



## Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorios de docencia



# Laboratorio de Redes y Seguridad

*Profesor:* Ing. Magdalena Reyes Granados

*Asignatura:* Lab. Redes de Datos Seguras

*Grupo:* 08

*No de Práctica(s):* 08

*Integrante(s):* Martínez Rojas José Eduardo

Mateos Flores Erik Esteban


*No. de Equipo de  
cómputo empleado:*

*Semestre:* 2022-1

*Fecha de entrega:* 25/10/2021

*Observaciones:*


**CALIFICACIÓN:** \_\_\_\_\_

	<b>Manual de prácticas del Laboratorio de Redes de Datos Seguras</b>	Código:	MADO-31
		Versión:	04
		Página	106/297
		Sección ISO	8.3
		Fecha de emisión	17 de agosto de 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de Redes y Seguridad	
La impresión de este documento es una copia no controlada			

## Práctica 8

### TCP Y UDP

#### Capa 4 del Modelo OSI

	<b>Manual de prácticas del Laboratorio de Redes de Datos Seguras</b>	Código:	MADO-31
		Versión:	04
		Página	107/297
		Sección ISO	8.3
		Fecha de emisión	17 de agosto de 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de Redes y Seguridad	
La impresión de este documento es una copia no controlada			

### 1.- Objetivos de Aprendizaje

- El alumno podrá utilizar un programa que le permita enviar y recibir información utilizando los protocolos TCP y UDP y reafirmando conceptos teóricos.
- El alumno creará un socket servidor y un socket cliente

### 2.- Conceptos teóricos

#### El programa Sock

El programa sock ofrece un modo de acceder a la interfaz de los sockets sin tener que programar. Conecta la entrada/salida estándar (teclado/pantalla) con un socket cuyas características se especifican mediante parámetros al ejecutar la orden. Mediante la redirección de la entrada o la salida se puede enviar el contenido de un archivo o almacenar en un archivo la información recibida.

Los sockets pueden ser de dos tipos: UDP o TCP, que se corresponden con un servicio sin conexión, que no garantiza ni la entrega ni el orden de entrega de la información (UDP) y otro servicio que garantiza la entrega ordenada y sin errores de la información (TCP).

Además, se sabe que una aplicación puede comenzar iniciando la comunicación (enviando información) o bien puede esperar pacientemente hasta que la otra le solicite el inicio de la comunicación (espera petición).

El programa sock va a permitir imitar cualquiera de estas situaciones entre otras.

### 3.- Equipo y material necesario

#### 3.1 Material del alumno:

- Imagen extensión BMP con calidad de una imagen fotográfica.

#### 3.2 Equipo del Laboratorio:


- Programa sock (sock-1.1.tar.tar).

### 4.- Desarrollo:

#### Modo de trabajar

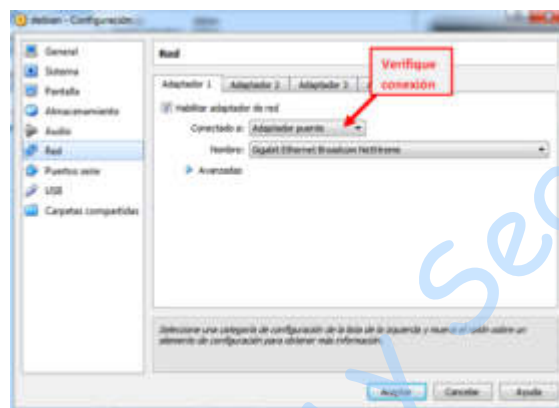
- La práctica se desarrollará en parejas.

#### 4.1 Preparación del programa Sock

	<b>Manual de prácticas del Laboratorio de Redes de Datos Seguras</b>	Código:	MADO-31
		Versión:	04
		Página	108/297
		Sección ISO	8.3
		Fecha de emisión	17 de agosto de 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de Redes y Seguridad	
La impresión de este documento es una copia no controlada			

#### 4.1.1 Abra la aplicación VirtualBox

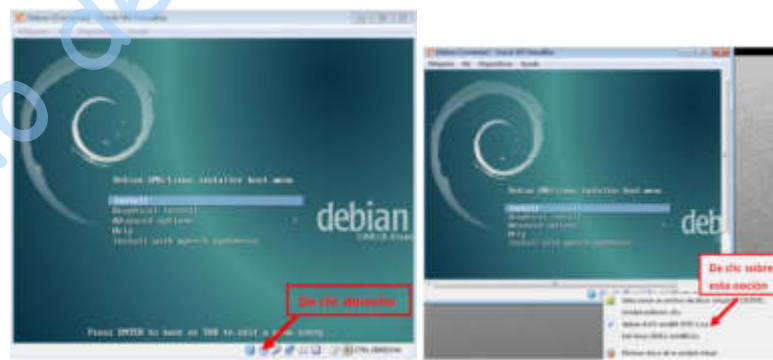
**NOTA:** Antes de iniciar la máquina virtual verifique en la opción Red que se encuentre marcada la opción Habilitar adaptador de red->Conectado a: Adaptador puente (Figura No. 1)



**Figura No. 1. Conexión de red.**

#### 4.1.2 Elija la opción de cargar Linux, distribución Debian.


**NOTA:** En caso de que le aparezca la imagen de instalación (Figura No. 2), dé clic derecho sobre el disco duro. Seleccione la opción que se encuentra palomeada para deseccionarla, apague la máquina virtual y vuelva a iniciarla.



**Figura No. 2. Inicio de Máquina Virtual.**

#### 4.1.3 Inicie sesión como usuario **redes**.

#### 4.1.4 Abra una terminal e ingrese como super usuario, para ello teclee el comando que se muestra a continuación. (Ver Figura No. 14)

	<b>Manual de prácticas del Laboratorio de Redes de Datos Seguras</b>	Código:	MADO-31
		Versión:	04
		Página	109/297
		Sección ISO	8.3
		Fecha de emisión	17 de agosto de 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de Redes y Seguridad	
La impresión de este documento es una copia no controlada			

**NOTA:** *su* significa super usuario, por lo que se emplea la misma contraseña de root

**redes@debian:~\$ su**

**NOTA:** Para realizar la práctica exitosamente debe tener instalado los paquetes ifconfig, gcc y ssh.

- 4.1.5** Verifique que la tarjeta de red esté debidamente configurada y que tenga asignada una dirección IP dentro del rango: 192.168.2.25-192.168.2.60. Emplee el comando ifconfig

**root@debian:/home/redes# ifconfig**

Anote la dirección IP 192.163.1.110


En caso de no cumplir con lo indicado en el punto 4.1.5, configure debidamente la tarjeta. Teclee:

**root@debian:/home/redes# ifconfig eth0 192.168.2.X netmask 255.255.255.0**

**NOTA:** X se sustituye por una IP que se encuentre dentro del rango mencionado en el punto 4.1.5 para que esté dentro de la misma subred.

