

# Trabajo practico N° 1: Sockets de internet en sistemas tipo UNIX

CÁTEDRA DE SISTEMAS OPERATIVOS II

CASABELLA MARTIN, 39694763  
martin.casabella@alumnos.unc.edu.ar

14 de abril de 2019

# 1. Introducción

Los sockets son una abstracción de comunicación entre procesos (*IPC*) que en un sistema de 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.

Son una herramienta muy importante, uno de los pilares de comunicación entre procesos, y además, sumamente utilizada en la mayoría de las aplicaciones de red.

# 2. Propósito

En este trabajo se pretende entender el funcionamiento de esta abstracción, donde por un lado se trabaja con la familia de sockets UNIX(AF\_UNIX), y por el otro, la familia de sockets TCP/IP (AF\_INET).

Se utilizarán para simular la conexión de una estación terrena que se comunica con un satélite geostacionario. La estación terrena enviará comandos al satélite y este devolverá una respuesta y/o realizará cierta acción

# 3. Ámbito del sistema

Se requiere realizar IPC mediante sockets orientados a conexión y no orientados a conexión, de ambas familias, UNIX y TCP/IP.

Ambos sistemas trabajan con sistemas tipo UNIX. La estación terrena (servidor) estará a la escucha en un puerto TCP fijo. Antes de levantar la comunicación con el satélite, el usuario debe autenticarse con usuario y contraseña localmente, con un número máximo de intentos. Caso contrario, no se provee mecanismo de comunicación ya que el usuario no está autorizado a interactuar con el satélite.

Si el usuario se autentica de forma correcta, el satélite (cliente), deberá poder conectarse al servidor y responder a los comandos ingresados por un prompt desde el servidor.

# 4. Definiciones, Acrónimos y Abreviaturas

## 4.1. Protocolo TCP

Muchos programas dentro de una red de datos compuesta por redes de computadoras, pueden usar TCP para crear “conexiones” entre sí a través de las cuales puede enviarse un flujo de datos. El protocolo garantiza que los datos serán entregados en su destino sin errores y en el mismo orden en que se transmitieron. También proporciona un mecanismo para distinguir distintas aplicaciones dentro de una misma máquina, a través del concepto de puerto.

## 4.2. Protocolo UDP

El protocolo de datagramas de usuario es un protocolo del nivel de transporte basado en el intercambio de datagramas. Permite el envío de datagramas a través de la red sin que se haya establecido previamente una conexión, ya que el propio datagrama incorpora suficiente información de direccionamiento en su cabecera. Tampoco tiene confirmación ni control de flujo, por lo que los

paquetes pueden adelantarse unos a otros; y tampoco se sabe si ha llegado correctamente, ya que no hay confirmación de entrega o recepción.

### 4.3. Dirección IP

La dirección IP es un número que identifica, de manera lógica y jerárquica, a una Interfaz en red (elemento de comunicación/conexión) de un dispositivo (computadora, tableta, portátil, smartphone) que utilice el protocolo IP o (Internet Protocol), que corresponde al nivel de red del modelo TCP/IP.

### 4.4. Puerto

Un puerto es el valor que se usa, en el modelo de la capa de transporte, para distinguir entre las múltiples aplicaciones que se pueden conectar al mismo host, o puesto de trabajo.

### 4.5. Sockets IPC

Socket es una forma de IPC (InterProcess Communication) introducida por la Universidad de Berkeley en su versión de Unix (BSD). El concepto de socket nos evita tener que aprender una interfaz de programación diferente para cada protocolo.

La comunicación por sockets sigue el modelo cliente-servidor. El cliente debe conocer la dirección del servidor para la comunicación, pero el servidor no conoce la existencia del cliente.

Una vez establecido el contacto ambas partes pueden enviar y recibir datos, no se reconoce un maestro o esclavo.

Los procesos generan sockets que son la abstracción utilizada para la comunicación. A los sockets que van a prestar un servicio se les debe asociar una dirección. El cliente utilizando su socket, contacta al servidor (del que debe conocer la dirección) para transferir datos.

### 4.6. Cliente

El cliente es una aplicación informática o un ordenador que consume un servicio remoto en otro ordenador conocido como servidor, normalmente a través de una red de telecomunicaciones.

### 4.7. Servidor

un servidor basado en software es un programa que ofrece un servicio especial que otros programas denominados clientes pueden usar a nivel local o a través de una red. La base de la comunicación es el modelo cliente-servidor y, en lo que concierne al intercambio de datos, entran en acción los protocolos de transmisión específicos del servicio.

