Universidad Nacional
de Córdoba

Cátedra de Sistemas Operativos II

Trabajo Práctico Nº I

ZIMMEL CECCÓN, Ezequiel José

2 de febrero de 2020

Índice

Introducción

Los sockets son una abstracción de comunicación entre procesos (IPC) que, en un sistema tipo UNIX, se implementan en un descriptor de archivo, sobre el cual se envía o recibe información, al igual que como se lee o escribe un archivo. Éstos a su vez permiten la comunicación entre sistemas remotos mediante la implementación del socket sobre el stack TCP/IP. Son una herramienta muy importante, uno de los pilares de la comunicación entre procesos, y sumamente utilizada en la mayoría de las aplicaciones de red.

En esta sección se encuentra la especificación de los requerimientos de la aplicación Sockets correspondiente al trabajo práctico N° 1. Ésto es, se establecen las características que debe tener el sistema o las restricciones que debe satisfacer para la aceptación del cliente.

Propósito

El propósito de este Trabajo Práctico es aprender a utilizar la estructura de sockets de Linux, sus características y limitaciones, el uso de sus diferentes tipos (orientados o no a la conexión), y su posterior implementación y testing.
Los conceptos deben aplicarse a un caso particular referido a una estación terrestre (servidor) que debe interactuar con un satélite (cliente), y entre los mismos enviar o recibir distintos datos, como la información del satélite, archivos de actualización de firmware o la imagen de la tierra vista desde el satélite.

Ámbito del Sistema

Bajo implementación de socket INET, el sistema está conformado por dos partes: la aplicación servidor, que se ejecuta desde la computadora del usuario, y la aplicación cliente, que se ejecuta en una placa Raspberry Pi 3. Ambos comunicándose por medio de sockets INET.
Por otro lado, bajo implementación de socket UNIX, el sistema se conforma por una única parte, compartiendo un archivo a los fines de la comunicación entre el proceso cliente y el proceso servidor.

Definiciones, Acrónimos y Abreviaturas

- TCP: Transmission Control Protocol (Protocolo de Control de Transmission).
- UDP: User Datagram Protocol (Protocolo de Datagramas de Usuarios).
- Sockets: "canal de comunicación" entre dos programas que corren sobre ordenadores distintos o incluso en el mismo ordenador.
- INET: socket de internet.
- UNIX: socket interprocesos.

Referencias

1. Michael Kerrisk, The Linux Programming Interface, 2010, Adison-Wesley
2. Imagen del satelite Goes16, ”https://cdn.star.nesdis.noaa.gov/GOES16/ABI/FD/13/20190811400GOES16-ABI-FD-13-10848x10848.jpg.zip”
3. Making The Best Use of C, https://www.gnu.org/prep/standards/html_node/Writing-C.html
4. L.Trovalds, Et al., https://github.com/torvalds/linux/tree/master/Documentation/process

Descripción General del Documento

El propósito de este documento es proporcionar a quien lo lea un entendimiento básico de los requisitos y alcance del proyecto así como los pasos que se tomaron para su diseño e implementación.

El documento está organizado en 6 secciones principales:

- Descripción de las generalidades del documento y del proyecto.
- Descripción de la perspectiva del producto, sus funciones y características del usuario y sus restricciones.
- Referencias a los requerimientos del producto, interfaces y casos de uso, incluyendo diagramas y gráficos que facilitan su entendimiento.
- Diseño de la solución.
- Implementación de la solución y resultados obtenidos.
- Conclusiones a las tareas realizadas.

Descripción General

En la presente sección se explicarán los distintos aspectos del trabajo. Las funciones que ofrece el producto, la perspectiva del mismo, las características de los usuarios, requisitos para futuras iteraciones.

Perspectiva del Producto

El producto está orientado a la comunicación de satélites con la estación terrestre para que ésta pueda recibir las distintas imágenes satelitales, el estado del satélite, y en caso de ser necesario actualizar el firmware del satélite desde la tierra.