- 4.1.6** Verifique que la aplicación SSH se encuentre instalada (Active: active (running)) (Figura No. 3), para ello teclee:

**root@debian:/home/redes# service sshd status**

	<b>Manual de prácticas del Laboratorio de Redes de Datos Seguras</b>	Código:	MADO-31
		Versión:	04
		Página	110/297
		Sección ISO	8.3
		Fecha de emisión	17 de agosto de 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de Redes y Seguridad	
La impresión de este documento es una copia no controlada			

```

redes@debian: ~
Archivo Editar Ver Buscar Terminal Ayuda
redes@debian:~$ su
Contraseña:
root@debian:/home/redes# service sshd status
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled)
   Active: active (running) since mar 2017-05-23 21:19:02 MDT; 9min ago
   Process: 849 ExecReload=/bin/kill -HUP $MAINPID (code=exited, status=0/SUCCESS)
   Main PID: 388 (sshd)
   CGroup: /system.slice/ssh.service
           └─388 /usr/sbin/sshd -D

may 23 21:19:09 debian sshd[388]: Server listening on 0.0.0.0 port 22.
may 23 21:19:09 debian sshd[388]: Server listening on :: port 22.
may 23 21:19:29 debian sshd[388]: Received SIGHUP; restarting.
may 23 21:19:29 debian sshd[388]: Server listening on 0.0.0.0 port 22.
may 23 21:19:29 debian sshd[388]: Server listening on :: port 22.
may 23 21:19:29 debian sshd[388]: Received SIGHUP; restarting.
may 23 21:19:29 debian sshd[388]: Server listening on 0.0.0.0 port 22.
may 23 21:19:29 debian sshd[388]: Server listening on :: port 22.
root@debian:/home/redes#

```

**Figura No. 3. Verificación de SSH**

**NOTA:** En caso de que no se encuentre instalada, debe teclear el siguiente comando para instalarla (Figura No. 4):

**root@debian:/home/redes# apt-get install ssh**

```


redes@debian: ~
Archivo Editar Ver Buscar Terminal Ayuda
root@debian:/home/redes# apt-get install ssh

```

**Figura No. 4. Instalación de SSH**

**4.1.7** Teclee los siguientes comandos para eliminar cualquier archivo existente cuyo nombre inicie con prac (Figura No. 5)

**root@debian:/home/redes# rm -rf prac\***  
**root@debian:/home/redes# exit**

	<b>Manual de prácticas del Laboratorio de Redes de Datos Seguras</b>	Código:	MADO-31
		Versión:	04
		Página	111/297
		Sección ISO	8.3
		Fecha de emisión	17 de agosto de 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de Redes y Seguridad	
La impresión de este documento es una copia no controlada			

```

redes@debian: ~
Archivo Editar Ver Buscar Terminal Ayuda
root@debian:/home/redes# rm -rf prac*
root@debian:/home/redes# exit

```

**Figura No. 5. Eliminación de archivos**

**4.1.8** Salga de la cuenta de superusuario y emplee la cuenta de redes.

**4.1.9** Cree el subdirectorio **practica** dentro del directorio actual (Ver Figura No. 6)

**NOTA:** Evite cambiarle el nombre al subdirectorio, deberá llamarse **practica**, sin ningún número posteriormente ni abreviatura alguna, nombres como **prac8**, **p8**, **practica8**, etcétera, serán inválidos.

**redes@debian:~\$ mkdir practica**

```

redes@debian: ~
Archivo Editar Ver Buscar Terminal Ayuda
redes@debian:~$ mkdir practica
redes@debian:~$

```

**Figura No. 6. Creación del subdirectorio practica**

**4.1.10** Copie el archivo **sock-1.1.tar.tar** dentro del subdirectorio **practica**. (Ver figura No. 7)

**redes@debian:~\$ cp sock-1.1.tar.tar /home/redes/practica**

```

redes@debian: ~
Archivo Editar Ver Buscar Terminal Ayuda
redes@debian:~$ cp sock-1.1.tar.tar /home/redes/practica/


```

**Figura No. 7. Copia del archivo sock**

**4.1.11** Cámbiese al subdirectorio **practica** y descomprima el archivo **sock-1.1.tar.tar** (Ver Figura No. 8)

**redes@debian:~\$ cd practica**  
**redes@debian:~/practica\$ tar xvf sock-1.1.tar.tar**



	<b>Manual de prácticas del Laboratorio de Redes de Datos Seguras</b>	Código:	MADO-31
		Versión:	04
		Página	112/297
		Sección ISO	8.3
		Fecha de emisión	17 de agosto de 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de Redes y Seguridad	
La impresión de este documento es una copia no controlada			

```

redes@debian: ~/practica
Archivo Editar Ver Buscar Terminal Ayuda
redes@debian:~$ cd practica/
redes@debian:~/practica$ tar xvf sock-1.1.tar.tar
sock-1.1/
sock-1.1/ChangeLog
sock-1.1/Makefile.in
sock-1.1/config.h.in
sock-1.1/configure
sock-1.1/configure.in
sock-1.1/install-sh
sock-1.1/sock.c
sock-1.1/README
sock-1.1/sock.l
sock-1.1/sock.lsm
sock-1.1/debian/
sock-1.1/debian/changelog
sock-1.1/debian/control
sock-1.1/debian/copyright
sock-1.1/debian/rules
redes@debian:~/practica$

```

**Figura No. 8. Archivos en sock antes comprimidos.**

**4.1.12** Sitúese dentro del subdirectorio sock-1.1 y ejecute la orden **./configure** con la que el programa quedará preparado para su compilación y montaje. (Ver Figura No.9)

```

redes@debian:~/practica$ cd sock-1.1
redes@debian:~/practica/sock-1.1$ ./configure

```


```

redes@debian: ~/practica/sock-1.1
Archivo Editar Ver Buscar Terminal Ayuda
redes@debian:~/practica$ cd sock-1.1/
redes@debian:~/practica/sock-1.1$ ./configure
creating cache ./config.cache
checking for gcc... gcc
checking whether the C compiler (gcc ) works... yes
checking whether the C compiler (gcc ) is a cross-compiler... no
checking whether we are using GNU C... yes
checking whether gcc accepts -g... yes
checking whether warnings should be enabled... yes
checking for a BSD compatible install... /usr/bin/install -c
checking for gethostbyname in -lresolv... yes
checking for socket in -lsocket... no
checking for gethostbyname in -lnsl... yes
checking how to run the C preprocessor... gcc -E
checking for ANSI C header files... yes
checking for pid_t... yes
checking return type of signal handlers... void
updating cache ./config.cache
creating ./config.status
creating Makefile
creating config.h
redes@debian:~/practica/sock-1.1$