### 4.8. Prompt

Se llama prompt al conjunto de caracteres que se muestran en una línea de comandos para indicar que está a la espera de órdenes. Éste puede variar dependiendo del intérprete de comandos.

## 5. Descripción general del documento

Este documento esta organizado en 6 secciones principales:

- Describe las generalidades del documento y del proyecto
- Describe la perspectiva del producto, funciones del mismo, características de usuario y restricciones
- Referencia a los requerimientos del producto, interfaces y casos de uso. Incluye diagramas y gráficos que facilitan su entendimiento
- Abarca un diseño de la solución
- La última parte incluye la implementación de la solución y resultados
- Por último, se encuentran las conclusiones

## 6. Descripción general

### 6.1. Perspectiva del Producto

Como objetivo esta el diseño e implementación de un software desarrollado en lenguaje C basado acatando el modelo cliente/servidor, permitiendo la comunicacion entre ambas partes a través de sockets de internet y tipo UNIX.

El servidor proveerá al usuario un medio para ejecutar acciones definidas y obtener información del cliente remoto a través de un prompt.

### 6.2. Funciones del Producto

#### 6.2.1. Conexión

Permite establecer una conexión mediante socket TCP, desde el cliente al servidor en la estación terrena. A su vez, puede emplearse el mismo software pero utilizando la familia de sockets UNIX.

#### 6.2.2. Autenticación

Característica del servidor, que verificará la existencia de login (usuario, password) y dará o no autorización al shell interactivo. De haber sido autenticado correctamente, se levanta la comunicacion con el satélite.

#### 6.2.3. Prompt interactivo

Luego de una autenticación el servidor esta a la espera de clientes, cuando estos se conectan provera el prompt al usuario.

#### 6.2.4. Telemetría

El cliente obtiene y envia ciertos datos de telemetría definidos y los envía al servidor.

#### 6.2.5. Update firmware

El servidor puede enviar un nuevo firmware destinado a ser cargado en el cliente, que se reiniciara levantando el nuevo archivo binario (firmware).

### 6.2.6. Start scanning

El satélite puede iniciar el escaneo de la cara de la tierra y enviarla al servidor en formato de archivo.

## 6.3. Características de los Usuarios

Autenticacion		
Funcion	Cliente	Servidor
Solicitar usuario		x
Solicitar contraseña		x
Validar ingreso en la base de datos		x
Enviar mensaje en funcion al resultado		x
Levantar socket en caso de logeo exitoso		x
No permitir la conexion en caso fallido		

Prompt interactivo		
Funcion	Cliente	Servidor
Solicitar prompt		x
Autenticar usuario		x
Levantar socket TCP		x
Codificación y envío de comandos	x	x
Ejecución de comandos localmente	x	
envío y recepción de datos full duplex	x	x

Update firmware		
Función	Cliente	Servidor
Ingreso comando firmware update		x
Validación comando firmware update		x
Envio de archivo binario		x
Chequeo de existencia del archivo		x
Informe de errores emergentes (de haber)		x
Recibir archivo binario (firmware) y lanzarlo	x	

Start scanning		
Función	Cliente	Servidor
Ingreso comando start scanning		x
Validación comando start scanning		x
Obtención del archivo(escaneo)	x	
Informe de errores emergentes (de haber)	x	
Envio de archivo .jpg	x	
Recepcion y almacenamiento del archivo		x

Telemetry		
Función	Cliente	Servidor
Ingreso comando get telemetry		x
Levantar socket UDP a la espera de resultado de ejecución		x
Levantar socket UDP para comunicación y envío	x	
Recolección de datos	x	
Envío de datos mediante socket UDP	x	
Interpretación y muestra de resultado de ejecución		x

## 6.4. Restricciones

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

Si se utiliza el software con la familia de sockets UNIX, al ingresar el comando **update firmware**, se deberá volver a correr el programa.

Deben respetarse los nombres de los directorios, y de no existir, se deben crear en sus respectivos path previo a compilar y ejecutar el programa.

## 6.5. Suposiciones y Dependencias

Ambos sistemas deben representar de la misma forma los número Little Endian o Big Endian. Ambos host deben ser directamente alcanzables por ethernet.

Para la compilación es totalmente necesario tener instalados las librerías *GNU Glibc* >= 2,2, para la estandarización de tipos.

# 7. Requisitos específicos

Se detallan a continuación los requisitos específicos del sistema, especificados por el cliente.