Funciones del Producto

El producto provee las siguientes funciones:

- Conectividad usuario-servidor mediante IP y puerto (socket INET), o mediante archivo (socket UNIX).
- Autentificación de usuario para brindar seguridad en los datos. Característica del servidor, que verificará la existencia del usuario y contraseña, dando acceso a la terminal interactiva.
- Control de acceso con 3 intentos de autenticación.
- Terminal interactiva que posibilita la comunicación del servidor con los satélites.
- Actualización remota del firmware del satélite.
- Envió de imágenes satelitales a la estación.
- Obtención de los datos del satélite (telemetría).

Características de los Usuarios

Usuarios con conocimiento básico para manejarse con la consola ya que no se cuenta con una interfaz gráfica.

FUNCIÓN	CLIENTE	SERVIDOR
Autenticación		
Solicitud de usuario y contraseña		✓
Validación de credenciales		✓
Notificación en control de acceso		✓
Inicialización de socket STREAM		✓
Establecer conexión satélite-estación		
Solicitud de conexión con estación terrestre	✓	
Validación de parámetros de conexión	✓	
Inicialización de socket STREAM	✓	
Terminal		
Interactiva		✓
Informativa	✓	✓
Actualización de Firmware		
Ingreso de comando *firmware_update*		✓
Validación del comando	✓	✓
Obtención del tamaño del nuevo Firmware	✓	✓
Envío del archivo de actualización		✓
Recepción del archivo y ajustes de ejecución	✓	
Ejecución del binario recibido	✓	

Obtener datos telemétricos		
Ingreso de comando *obtener_telemetria*		✓
Validación del comando	✓	✓
Inicialización de socket DATAGRAM	✓	✓
Obtención de los datos	✓	
Envío de datos	✓	
Recepción de datos		✓
Cierre de socket DATAGRAM	✓	✓
Iniciar escaneo de superficie terreste		
Ingreso de comando *start_scanning*		✓
Validación del comando	✓	✓
Obtención del tamaño del archivo	✓	✓
Envío del archivo	✓	
Notificación de progreso en la transferencia		✓
Recepción y almacenamiento del archivo		✓
Listado de opciones		
Ingreso de comando *opciones*		✓
Validación del comando		✓
Despliegue de opciones soportadas		✓
Finalización de conexión con satélite		
Ingreso de comando *sat_logoff*		✓
Validación del comando	✓	✓
Cierre de socket STREAM	✓	✓

Restricciones

Las aplicaciones deben compilarse y ejecutarse en sistemas operativos UNIX con permisos para levantar sockets y nuevos procesos.

Se requiere de una terminal con requerimientos de hardware capaz de correr el programa del servidor y de la misma forma, una terminal con requerimientos de hardware, con MMU, que ejecute el programa del cliente.

Suposiciones y Dependencias

- El proyecto se ejecutará sobre UNIX.
- Existe conectividad entre el servidor y el cliente, para poder realizar la comunicación.
- No hay ninguna dependencia específica, el proyecto debe funcionar en una distribución Linux estándar utilizando *make* para compilar el código fuente.
- Tanto el servidor como el cliente tendrán acceso a internet y tendrán una dirección Ipv4.

- El servidor dispondrá de memoria suficiente para almacenar las distintas imágenes satelitales.

Requisitos Futuros

- Soporte para múltiples conexiones satelitales.
- Cifrado de los datos telemétricos.
- Auto diagnóstico del estado del satélite.
- Modo Stand By del satélite.

Requisitos Específicos

Interfaces Externas

- Dispositivo del cliente que debe tener instalado un sistema operativo con kernel linux.
- Dispositivo del servidor que debe tener instalado un sistema operativo con kernel linux.
- Interfaz de usuario, la cual se ofrece al usuario a modo de Prompt en una consola. Simplifica el uso del programa por parte del usuario reduciendo el volumen de errores.
- Sockets empleados para la comunicación entre procesos.

Funciones

Una vez que el usuario se ha autentificado en el servidor, el mismo acepta los siguientes comandos:

- **update_firmware**: envía un archivo binario al satélite con una actualización de software (“firmware”), una vez actualizado, se debe reiniciar el satélite con la nueva actualización.
- **obtener_telemetría**: el satélite le envía a la estación terrena la siguiente información:
  - ID: Nombre del satélite.
  - Uptime: hora y fecha de la última actualización.
  - Versión: versión actual de firmware.
  - CPU: consumo de memoria y CPU.
- **start_scanning**: inicia un escaneo de toda la cara de la Tierra. Cada escaneo se envía a la estación terrena.
- **opciones**: muestra un mensaje de ayuda con los comandos disponibles.
- **sat_logoff:** cierra la sesión actual con el satélite.