```

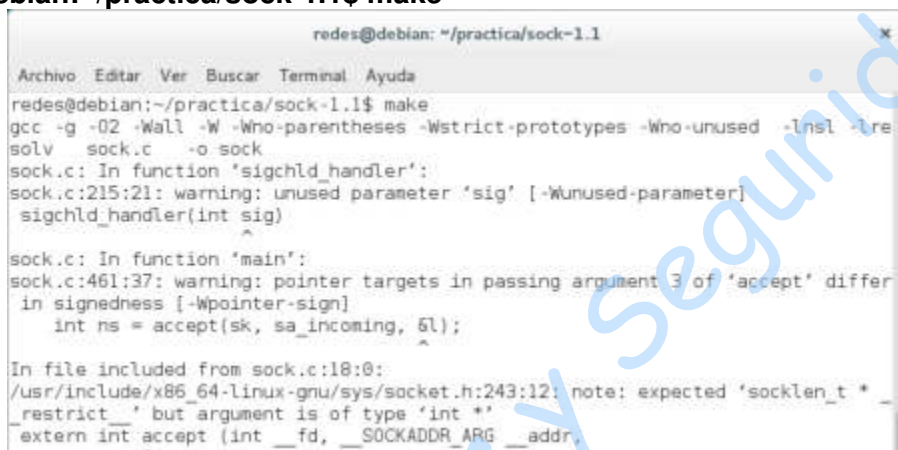
**Figura No. 9. Configuración de archivos y creación de un “Makefile”**



	<b>Manual de prácticas del Laboratorio de Redes de Datos Seguras</b>	Código:	MADO-31
		Versión:	04
		Página	113/297
		Sección ISO	8.3
		Fecha de emisión	17 de agosto de 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de Redes y Seguridad	
La impresión de este documento es una copia no controlada			

**4.1.13** Compile el programa. Ahora ya se dispone del programa sock ejecutable. (Ver figura No. 10)

**redes@debian:~/practica/sock-1.1\$ make**



```

redes@debian: ~/practica/sock-1.1
Archivo  Editor  Ver  Buscar  Terminal  Ayuda
redes@debian:~/practica/sock-1.1$ make
gcc -g -O2 -Wall -W -Wno-parentheses -Wstrict-prototypes -Wno-unused -lnsl -lsolv sock.c -o sock
sock.c: In function 'sigchld_handler':
sock.c:215:21: warning: unused parameter 'sig' [-Wunused-parameter]
  sigchld_handler(int sig)
                    ^
sock.c: In function 'main':
sock.c:461:37: warning: pointer targets in passing argument 3 of 'accept' differ
in signedness [-Wpointer-sign]
    int ns = accept(sk, sa_incoming, &l);
                                ^
In file included from sock.c:18:0:
/usr/include/x86_64-linux-gnu/sys/socket.h:243:12: note: expected 'socklen_t *'
_restrict__ but argument is of type 'int *'
extern int accept (int __fd, __SOCKADDR_ARG __addr,

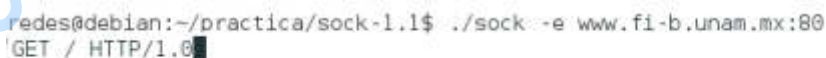
```

**Figura No. 10. Compilación de archivos**

## 4.2 Clientes TCP

**4.2.1** Observe qué sucede cuando un navegador se dirige a un servidor de web y le solicita una página. En el shell teclee lo siguiente y después de pulsar la tecla “ENTER”, escriba el texto GET / HTTP/1.0 Finalice presionando dos veces “ENTER” (Ver figura No. 11).

**redes@debian:~/practica/sock-1.1\$ ./sock -e www.fi-b.unam.mx:80**  
**GET / HTTP/1.0**




```

redes@debian:~/practica/sock-1.1$ ./sock -e www.fi-b.unam.mx:80
GET / HTTP/1.0

```

**Figura No. 11. Socket hacia www.fi-b.unam.mx**

Con esto se está conectando al servidor www.fi-b.unam.mx (que es el servidor web de la DIE) al puerto 80, que es donde se encuentra este servicio habitualmente (well-known port) y se utiliza el protocolo TCP. Lo que se está haciendo es crear un socket en nuestra computadora. Ese socket, que actúa como cliente, lo conectamos al servidor de web de la DIE y

	<b>Manual de prácticas del Laboratorio de Redes de Datos Seguras</b>	Código:	MADO-31
		Versión:	04
		Página	114/297
		Sección ISO	8.3
		Fecha de emisión	17 de agosto de 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de Redes y Seguridad	
La impresión de este documento es una copia no controlada			

le solicitamos que nos envíe el contenido de su página web inicial. La conexión iniciada por el programa sock se realiza al puerto 80 del servidor [www.fi-b.unam.mx](http://www.fi-b.unam.mx) y dura sólo lo indispensable hasta que se entrega la página web solicitada. Es importante destacar que la respuesta del servidor contiene una información del protocolo HTTP (o cabecera) a la que sigue, después de una línea en blanco, el código HTML de la página solicitada. Tras enviar esa información el servidor cierra la conexión, con lo cual la ejecución de la orden sock finaliza.

**4.2.2** En la terminal teclee lo siguiente:

**redes@debian:~/practica/sock-1.1\$ ./sock :22**

Deberá obtener como resultado algo similar a: (Ver figura No. 12).



```

redes@debian: ~/practica/sock-1.1
Archivo Editar Ver Buscar Terminal Ayuda
redes@debian:~/practica/sock-1.1$ ./sock :22
SSH-2.0-OpenSSH_6.7p1 Debian-5+deb8u3

```

**Figura No. 12. Socket usando el puerto 22**

Observará que el programa no finaliza, para que lo haga pulse las teclas <CTRL>+<c>.

En este ejercicio se está conectando con el servidor SSH local que se está ejecutando en la misma computadora desde el que ejecuta la orden. Esto es así porque al no especificar un servidor y sólo un puerto (22) se entiende que nos referimos a la computadora local.

El servidor SSH comienza enviando una cadena que identifica la versión del programa, y eso es lo que obtenemos como resultado.


### 4.3 Servidor TCP

Los programas pueden esperar pacientemente a que se les solicite algo antes de enviar alguna información. Éste es el comportamiento de muchos servidores. Utilizando el programa sock va a crear un servidor cuya única función es esperar a que un cliente se conecte y luego conecta la entrada y salida estándar con ese cliente.