## 7.1. Interfaces externas

En esta sección se especifican los requisitos del proceso de desarrollo de software, como la interfaz de la aplicación, la hardware y entorno del sistema. La interfaz de usuario del cliente y del servidor

serán por cualquier TTY, como por ejemplo una consola interactiva como bash o xterm,. Para el desarrollo del software se utiliza utiliza GNU/Linux, herramientas de compilación y librerías en su última versión estable, específicamente las versiones que cumplen los criterios son las rolling release como ArchLinux, Antergos, Manjaro y derivados.

- Kernel  $\geq$  5.0.7 ( 5.0.7-1-ARCH aarch64 GNU/Linux)
- GCC  $\geq$  version 8.2 (Provee mejor soporte gnu11/c11)
- GLibC  $\geq$  2.26 (Provee soporte de unificación/comunicacion de tipos para diferentes arquitecturas)
- Make  $\geq$  4.2.1
- libnet  $\geq$  1.1.6-3

## 7.2. Requisitos funcionales

### 7.2.1. Requisitos de conexión

1. Debe poder establecerse una conexión libre de errores entre los clientes y el servidor para la transferencia de información.
2. En caso de imposibilidad de conexión debe ser notificado
3. El servidor debe poder ingresar ingresa usuario y contraseña para solicitar acceso al sistema
4. Luego de autenticar se levanta el servidor
5. No se aceptan ni envían comandos previo a autenticar al usuario.
6. El cliente debe ser capaz de detectar una caída de conexión o un rechazo de la misma por errores de la red o por fallo en la autenticación.

### 7.2.2. Requisitos de conexión

1. Debe poder establecerse una conexión libre de errores entre los clientes y el servidor para la transferencia de información.
2. En caso de imposibilidad de conexión debe ser notificado
3. El servidor debe poder ingresar ingresa usuario y contraseña para solicitar acceso al sistema
4. Luego de autenticar se levanta el servidor
5. No se aceptan ni envían comandos previo a autenticar al usuario.
6. El cliente debe ser capaz de detectar una caída de conexión o un rechazo de la misma por errores de la red o por fallo en la autenticación.

### 7.2.3. Requisitos de prompt

1. No es posible ingresar comandos si no hay clientes conectados
2. El servidor provee prompt solo si se autenticar correctamente
3. Los comandos se ingresan solo si termino la ejecución del comando anterior, ya sea correctamente o si surgió algún error.
4. Debe identificar si el comando no existe
5. Debe salir del prompt si pierde conectividad con el cliente

#### 7.2.4. Requisitos autenticacion

1. El servidor debe poder verificar al usuario que intenta solicitar un prompt en una base de datos
2. Los usuarios o contraseñas no superaran los 20 caracteres
3. La base de datos debe estar guardada en un archivo de texto plano
4. En caso de fallo de autenticación 3 veces seguidas el servidor debe terminar la aplicación
5. No se aceptan ni envían comandos previo a autenticar al usuario.
6. El cliente debe ser capaz de detectar una caída de conexion o un rechazo de la misma por errores de la red o por fallo en la autenticación.

#### 7.2.5. Requisitos de Update Firmware

1. El único comando aceptado para esta función es "update firmware"
2. El archivo a enviar es el firmware por TCP
3. El archivo a enviar debe situarse en el directorio cuyo nombre es firmware dentro del directorio del servidor.
4. El archivo binario puede reemplazarse respetando formato
5. El cliente debe recibir el archivo, guardarlo cerrarse y lanzar su nueva actualización por el archivo recibido

#### 7.2.6. Requisitos de Start Scanning

1. El único comando aceptado para esta función es "start scanning"
2. En el escaneo enviar el archivo con el nombre ".aerial\_earth\_image.jpg"
3. El archivo se transmite por TCP
4. El archivo debe situarse en el directorio nombrado img dentro del directorio server
5. El archivo jpg puede reemplazarse respetando formato

#### 7.2.7. Requisitos de telemetría

1. El único comando aceptado para esta función es "get telemetry"
2. El comando se envia por TCP pero su respuesta es por UDP
3. El servidor debe levantar el socket para escuchar al cliente
4. El comando debe enviar informacion de: ID satellite, uptime, version del firmware, uso RAM



#### 7.2.8. Requisitos de shutdown

1. El único comando aceptado para esta función es “ shutdown now”
2. El comando se envia por TCP
3. El servidor debe cerrar su socket y el programa debe finalizar
4. El cliente debe cerrar su socket y el programa debe finalizar
5. El usuario recibe un mensaje a traves del prompt que finalizo el programa y ya no puede interactuar con el prompt

### 7.3. Requisitos de rendimiento

El sistema debe ser capaz de correr en un SoC (System On Chip) o una computadora personal (x86/AMD64), con al menos 1[GB] de RAM y 1[GB] de espacio libre, un procesador con una frecuencia de 1[GHz] contando que solo compite en los recursos con los procesos del sistema operativo base. El sistema operativo base puede ser GNU/Linux o derivados de BSD.