Requisitos Funcionales

Conexión	1. La conexión debe ser libre de errores entre el cliente y el servidor para la transferencia de información. 2. Debe notificarse el estado de conexión, direcciones de los extremos del enlace PtP. 3. La inicialización del socket STREAM debe ser posterior a la validación de las credenciales de acceso al servidor. 4. El cliente debe detectar una denegación de conexión al servidor.
Prompt interactivo	1. No es posible ingresar comandos si no hay clientes conectados. 2. El servidor da acceso al prompt solo si se validan las credenciales de acceso. 3. Los comandos se ingresan sólo si termino la ejecución del comando anterior. 4. Deben ignorarse aquellos comandos desconocidos. 5. Debe salir del prompt si pierde conectividad con el cliente.
Autenticación	1. El servidor debe poder verificar al usuario que intenta acceder al servidor en una base de datos. 2. Debe notificarse el estado de autenticación, así como los intentos restantes en el acceso de credenciales. 3. Debe de ocultarse la información durante el ingreso de la contraseña. 4. En caso de fallo de autentificación 3 veces seguidas el servidor debe terminar la aplicación 5. La base de datos debe estar guardada en un archivo de texto plano. 6. No se aceptan ni envían comandos previo a autenticar al usuario.
Actualización del Firmware	1. Solo debe ser aceptado el comando *update_firmware* para activar en ambos extremos de la conexión PtP el procedimiento de actualización. 2. El archivo a enviar es el firmware por TCP. 3. El archivo binario puede reemplazarse respetando formato. 4. El cliente debe recibir el archivo, almacenarlo y reiniciarse lanzando su

	nueva actualización a través del archivo recibido.
Escaneo	1. El único comando aceptado para la ejecución de esta función debe *start_scanning*. 2. En el escaneo se debe enviar un archivo de la forma *nombre.jpg.* 3. El archivo debe transmite por TCP. 4. El la transmisión de la imagen, el paquete en capa 3 del stack TCP/IP no debe fragmentarse. 5. El archivo jpg puede reemplazarse respetando formato.
Obtener telemetría	1. El único comando aceptado para esta función debe *obtener_telemetria*. 2. Los datos deben enviarse mediante un socket UDP. 3. El servidor debe levantar el socket UDP para escuchar al cliente. 4. El comando debe enviar información de: ID satélite, uptime, versión del firmware, uso RAM
Cierre de conexión	1. El único comando aceptado para esta función es *sat_logoff*. 2. El servidor debe cerrar su socket y el programa debe finalizar. 3. El cliente debe cerrar su socket y el programa debe finalizar. 4. El usuario debe recibir un mensaje a través del prompt que finalizó el programa y ya no puede interactuar con el prompt.

Requisitos de Rendimiento

- El proceso de enviar la imagen del satélite a la estación terrena debe ser lo más óptima y en el menor tiempo posible.
- Uso de memoria mínimo y apropiado para que el programa pueda ser corrido en una placa de desarrollo con MMU.

Restricciones de Diseño

- En el servidor no se podrán conectar más de un usuario por vez.
- Tanto el código del cliente como el del servidor serán desarrollados en lenguaje C.
- Se utilizará el puerto 6020 como puerto para la conexión TCP, puerto en el cual se encontrará en todo momento escuchando el servidor una vez realizada la autenticación.
- Se utilizará el puerto 6020 como puerto para la conexión UDP en la transferencia de datos telemétricos.
- El servidor estará montado en una placa de desarrollo que soporte sistemas operativos del tipo GNU/Linux.
- Sintaxis de código del tipo GNU o Linux Kernel.