**4.3.1** Para crear un socket servidor, teclee lo siguiente en el shell:

**NOTA: PUERTO deberá sustituirse por un número que esté dentro del rango de puertos 1024-65535.**

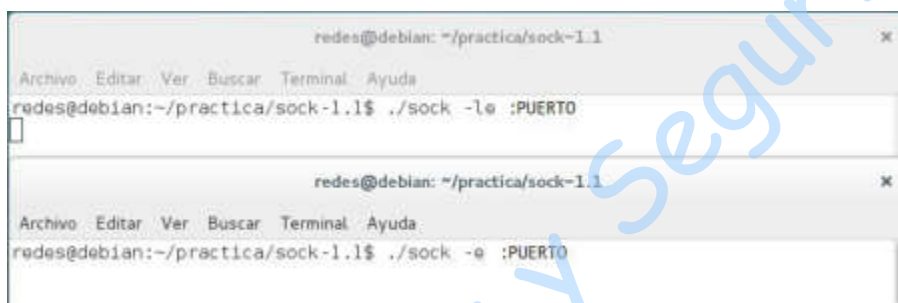
**redes@debian:~/practica/sock-1.1\$ ./sock -le :PUERTO**

	<b>Manual de prácticas del Laboratorio de Redes de Datos Seguras</b>	Código:	MADO-31
		Versión:	04
		Página	115/297
		Sección ISO	8.3
		Fecha de emisión	17 de agosto de 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de Redes y Seguridad	
La impresión de este documento es una copia no controlada			

**4.3.2** Ahora, abra un nuevo shell, sitúese en el subdirectorio sock-1.1 y ejecute la siguiente orden: (Ver figura No. 13).

**NOTA:** *PUERTO* deberá sustituirse por un número que esté dentro del rango de puertos 1024-65535 e igual al del punto 4.3.1

**redes@debian:~/practica/sock-1.1\$ ./sock -e :PUERTO**




**Figura No. 13. Creación de un socket servidor y de un socket cliente**

**4.3.3** Escriba en el Shell cliente y después teclee “ENTER” observe los que sucede en el Shell servidor. Seguidamente escriba en el Shell servidor, ¿qué sucede en el Shell cliente? (Ver figura No. 14).

Por el protocolo STP nos garantizan que lleguen los mensajes, lo que se escribe en servidor o en el cliente se refleja, mientras que UDP no garantiza la llegada de los paquetes, en STP hay una comunicación cliente-servidor, servidor-cliente



**Figura No. 14. Comunicación entre terminales**

	<b>Manual de prácticas del Laboratorio de Redes de Datos Seguras</b>	Código:	MADO-31
		Versión:	04
		Página	116/297
		Sección ISO	8.3
		Fecha de emisión	17 de agosto de 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de Redes y Seguridad	
La impresión de este documento es una copia no controlada			

Salga con CTRL + C

La orden del punto 4.3.2 es equivalente a: *telnet localhost PUERTO*

El parámetro -l hace que la aplicación configure el socket en modo escucha (*listen*) y acepte peticiones. Por tanto, en el punto 4.3.1 ha puesto en marcha, en su computadora, un servidor que escucha en el puerto seleccionado Mientras que las órdenes de los pasos 4.3.2 y 4.3.3 han arrancado clientes TCP que se han conectado a ese puerto.

**4.3.4** En un shell, sitúese en el subdirectorío sock-1.1 y cree un socket servidor tecleando lo siguiente:


**NOTA:** *PUERTO* deberá sustituirse por un número que esté dentro del rango de puertos 1024-65535.

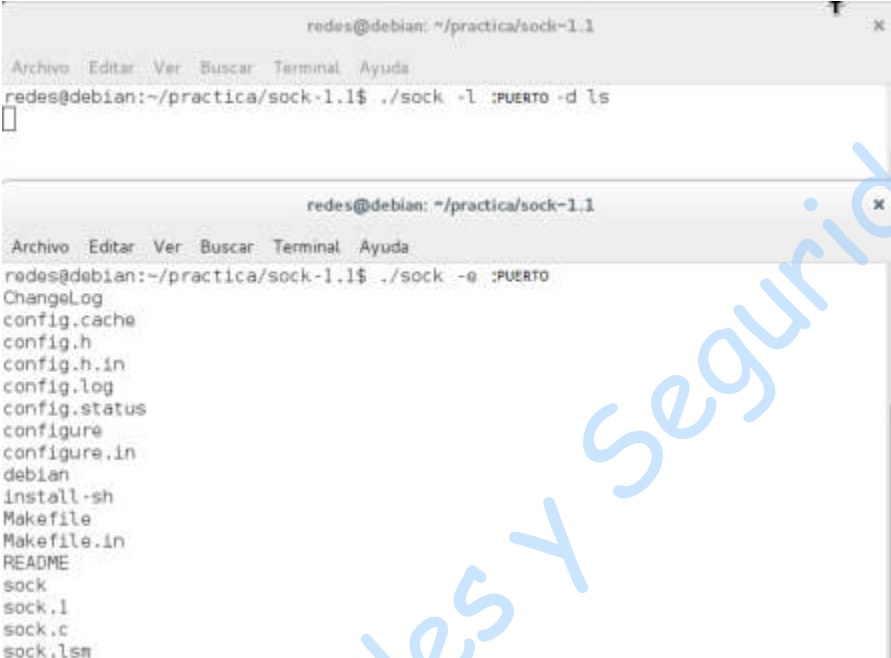
**redes@debian:~/practica/sock-1.1\$ ./sock -l :PUERTO -d ls**

**4.3.5** Ahora, en otro shell, sitúese en el subdirectorío sock-1.1 y cree un socket cliente ejecutando la orden: (Ver figura No. 15).

**NOTA:** *PUERTO* deberá sustituirse por un número que esté dentro del rango de puertos 1024-65535 e igual al del punto 4.3.4

**redes@debian:~/practica/sock-1.1\$ ./sock -e :PUERTO**

	<b>Manual de prácticas del Laboratorio de Redes de Datos Seguras</b>	Código:	MADO-31
		Versión:	04
		Página	117/297
		Sección ISO	8.3
		Fecha de emisión	17 de agosto de 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de Redes y Seguridad	
La impresión de este documento es una copia no controlada			



```

redes@debian: ~/practica/sock-1.1
Archivo Editar Ver Buscar Terminal Ayuda
redes@debian:~/practica/sock-1.1$ ./sock -l :PUERTO -d ls
█

redes@debian: ~/practica/sock-1.1
Archivo Editar Ver Buscar Terminal Ayuda
redes@debian:~/practica/sock-1.1$ ./sock -e :PUERTO
ChangeLog
config.cache
config.h
config.h.in
config.log
config.status
configure
configure.in
debian
install-sh
Makefile
Makefile.in
README
sock
sock.l
sock.c
sock.lsm

```

**Figura No. 15. Creación de un socket servidor y cliente**

#### 4.3.6 Observe lo que sucede.


En este experimento se ha construido un “miniservidor”. Lo que hace el programa es esperar la conexión de un usuario al puerto indicado y cuando el cliente se conecta (mediante la orden sock o el programa telnet) entonces ejecuta la orden `ls` que lista el contenido del directorio y lo envía a través del socket. Una vez finalizada la orden `ls` el servidor corta la conexión del cliente telnet, pero sigue escuchando en el puerto para atender nuevas peticiones de otros clientes.

Si se sustituye la orden `ls` por la orden `date` en el punto 4.3.4 tendrá un miniservidor de fecha y hora.

### 4.4 El protocolo UDP

Del mismo modo que en los ejemplos anteriores ha utilizado el protocolo TCP, ahora va a ver cómo se puede enviar información mediante el protocolo UDP. Para ello mantendrá los dos shells que tiene abiertos.