### 7.4. Restricciones de diseño

No se utilizaron restricciones de diseño en el desarrollo del sistema.

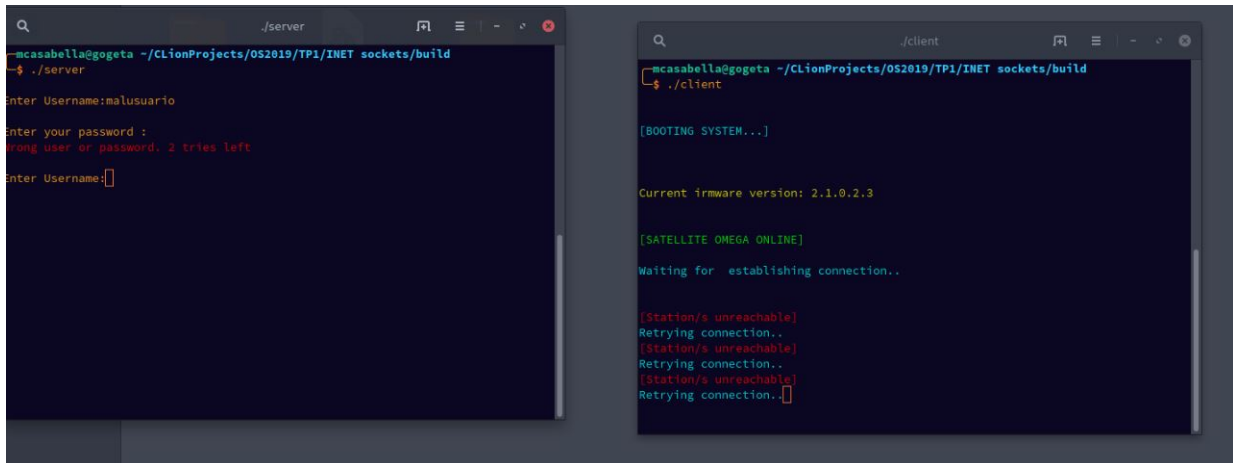
### 7.5. Atributos del Sistema

El atributo más importante del sistema es la portabilidad, la capacidad de la utilización de sistemas operativos tipo UNIX y los escasos recursos necesarios para correr el software lo hacen ideal para pequeños sistemas embebidos de bajos recursos

## 8. Diseño de la solución

La solución está documentada en el repositorio del proyecto. [Repositorio del Proyecto]

## 9. Implementacion y resultados



```
./server
Enter Username: malusuario
Enter your password:
Wrong user or password. 2 tries left
Enter Username:

./client
[BOOTING SYSTEM...]

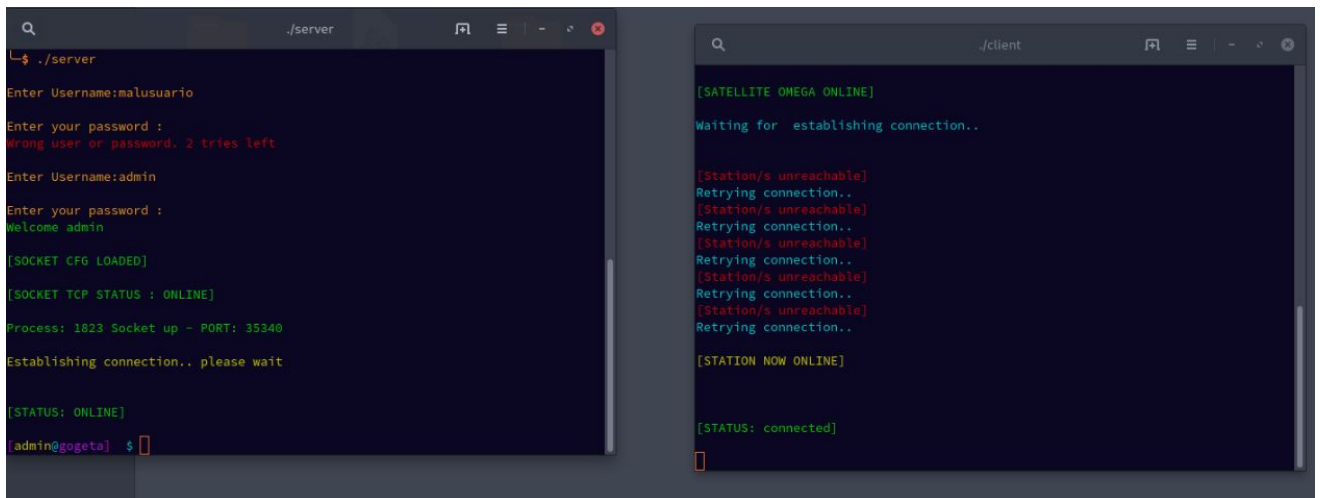
Current firmware version: 2.1.0.2.3

[SATELLITE OMEGA ONLINE]

Waiting for establishing connection..

[Station/s unreachable]
Retrying connection..
[Station/s unreachable]
Retrying connection..
[Station/s unreachable]
Retrying connection..
```