Atributos del Sistema

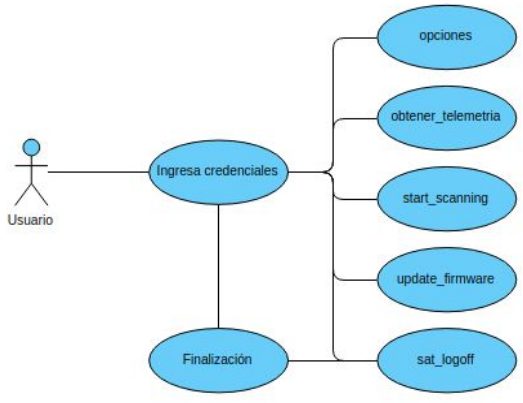
El servidor debe correr en una computadora con distribución Linux, mientras que el cliente (satélite) debe correr sobre una terminal con MMU. A su vez se destaca la portabilidad del sistema, la capacidad de su utilización en sistemas operativos tipo UNIX en equipos de escasos recursos, ideal para sistemas embebidos.

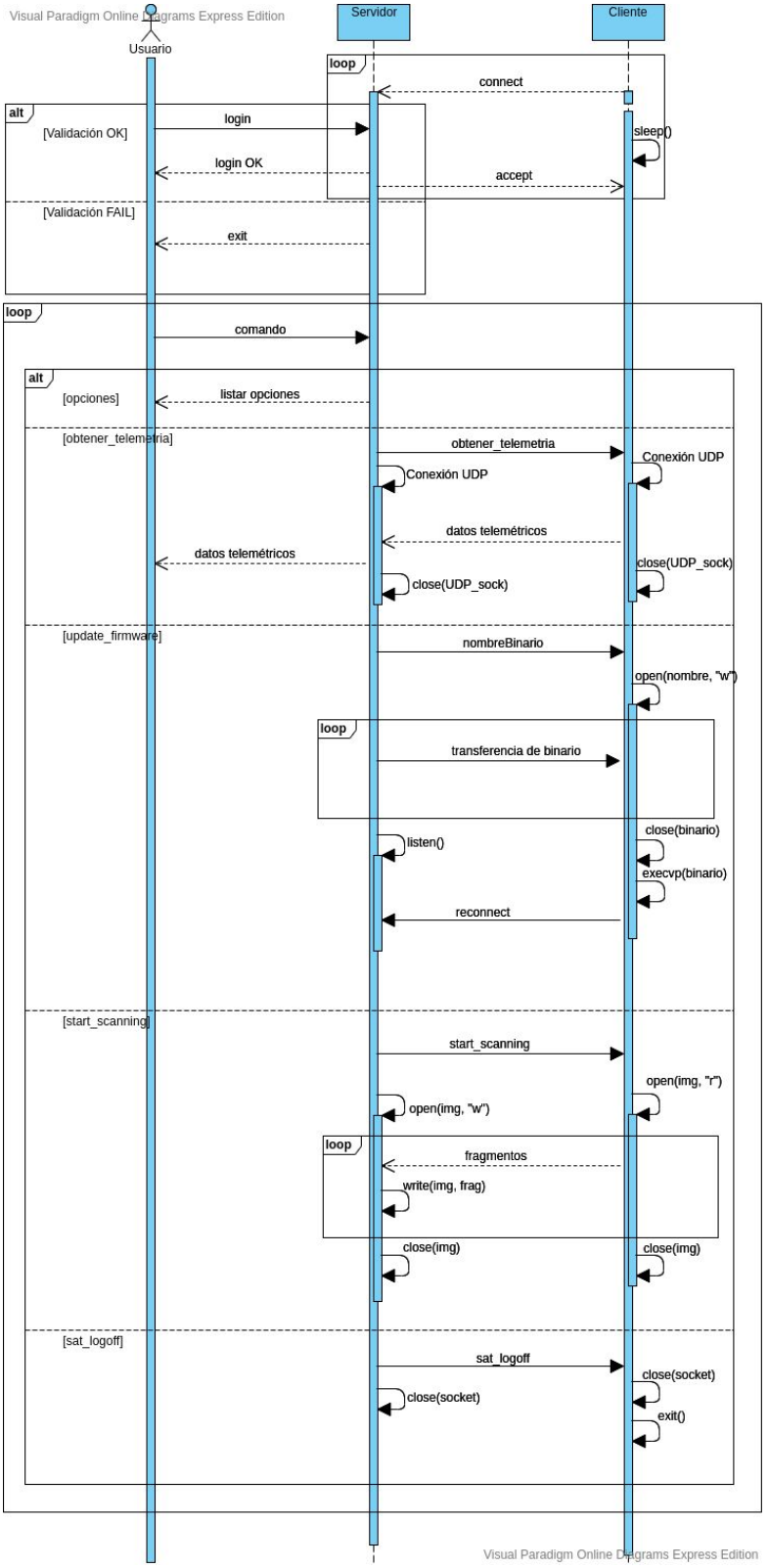
Diseño de solución

Se desarrollaron dos aplicaciones, una que es ejecutada en el cliente y otra en el servidor.

- *Cliente*: se conecta al servidor y queda esperando los comandos que son introducidos desde el servidor para realizar las distintas funciones.
- *Servidor*: crea los sockets para la conexión, autentifica los usuarios que luego podrán pedir un dato específico al cliente mediante el ingreso de comandos disponibles.

El primer paso es el logueo del usuario al sistema. Una vez que se validan sus credenciales, el sistema despliega el prompt en donde se pueden acceder a todas las funciones provistas.




Visual Paradigm Online Diagrams Express Edition
Usuario
Servidor
Cliente
loop
connect
alt
[Validación OK]
login
login OK
accept
sleep()
[Validación FAIL]
exit
loop
comando
alt
[opciones]
listar opciones
[obtener_telemetria]