#### 4.4.1 En un shell cree un socket servidor tecleando lo siguiente:

	<b>Manual de prácticas del Laboratorio de Redes de Datos Seguras</b>	Código:	MADO-31
		Versión:	04
		Página	118/297
		Sección ISO	8.3
		Fecha de emisión	17 de agosto de 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de Redes y Seguridad	
La impresión de este documento es una copia no controlada			

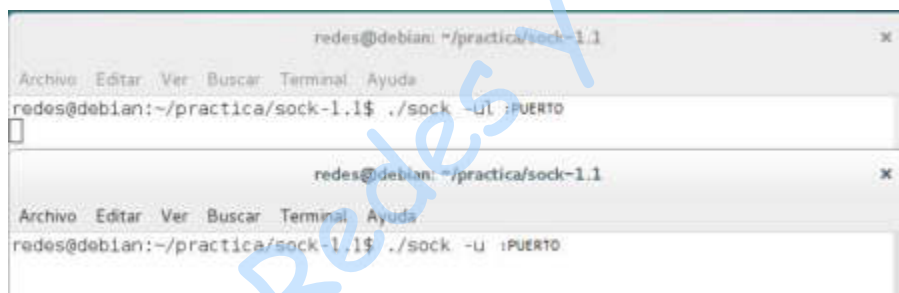
**NOTA:** *PUERTO* deberá sustituirse por un número que esté dentro del rango de puertos 1024-65535.

**redes@debian:~/practica/sock-1.1\$ ./sock -ul :PUERTO**

**4.4.2** Y en otro shell ejecute la orden: (Ver Figura No. 16).

**NOTA:** *PUERTO* deberá sustituirse por un número que esté dentro del rango de puertos 1024-65535 e igual al del punto 4.4.1

**redes@debian:~/practica/sock-1.1\$ ./sock -u :PUERTO**



```

redes@debian: ~/practica/sock-1.1
Archivo Editar Ver Buscar Terminal Ayuda
redes@debian:~/practica/sock-1.1$ ./sock -ul :PUERTO

redes@debian: ~/practica/sock-1.1
Archivo Editar Ver Buscar Terminal Ayuda
redes@debian:~/practica/sock-1.1$ ./sock -u :PUERTO

```

**Figura No.16. Socket servidor y cliente.**

**4.4.3** Escriba en el Shell cliente y después del ENTER observe lo que sucede en el Shell servidor. (Ver figura No. 17). Realice la prueba del shell servidor hacia el cliente.



```


redes@debian: ~/practica/sock-1.1
Archivo Editar Ver Buscar Terminal Ayuda
redes@debian:~/practica/sock-1.1$ ./sock -ul :PUERTO
ya falta porco para que termine la clase

redes@debian: ~/practica/sock-1.1
Archivo Editar Ver Buscar Terminal Ayuda
redes@debian:~/practica/sock-1.1$ ./sock -u :PUERTO
ya falta porco para que termine la clase

```

**Figura No. 17. Comunicación entre terminales.**



	<b>Manual de prácticas del Laboratorio de Redes de Datos Seguras</b>	Código:	MADO-31
		Versión:	04
		Página	119/297
		Sección ISO	8.3
		Fecha de emisión	17 de agosto de 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de Redes y Seguridad	
La impresión de este documento es una copia no controlada			

Comente lo que sucede

Lo que resulta es que el protocolo UDP no garantiza la llegada de paquetes solo es cliente-servidor, pero si escribimos algo en el servidor no se refleja en el cliente

Salga con CTRL + C, en el Shell del cliente.

**4.4.4** Ahora en el Shell cliente cambie la orden del paso número 4.4.2 por la siguiente: (Ver figura No. 18).

**NOTA: PUERTO deberá sustituirse por un número que esté dentro del rango de puertos 1024-65535 e igual al del punto 4.4.1**

**redes@debian:~/practica/sock-1.1\$ date | ./sock -u :PUERTO**



```

redes@debian: ~/practica/sock-1.1
Archivo Editar Ver Buscar Terminal Ayuda
redes@debian:~/practica/sock-1.1$ ./sock -ul :PUERTO
jue jun 22 09:05:29 MDT 2017

redes@debian: ~/practica/sock-1.1
Archivo Editar Ver Buscar Terminal Ayuda
redes@debian:~/practica/sock-1.1$ date | ./sock -u :PUERTO
redes@debian:~/practica/sock-1.1$

```


Figura No. 18. Comunicación entre terminales.

Como ve el funcionamiento es bastante similar, pero al carecer UDP del concepto de conexión no se puede construir un servidor de manera tan sencilla.

Pero la razón que hace que UDP tenga utilidad para muchas aplicaciones es su capacidad para hacer difusiones (enviando a la dirección 255.255.255.255 realmente se envía un datagrama que será recibido por todas las computadoras de la misma red IP). Sin embargo, y por motivos de seguridad, el uso de esta característica está restringido y no se empleará en esta práctica.

Una forma de evitar esta restricción es emplear la dirección IP de multicast que esté configurada en todos sus equipos como si se tratara de una dirección de difusión.



	<b>Manual de prácticas del Laboratorio de Redes de Datos Seguras</b>	Código:	MADO-31
		Versión:	04
		Página	120/297
		Sección ISO	8.3
		Fecha de emisión	17 de agosto de 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de Redes y Seguridad	
La impresión de este documento es una copia no controlada			

#### 4.5 Transferencia de archivos

En los ejercicios anteriores ha visto algunos de los usos que nos permite un socket. Ahora va a utilizar los servicios de TCP y UDP para el envío de archivos entre dos computadoras.

En el siguiente ejercicio se mostrará cómo transferir un archivo empleando el programa sock:

**4.5.1** Copie una imagen (por ejemplo dibujo.bmp) al subdirectorio /home/redes/practica/sock-1.1

**4.5.2** Ahora va a enviar la imagen tecleando en el Shell emisor (Ver figura No. 19):

**NOTA 1:** *cat* es un comando que no puede ser omitido.

**NOTA 2:** “dibujo.bmp” es el nombre original de la imagen.

**NOTA 3:** *PUERTO* deberá sustituirse por un número que esté dentro del rango de puertos 1024-65535.

**redes@debian:~/practica/sock-1.1\$ ./sock -l :PUERTO -d 'cat dibujo.bmp'**




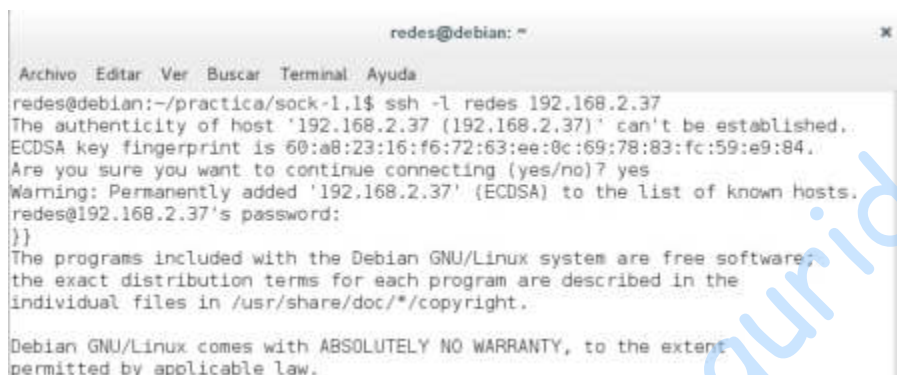
**Figura No.19.** Envío de la imagen desde el Shell emisor.