Figura 1: Mal ingreso de usuario



```
./server
Enter Username: admin
Enter your password:
Welcome admin

[SOCKET CFG LOADED]

[SOCKET TCP STATUS : ONLINE]

Process: 1823 Socket up - PORT: 35340

Establishing connection.. please wait

[STATUS: ONLINE]

[admin@gogeta] $

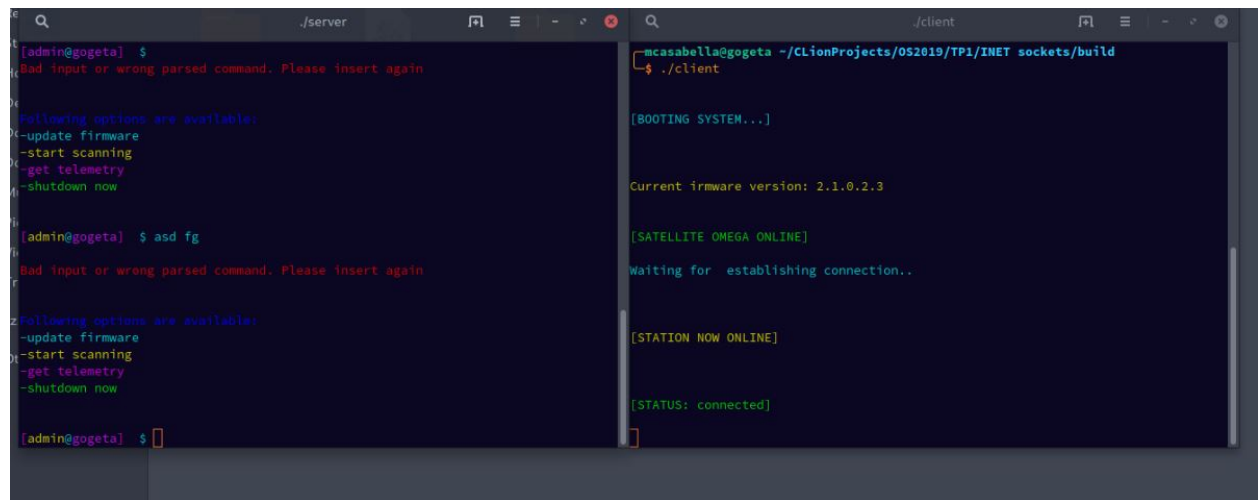
./client
[SATELLITE OMEGA ONLINE]

Waiting for establishing connection..

[Station/s unreachable]
Retrying connection..
[Station/s unreachable]
Retrying connection..
[Station/s unreachable]
Retrying connection..
[Station/s unreachable]
Retrying connection..
[STATION NOW ONLINE]

[STATUS: connected]
```