obtener_telemetria
Conexión UDP
Conexión UDP
datos telemétricos
datos telemétricos
close(UDP_sock)
close(UDP_sock)
[update_firmware]
nombreBinario
open(nombre, "w")
loop
transferencia de binario
close(binario)
listen()
execvp(binario)
reconnect
[start_scanning]
start_scanning
open(img, "r")
open(img, "w")
loop
fragmentos
write(img, frag)
close(img)
close(img)
[sat_logoff]
sat_logoff
close(socket)
close(socket)
exit()
Visual Paradigm Online Diagrams Express Edition

Implementación y Resultados

Para la implementación y desarrollo se utilizó el software *Visual Studio Code* y los códigos sockets aportados por la cátedra. A su vez también se crearon funciones las cuales se utilizaron para dicho proyecto.
Primero se procedió a trabajar en la implementación sobre sockets UNIX. Una vez que el trabajo estuvo implementado en dichos sockets, se comprobó la funcionalidad de los mismos.
Luego, se procedió a implementar el desarrollo sobre sockets INET.

A continuación se muestra el funcionamiento de las implementaciones UNIX y INIT. Las imágenes ubicadas a la izquierda corresponden al comportamiento del cliente, mientras que las ubicadas a la derecha, al servidor.

UNIX

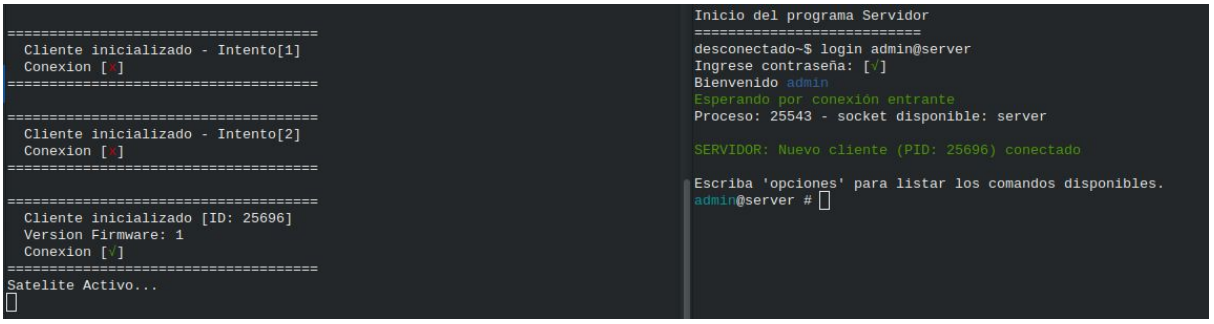
Los programas se ejecutan de la siguiente manera:

- ./cliente <socket UNIX>
- ./servidor

1. El cliente intentará conectarse al servidor de manera indefinida hasta que este último esté disponible.

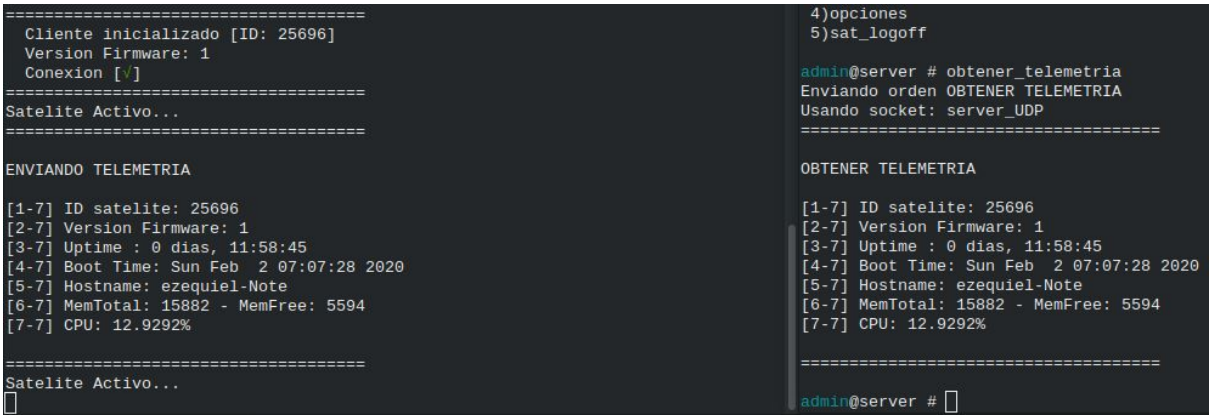
2. El usuario del programa servidor debe presentar credenciales válidas, en caso de no corresponderse con las almacenadas en la base de datos se le denegará el acceso una vez excedida la cantidad de reintentos.

3. Una vez que el usuario presenta credenciales válidas, se despliega el estado de la conexión y un prompt para acceder a las opciones del sistema.

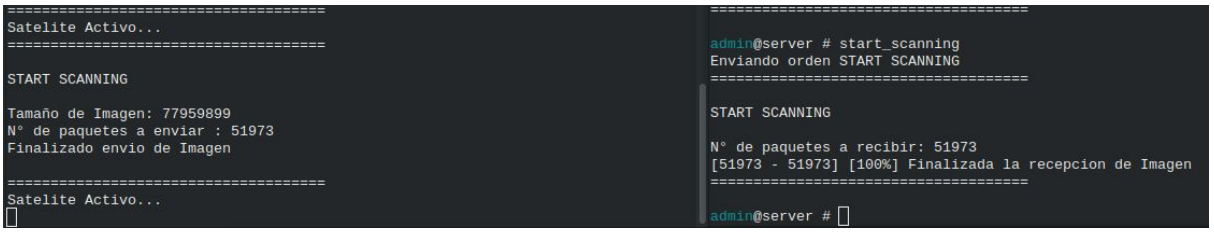


4. El comando *opciones* lista los comandos disponibles para la interacción con el satélite.

5. El comando *obtener_telemetria* envía los datos del satélite hacia la estación terrestre.



6. Ejecutando el comando *start_scanning* se transfiera la imagen satelital a la estación terrestre. Se muestra información del tamaño en bytes de la imagen, la cantidad de paquetes a enviar y el porcentaje de avance durante la transmisión.



7. La actualización del firmware del satélite se realiza mediante el comando *update_firmware*. Una vez que el satélite recibe el nuevo binario, procede a reiniciarse cargando el nuevo firmware y re-enlazarse a la estación terrestre.

8. La desconexión de la estación con el satélite se realiza mediante *sat_logoff*. Una vez cerrada en ambos extremos la conexión, la estación queda a la espera de nuevas conexiones entrantes.