**4.5.3** Conéctese a la máquina que le indique su profesor con la cuenta **redes** desde uno de los shells tecleando: (Ver figura No. 20).

**redes@debian:~/practica/sock-1.1\$ ssh -l redes 192.168.2.X**

**NOTA:** X se sustituirá por la IP de la computadora.

	<b>Manual de prácticas del Laboratorio de Redes de Datos Seguras</b>	Código:	MADO-31
		Versión:	04
		Página	121/297
		Sección ISO	8.3
		Fecha de emisión	17 de agosto de 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de Redes y Seguridad	
La impresión de este documento es una copia no controlada			



```

redes@debian: ~
Archivo Editar Ver Buscar Terminal Ayuda
redes@debian:~/practica/sock-1.1$ ssh -l redes 192.168.2.37
The authenticity of host '192.168.2.37 (192.168.2.37)' can't be established.
ECDSA key fingerprint is 60:a8:23:16:f6:72:63:ee:8c:69:78:83:fc:59:e9:84.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.2.37' (ECDSA) to the list of known hosts.
redes@192.168.2.37's password:
}}
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

```

**Figura No. 20. Conexión por medio de ssh en el Shell receptor**

**4.5.4** En el Shell del paso anterior, sitúese en el subdirectorio sock-1.1 y teclee: (Ver figura No. 21).

**NOTA 1:** *PUERTO* deberá sustituirse por un número que esté dentro del rango de puertos 1024-65535 e igual al del punto 4.5.2

```

redes@debian:~$ cd practica/sock-1.1
redes@debian:~/practica/sock-1.1$ ./sock -e 192.168.2.X:PUERTO>imagen2.bmp

```

**NOTA 2:** X se sustituirá por la IP de su computadora



```


redes@debian: ~/practica/sock-1.1
Archivo Editar Ver Buscar Terminal Ayuda
redes@debian:~/practica/sock-1.1$ ./sock -e 192.168.2.35:PUERTO>imagen2.bmp

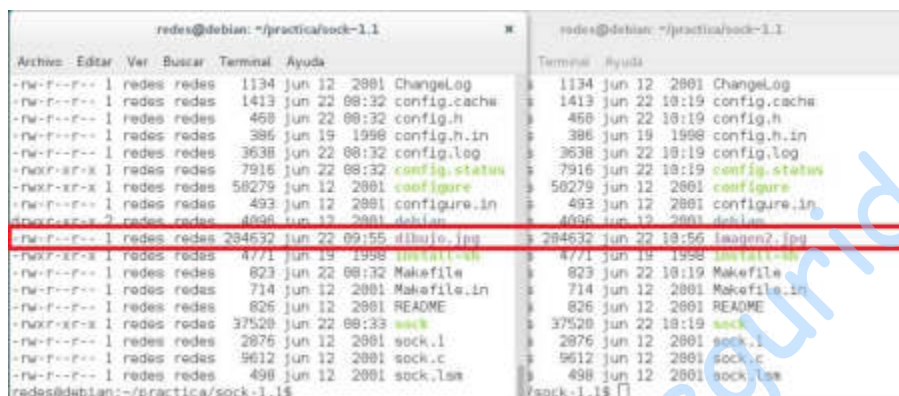
```

**Figura No. 21. Recepción de la imagen en el Shell receptor**

**NOTA 3:** "imagen2.bmp" es un segundo nombre para la imagen

**4.5.5** Compruebe que el archivo recibido en la máquina con la cual se conectó tiene el mismo tamaño que el original, utilice el comando: *ls -la*. (Ver figura No. 22).

	<b>Manual de prácticas del Laboratorio de Redes de Datos Seguras</b>	Código:	MADO-31
		Versión:	04
		Página	122/297
		Sección ISO	8.3
		Fecha de emisión	17 de agosto de 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de Redes y Seguridad	
La impresión de este documento es una copia no controlada			



**Figura No. 22. Comparación de los archivos.**

En este ejercicio se ha realizado la transferencia del archivo mediante el protocolo TCP. Su computadora ha quedado a la espera de un cliente en el paso 4.5.2. Y desde la máquina de al lado se ha conectado como tal cliente en el paso 4.5.4.

Es interesante resaltar que aunque el archivo resultante tenga el mismo tamaño, eso no garantiza que la transferencia ha tenido éxito (¿y sí el contenido fuera diferente?). Ahora enviará el archivo de vuelta para poderlo comprobar, pero empleando el protocolo UDP.

Escriba “exit” en ambos Shells hasta cerrarlos.

**4.5.6** Abra un shell, sitúese en el subdirectorio sock-1.1 y teclee (Ver figura No. 23):

**NOTA:** *PUERTO* deberá sustituirse por un número que esté dentro del rango de puertos 1024-65535.

```
redes@debian:~$ cd practica/sock-1.1
```


```
redes@debian:~/practica/sock-1.1$ ./sock -ul :PUERTO>dibujo2.bmp
```



**Figura No.23. Recepción del archivo**

Lo que le prepara para recibir el archivo, -u indica UDP

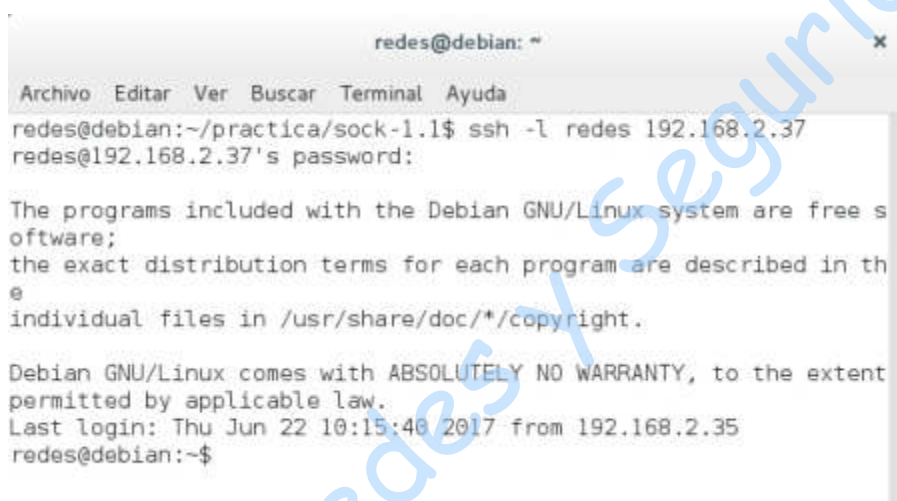
**NOTA:** “dibujo2.bmp” es un tercer nombre para la imagen para diferenciarlo de los anteriores.