Figura 2: Ingreso correcto de usuario

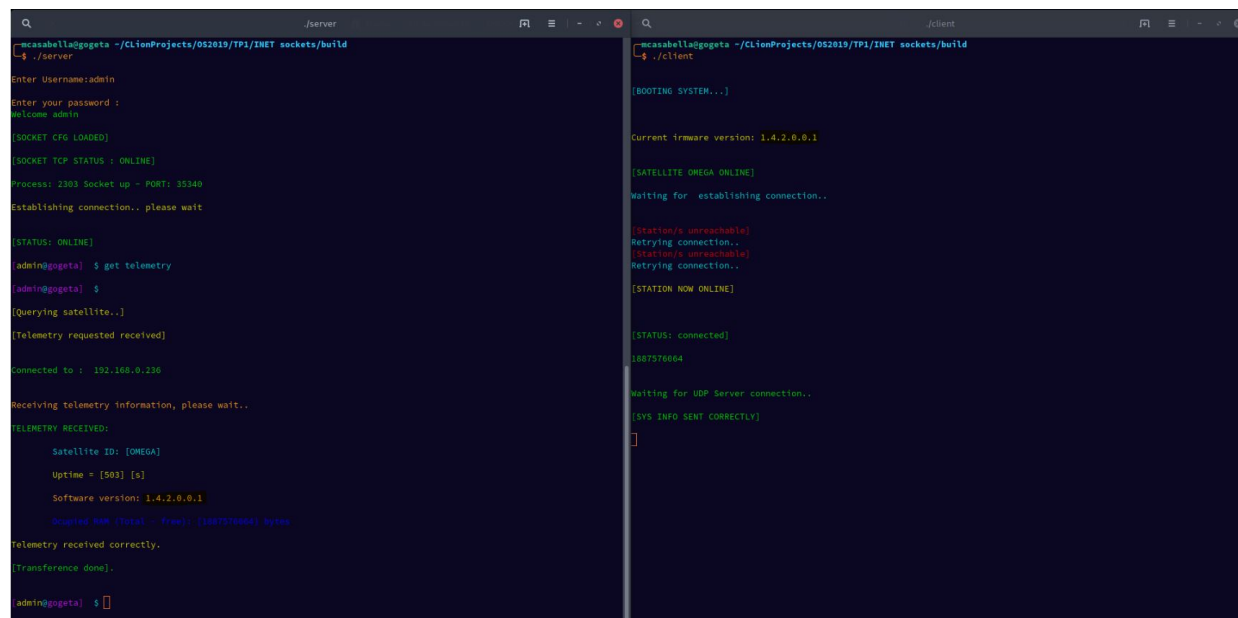


```
[admin@gogeta] $  
Bad input or wrong parsed command. Please insert again  
  
Following options are available:  
-update firmware  
-start scanning  
-get telemetry  
-shutdown now  
  
[admin@gogeta] $ asd fg  
Bad input or wrong parsed command. Please insert again  
  
Following options are available:  
-update firmware  
-start scanning  
-get telemetry  
-shutdown now  
  
[admin@gogeta] $
```

```
mcasabella@gogeta ~/CLionProjects/052019/TP1/INET sockets/build  
$ ./client  
  
[BOOTING SYSTEM...]  
  
Current firmware version: 2.1.0.2.3  
  
[SATELLITE OMEGA ONLINE]  
Waiting for establishing connection..  
  
[STATION NOW ONLINE]  
  
[STATUS: connected]
```

Figura 3: Ingreso de comando erroneo

Se cita la siguiente secuencia de comandos: primero se ingresa **get telemetry**, para obtener información del satellite. Se resalta ahí la versión del firmware actual:

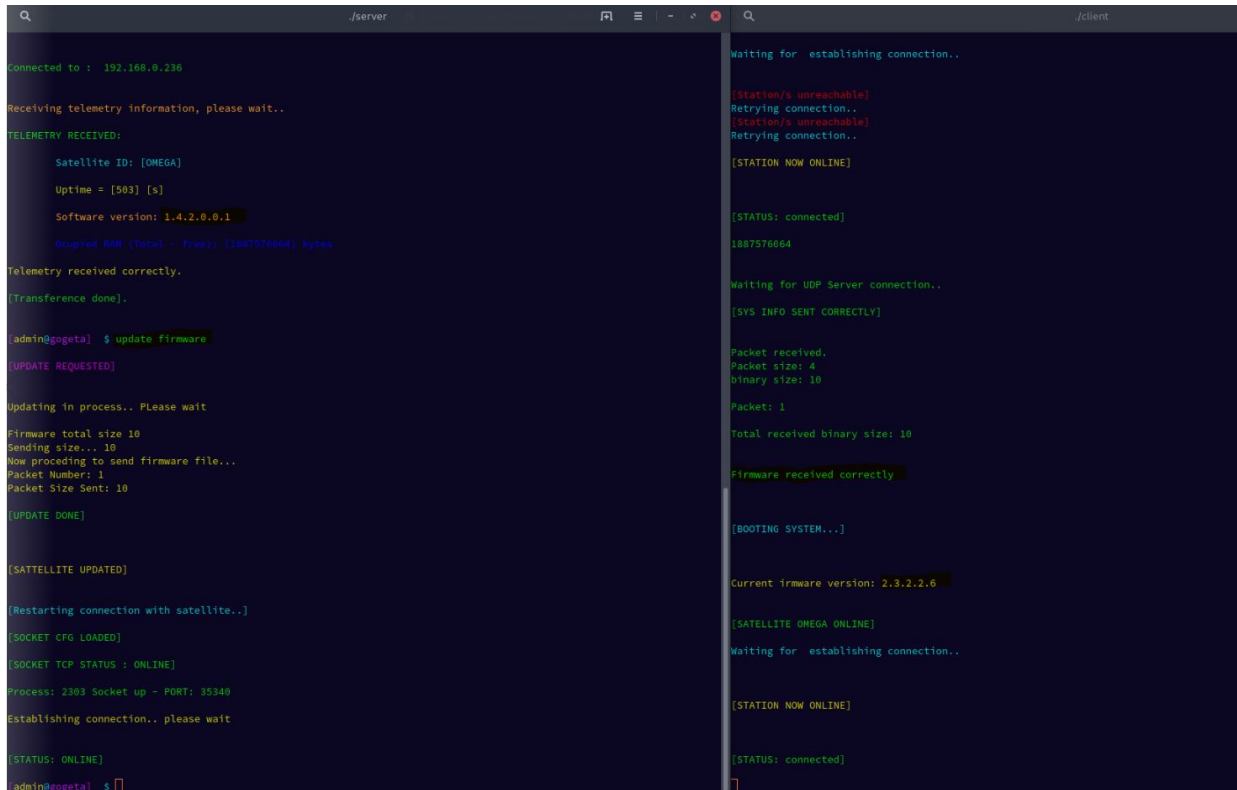


```
mcasabella@gogeta ~/CLionProjects/052019/TP1/INET sockets/build  
$ ./server  
  
Enter Username:admin  
Enter your password:  
Welcome admin  
  
[SOCKET CFG LOADED]  
[SOCKET TCP STATUS : ONLINE]  
Process: 2303 Socket up ~ PORT: 35340  
Establishing connection.. please wait  
  
[STATUS: ONLINE]  
  
[admin@gogeta] $ get telemetry  
[admin@gogeta] $  
  
[Querying satellite..]  
[Telemetry requested received]  
  
Connected to : 192.168.0.236  
  
Receiving telemetry information, please wait..  
TELEMETRY RECEIVED:  
  
Satellite ID: [OMEGA]  
Uptime = [503] [s]  
Software version: 1.4.2.0.0.1  
Supplied ROM [Total - Free]: [1687576664] bytes  
Telemetry received correctly.  
[Transference done].  
  
[admin@gogeta] $
```

```
mcasabella@gogeta ~/CLionProjects/052019/TP1/INET sockets/build  
$ ./client  
  
[BOOTING SYSTEM...]  
  
Current firmware version: 1.4.2.0.0.1  
  
[SATELLITE OMEGA ONLINE]  
Waiting for establishing connection..  
  
[station's unreachable]  
Retrying connection..  
[station's unreachable]  
Retrying connection..  
[STATION NOW ONLINE]  
  
[STATUS: connected]  
1687576664  
  
Waiting for UDP Server connection..  
[SYS INFO SENT CORRECTLY]  
  
]
```

Figura 4: Resultado de get telemetry

Ahora se ingresa **update firmware**, para forzar al satélite a actualizarse:



```

./server
Connected to : 192.168.0.236

Receiving telemetry information, please wait..

TELEMETRY RECEIVED:

    Satellite ID: [OMEGA]

    Uptime = [503] [s]

    Software version: 1.4.2.0.0.1

    Occupied RAM (Total - Free): [1887570664] bytes

Telemetry received correctly.

[Transference done].

admin@pageta] $ update firmware

[UPDATE REQUESTED]

Updating in process.. Please wait

Firmware total size 10
Sending size... 10
Now proceeding to send firmware file...
Packet Numbers: 1
Packet Size Sent: 10

[UPDATE DONE]

[Satellite updated]

[Restarting connection with satellite..]

[Socket CPG loaded]

[Socket TCP status : ONLINE]

Process: 2303 Socket up - PORT: 35340

Establishing connection.. please wait

[STATUS: ONLINE]

admin@pageta] $

./client
Waiting for establishing connection..

[Station/s unreachable]
Retrying connection..
[Station/s unreachable]
Retrying connection..

[STATION NOW ONLINE]

[STATUS: connected]
1887570664

Waiting for UDP Server connection..

[SYS INFO SENT CORRECTLY]

Packet received.
Packet size: 4
Binary size: 10

Packet: 1

Total received binary size: 10

Firmware received correctly

[BOOTING SYSTEM...]

Current firmware version: 2.3.2.2.6

[SATELLITE OMEGA ONLINE]

Waiting for establishing connection..

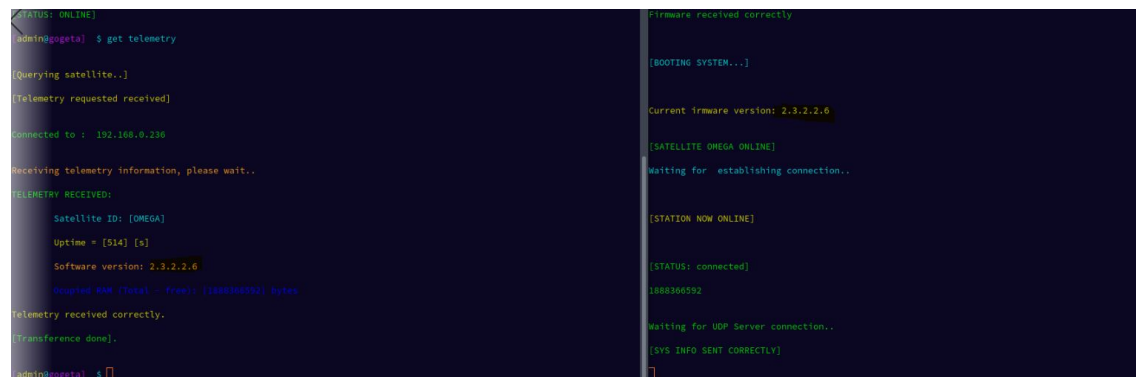
[STATION NOW ONLINE]

[STATUS: connected]