INET

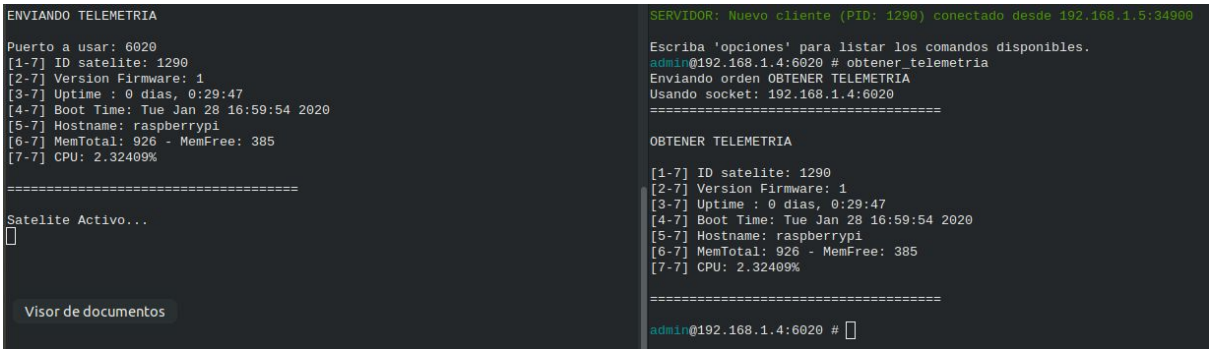
Los programas se ejecutan de la siguiente manera:

- ./cliente <IP servidor>:<Puerto servidor>
- ./servidor

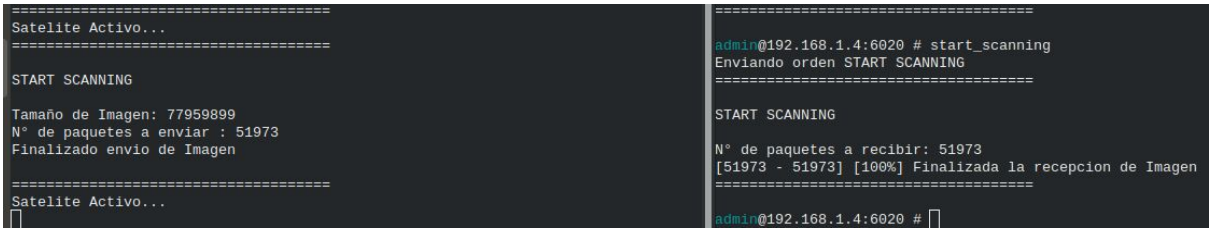
Las características son idénticas a las desarrolladas en socket UNIX.

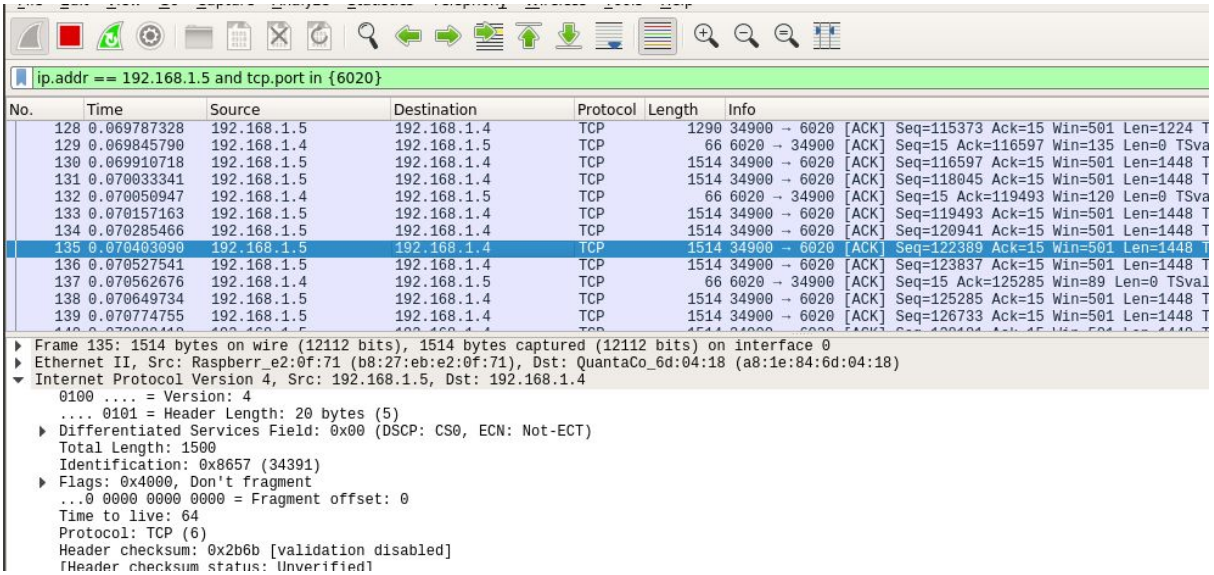
1. El proceso de validación de credenciales y conexión con el servidor son identicas que en socket UNIX. La diferencia radica el parámetro pasado para la conexión, siendo en este caso la dirección IP y puesto del programa servidor.