	<b>Manual de prácticas del Laboratorio de Redes de Datos Seguras</b>	Código:	MADO-31
		Versión:	04
		Página	123/297
		Sección ISO	8.3
		Fecha de emisión	17 de agosto de 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de Redes y Seguridad	
La impresión de este documento es una copia no controlada			

**4.5.7** Abra un segundo Shell y conéctese con la cuenta redes a la máquina con la que realizó la conexión anterior desde un shell tecleando: (Ver figura No. 24).

**redes@debian:~\$ ssh -l redes 192.168.2.X**

**NOTA:** X se sustituirá por la IP de la computadora remota.



```

redes@debian: ~
Archivo Editar Ver Buscar Terminal Ayuda
redes@debian:~/practica/sock-1.1$ ssh -l redes 192.168.2.37
redes@192.168.2.37's password:

The programs included with the Debian GNU/Linux system are free s
oftware;
the exact distribution terms for each program are described in th
e
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Jun 22 10:15:40 2017 from 192.168.2.35
redes@debian:~$

```

**Figura No. 24. Conexión por medio de ssh**

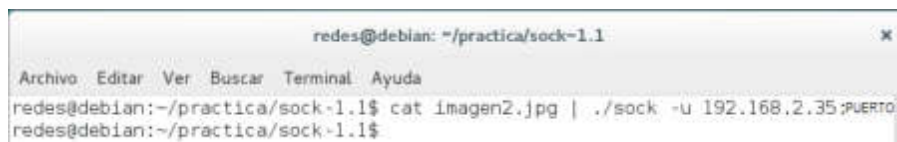
**4.5.8** En el mismo Shell del paso anterior, sitúese en el subdirectorio sock-1.1 y teclee lo siguiente para enviar el archivo: (Ver figura No. 25).

**NOTA:** *PUERTO* deberá sustituirse por un número que esté dentro del rango de puertos 1024-65535 e igual al del punto 4.5.6

**redes@debian:~\$ cd practica/sock-1.1**

**redes@debian:~/practica/sock-1.1\$ cat imagen2.bmp | ./sock -u 192.168.2.X:PUERTO**

**NOTA:** XX se sustituirá por la IP de su computadora




```

redes@debian: ~/practica/sock-1.1
Archivo Editar Ver Buscar Terminal Ayuda
redes@debian:~/practica/sock-1.1$ cat imagen2.jpg | ./sock -u 192.168.2.35:PUERTO
redes@debian:~/practica/sock-1.1$

```

**Figura No.25. Envío del archivo**

	<b>Manual de prácticas del Laboratorio de Redes de Datos Seguras</b>	Código:	MADO-31
		Versión:	04
		Página	124/297
		Sección ISO	8.3
		Fecha de emisión	17 de agosto de 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de Redes y Seguridad	
La impresión de este documento es una copia no controlada			

**4.5.9** A continuación, finalice la orden del paso 4.5.8 pulsando <Ctrl>+<c> en el primer shell (asegúrese de que la ha seleccionado primero, haciendo clic con el ratón). (Ver figura No. 26).



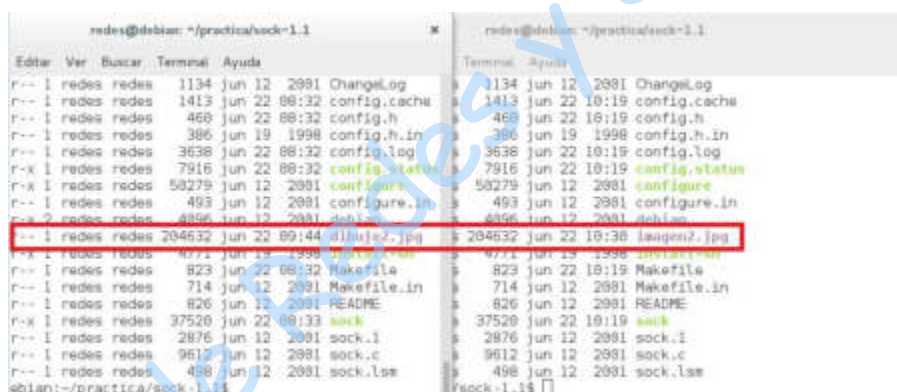
```

redes@debian: ~/practica/sock-1.1
Archivo Editar Ver Buscar Terminal Ayuda
redes@debian:~/practica/sock-1.1$ ./sock -ul :PUERTO>dibujo2.jpg
^C
redes@debian:~/practica/sock-1.1$

```

**Figura No. 26. Final de la instrucción**

**4.5.10** Compruebe que los archivos “imagen2.bmp” (enviado) y “dibujo2.bmp” (recibido) son iguales con la orden *ls -la*. (Ver figura No. 27).



```

redes@debian: ~/practica/sock-1.1
ls -la
-rw-r--r-- 1 redes redes 204632 jun 22 09:44 dibujo2.jpg
...

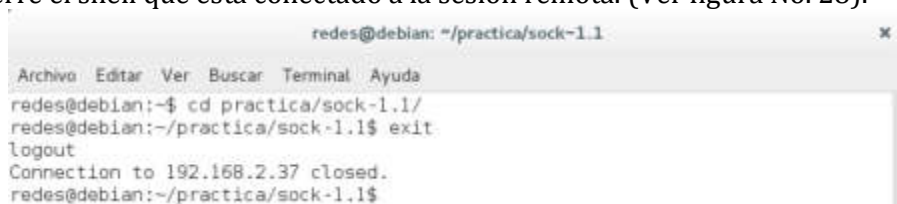
redes@debian: ~/practica/sock-1.1
ls -la
-rw-r--r-- 1 redes redes 204632 jun 22 10:38 imagen2.jpg
...

```

**Figura No. 27. Comparación de la imagen enviada y recibida**

Si ambos archivos son iguales entonces podrá concluir que tanto la transmisión desde su computadora a la de al lado, empleando TCP, como la vuelta, empleando UDP, no han sufrido errores. Si repite la operación con un archivo mayor (por ejemplo, el enunciado de esta práctica en pdf) encontrará que la transmisión por TCP no tiene problemas pero la de UDP fallará eventualmente, aunque este punto no se realizará.