```

Figura 5: Actualización de firmware

Y finalmente, se vuelve a ingresar **get telemetry** para corroborar si el satélite se actualizó:



```

[STATUS: ONLINE]

admin@pageta] $ get telemetry

[Querying satellite..]

[Telemetry requested received]

Connected to : 192.168.0.236

Receiving telemetry information, please wait..

TELEMETRY RECEIVED:

    Satellite ID: [OMEGA]

    Uptime = [514] [s]

    Software version: 2.3.2.2.6

    Occupied RAM (Total - Free): [1883963592] bytes

Telemetry received correctly.

[Transference done].

admin@pageta] $

Firmware received correctly

[BOOTING SYSTEM...]

Current firmware version: 2.3.2.2.6

[SATELLITE OMEGA ONLINE]

Waiting for establishing connection..

[STATION NOW ONLINE]

[STATUS: connected]
1883963592

Waiting for UDP Server connection..

[SYS INFO SENT CORRECTLY]

```

Figura 6: Resultado luego de actualizar el satélite

Se anexan los restantes comandos:

```

[admin@gogeta] $ start scanning

[Querying SCANNING...]

[Starting scanning...]

Number of bytes to receive: 77959899d

[RECEPTION FINISHED]

Total transference time: 0.141266

[SCANNED IMAGE RECEIVED CORRECTLY]
You can find it here: ../src/server/img/earth_aerial_img_rcv.jpg

[admin@gogeta] $

[STATION NOW ONLINE]

[STATUS: connected]

1672630272

Waiting for UDP Server connection..

[SYS INFO SENT CORRECTLY]

Filesize: 77959899

[IMAGE SENT CORRECTLY]

```

Figura 7: Resultado de start scanning

```

[STATUS: ONLINE]

[admin@gogeta] $ get telemetry

[Querying satellite..]

[Telemetry requested received]

Connected to : 192.168.0.236

Receiving telemetry information, please wait..

TELEMETRY RECEIVED:

  Satellite ID: [OMEGA]

  Uptime = [514] [s]

  Software version: 2.3.2.2.6

  Occupied RAM (Total - Free): [144800502] Bytes

Telemetry received correctly.

[Transference done].

[admin@gogeta] $ shutdown now

[SYSTEM OFF]

mcasabella@gogeta ~/CLionProjects/052019/TP1/INET sockets/build
$

Current firmware version: 2.3.2.2.6

[SATELLITE OMEGA ONLINE]

Waiting for establishing connection..

[STATION NOW ONLINE]

[STATUS: connected]

1888366592

Waiting for UDP Server connection..

[SYS INFO SENT CORRECTLY]

[Closing connection..]

[CHANNEL OFF]

Closing sockets..

[SATELLITE OFFLINE]

Disconnecting..

mcasabella@gogeta ~/CLionProjects/052019/TP1/INET sockets/build
$

```

Figura 8: Apagado de sistema

## 10. Conclusiones

Se puso en práctica una forma de IPC, tanto para comunicación de procesos por Internet, a nivel local. Se maduraron conceptos relacionados al protocolo, las formas de comunicación (bloqueante, o no bloqueante) y se incorporó la transferencia de información con los mecanismos de abstracción mencionados.