2. Ejecución del comando *obtener_telemetria*.

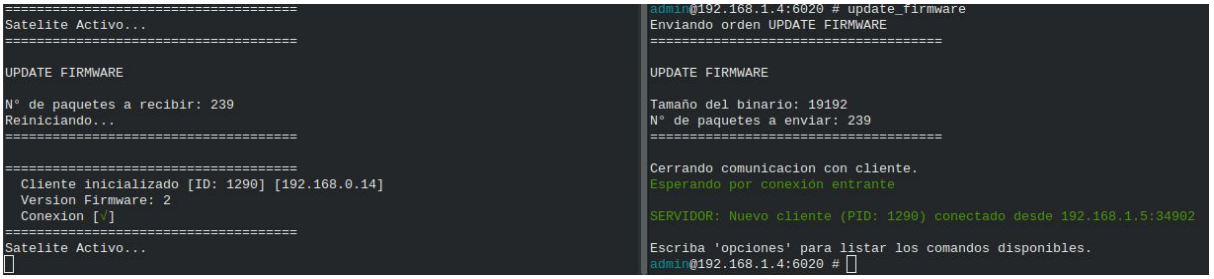


3. Ejecución del comando *start_scanning*. Se observa que los paquetes transmitidos no son fragmentados, cumpliendo con uno de los requerimientos.

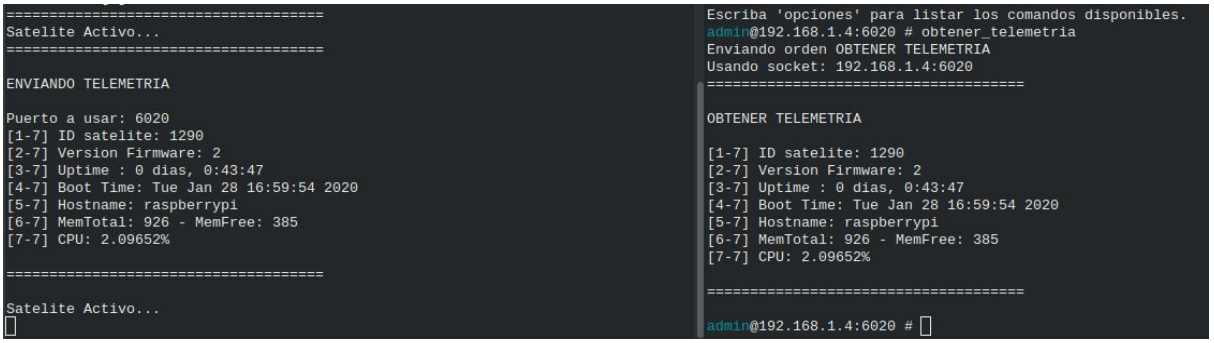




4. Ejecución del comando *update_firmware*.



5. Ejecución del comando *obtener_telemetria* posterior a la actualización del firmware.



6. Ejecución del comando *sat_logoff*.

Conclusiones

Por medio de la resolución del trabajo práctico se logró llevar a cabo una conexión de dos terminales distintas por medio de sockets. Logrando así llevar a cabo un proyecto de la vida real. Se puso en práctica una forma de IPC, tanto para comunicación de procesos por Internet como a nivel local.

Además, se profundizaron conocimiento de programación en C, uso de punteros, acceso a archivos entre otros.