: admin
PASS: 123
***Autenticado correctamente***

-- Esperando la conexión con el satélite --
```

Fig 1: Ingreso de credenciales y espera de conexión en el servidor

A terminal window with a dark background and light-colored text. The window title is 'ale@ale-PC: ~/Repositorios/SOII-I/Internet'. The prompt is 'admin@EstacionTerrena:~\$'. The command 'obtener telemetria' has been entered, and the output is displayed as follows:

```
admin@EstacionTerrena:~$ obtener telemetria
- ID: 51109357
- SOFTWARE VERSION: 2
- Uptime: 00D 10:13:24
- %CPU %MEM
  0.0  0.0
admin@EstacionTerrena:~$
```

Fig 2: prompt del servidor y función 'obtener telemetría'

Restricciones

El software del servidor únicamente se puede conectar con un cliente a la vez, por lo que, si se desea agregar más 'satélites', se debe modificar el código.

También se debe tener en cuenta que el programa se escribió para ser ejecutado en Ubuntu, en el caso del servidor, y Raspbian, en el caso del cliente.

Por otro lado, el programa en sí es muy limitado, ya que cuenta solamente con 3 funciones.

Requisitos específicos

Restricciones de diseño

- El cliente se debe conectar a la estación terrena de forma segura (orientado a conexión), a una IP y puerto fijo.
- El cliente, debe tomar un número de un puerto libre de su sistema operativo.
- El programa que corre en la estación terrena debe proporcionar un prompt
- Al prompt se accede mediante validación de usuario y contraseña que se le solicita al usuario cuando inicia el programa.
- De ingresar credenciales erróneas, el sistema debe notificar al usuario el usuario y pedirle que las ingrese nuevamente.
- Al tercer intento incorrecto, debe terminarse el programa.
- Usar sockets Stream para todos los comandos, excepto obtener telemetría, el cual se debe implementar utilizando Datagram

- La estación terrena se debe correr en una computadora de propósito general y los satélites en un sistema embebido.
- Debe incluirse un mecanismo de control y manejo de errores en todo el sistema.
- Todos los procesos deben ser mono-thread.
- Se pide utilizar el estilo de escritura de código de GNU o el estilo de escritura del kernel de Linux.

Requisitos de Rendimiento

Se pide que el envío de la imagen mediante la función 'start scanning' sea lo más eficiente posible, y que se mida el tiempo que este proceso toma para completarse.

Diseño de solución

Se comenzó diseñando una solución que implementara los sockets de UNIX de forma local en la computadora, y a partir de esta se realizó una nueva implementación que utiliza sockets de Internet. Por último se debieron hacer algunas modificaciones para poder correr el cliente en la Raspberry Pi.

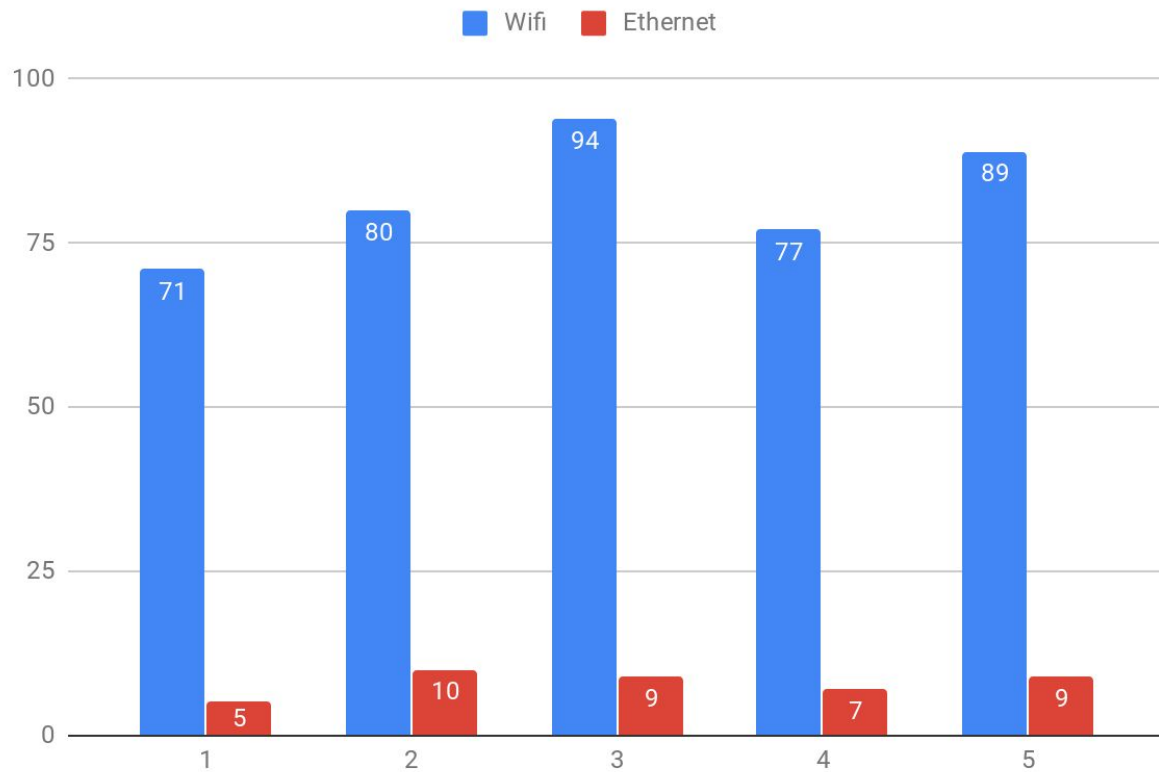
Implementación y resultados

Se fue probando exhaustivamente que los métodos funcionaran según lo esperado; en caso de no ser así se iban realizando las modificaciones necesarias a éstos.

Para ver el rendimiento de la función 'start scanning', ésta se corrió utilizando tanto una conexión Wi-Fi, como una cableada (Ethernet). En total se hicieron 5 pruebas con cada conexión.

Los resultados obtenidos se pueden observar en los siguientes gráficos; donde en el eje vertical se muestra el tiempo demorado en la recepción de la imagen en segundos, y en el horizontal el número de ejecución. Cabe destacar que el tiempo se midió haciendo una llamada al sistema con el comando 'date', tanto al inicio de la recepción como al final y haciendo la diferencia de ambos tiempos. Esto fue así ya que el comando clock() de la librería time.h no medía el tiempo de forma adecuada.

Tiempo de transferencia/recepción de Imagen Satelital



Como era de esperar la conexión cableada resultó ser mucho más rápida con una media de 8[s]

Conclusión

Se pudo comprender el concepto general de 'sockets' y el funcionamiento particular de los sockets tipo Internet y UNIX; además se hizo un buen repaso al lenguaje 'C', y muchas de sus funciones de gran utilidad general.

Por otro lado, se tuvo la experiencia de trabajar con una placa Raspberry Pi, la cual fue necesario configurar desde cero.

Referencias

1. Sockets <http://man7.org/linux/man-pages/man2/socket.2.html>