**4.5.11** Cierre el shell que está conectado a la sesión remota. (Ver figura No. 28).




```

redes@debian: ~/practica/sock-1.1
Archivo Editar Ver Buscar Terminal Ayuda
redes@debian:~/practica/sock-1.1$ cd practica/sock-1.1/
redes@debian:~/practica/sock-1.1$ exit
logout
Connection to 192.168.2.37 closed.
redes@debian:~/practica/sock-1.1$

```

**Figura No. 28. Cierre de la conexión por ssh.**



	<b>Manual de prácticas del Laboratorio de Redes de Datos Seguras</b>	Código:	MADO-31
		Versión:	04
		Página	125/297
		Sección ISO	8.3
		Fecha de emisión	17 de agosto de 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de Redes y Seguridad	
La impresión de este documento es una copia no controlada			

#### 4.5.12 Cierre sesión.

### 5.-Cuestionario

1. De acuerdo con lo visto en el desarrollo de la práctica ¿qué diferencias sustanciales existen entre TCP y UDP?

La transmisión de información usando TCP es más ordenada, es decir, requiere un constante monitoreo del correcto envío, recepción, orden y errores de información enviada. Esto se ve claramente en los "minichats" que se implementaron. En el caso de UDP, estas confirmaciones no se dan, por lo que el envío de información es más descuidada ya que no se garantiza ni la entrega ni el orden en que se envían las tramas.

2. ¿Por qué la conexión iniciada por el socket al servidor sólo dura lo necesario para recibir la información requerida?

El servidor está ejecutándose y esperando a que otro se conecte con él, él nunca da el primer paso de conexión es el que sirve información al que se la pida. Esto es para optimizar el desempeño, ya que esperar de forma indefinida para recibir información puede generar un desperdicio de recursos que pueden ser utilizados en otros enlaces de comunicación. Además, esto puede implicar corroborar de forma constante la conexión.

3. Mencione algunos ejemplos de los usos de TCP y UDP

TCP: Comunicación por texto (mensajerías como WhatsApp, Messenger, iMessage), Transferencia de archivos. (si se requiere evitar pérdidas), Servicios de correo electrónico (Gmail, Outlook, Yahoo), Acceso a página web con el protocolo HTTP  
UDP Videoconferencia (Zoom, Meet, Teams), Juegos en línea, Comunicación remota por voz, Mapeo de dominios a sus respectivas IP

### 6.- Conclusiones.

Revise los objetivos planteados al inicio de la práctica y concluya.


Martínez Rojas José Eduardo

En resumidas cuentas se pudo enviar información utilizando los protocolos UDP Y TCP, las diferencias que hay al enviar los paquetes con estos 2 tipos de protocolos y también dependiendo de lo que se requiera. La transferencia de archivos no se pudo realizar pero se pudo analizar como es el envío de información de una computadora a otra o en este caso de máquinas virtuales. En fin fue una buena práctica con información útil de que protocolos utilizar para el envío de información y como lo hacen.

Mateos Flores Erik Esteban

Se implementaron algunos escenarios para TCP y UDP para corroborar cómo operan en la comunicación entre cliente y servidor. Resultó interesante aprender que implementaciones como los chats requieren el uso de TCP para validar constantemente la correcta transmisión de datos. Por otro lado, la implementación de UDP que parece tener una comunicación más descuidada, en realidad proporciona cierto grado de seguridad al hacer que la información solo se envíe desde un solo lado.

Para ambos casos, existen aplicaciones que podemos encontrar de forma cotidiana y de las cuales estamos muy acostumbrados.

	<b>Manual de prácticas del Laboratorio de Redes de Datos Seguras</b>	Código:	MADO-31
		Versión:	04
		Página	126/297
		Sección ISO	8.3
		Fecha de emisión	17 de agosto de 2021
Facultad de Ingeniería		Área/Departamento: Laboratorio de Redes y Seguridad	
La impresión de este documento es una copia no controlada			

**PRÁCTICA 8**  
**TCP y UDP**  
***Cuestionario Previo***

1. Mencione al menos 2 funciones de la capa de transporte del Modelo OSI
2. Mencione algunos protocolos de transporte (no incluya TCP ni UDP).
3. ¿Qué es el protocolo de transporte TCP?
4. ¿Qué es el protocolo de transporte UDP?
5. Dibuje un datagrama UDP.
6. Dibuje un segmento TCP.
7. ¿Qué es un socket y qué se necesita para crearlo?
8. ¿Para qué se usa el comando apt-get install gcc o apt install gcc?
9. ¿Para qué se usa el comando apt-get install ssh o apt install ssh?
10. ¿Qué es un puerto?
11. ¿Cuáles son los rangos de puertos existentes?
12. ¿Qué rangos de puertos pueden utilizarse para establecer comunicaciones?



```
Actividades Terminal 25 de oct 18:26
edy@debian: ~/practica/sock-1.1
edy@debian:~$ su .
su: el usuario . no existe o la entrada del usuario no contiene todos los campos
requeridos
edy@debian:~$ su -
Contraseña:
root@debian:~# apt-get install gcc
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  binutils binutils-common binutils-x86-64-linux-gnu gcc-10 libasan6
  libatomic1 libbinutils libc-dev-bin libc-devtools libc6 libc6-dev libcc1-0
  libcrypt-dev libctf-nobfd0 libctf0 libgcc-10-dev libitm1 liblsan0 libnsl-dev
  libtirpc-dev libtsan0 libubsan1 linux-libc-dev manpages-dev
Paquetes sugeridos:
  binutils-doc gcc-multilib make autoconf automake libtool flex bison gdb
  gcc-doc gcc-10-multilib gcc-10-doc gcc-10-locales glibc-doc
Paquetes recomendados:
  libnss-nis libnss-nisplus
Se instalarán los siguientes paquetes NUEVOS:
  binutils binutils-common binutils-x86-64-linux-gnu gcc gcc-10 libasan6
  libatomic1 libbinutils libc-dev-bin libc-devtools libc6-dev libcc1-0
  libcrypt-dev libctf-nobfd0 libctf0 libgcc-10-dev libitm1 liblsan0 libnsl-dev
  libtirpc-dev libtsan0 libubsan1 linux-libc-dev manpages-dev
```

```
ja DISPOSITIVOS Ayuda
Actividades Terminal 25 de oct 18:26
edy@debian: ~/practica/sock-1.1
edy@debian:~/practica$ tar ' /home/edy/Descargas/sock-1.1.tar.tar'
tar: La opción antigua 'g' requiere un argumento
Pruebe 'tar --help' o 'tar --usage' para más información.
edy@debian:~/practica$ tar xvf ' /home/edy/Descargas/sock-1.1.tar.tar'
sock-1.1/
sock-1.1/ChangeLog
sock-1.1/Makefile.in
sock-1.1/config.h.in
sock-1.1/configure
sock-1.1/configure.in
sock-1.1/install-sh
sock-1.1/sock.c
sock-1.1/README
sock-1.1/sock.1
sock-1.1/sock.lsm
sock-1.1/debian/
sock-1.1/debian/changelog
sock-1.1/debian/control
sock-1.1/debian/copyright
sock-1.1/debian/rules
edy@debian:~/practica$ cd sock-1.1
edy@debian:~/practica/sock-1.1$ ./configure
creating cache ./config.cache
checking for gcc... gcc
```

```
Actividades Terminal 25 de oct 18:27
Carpeta pe

edy@debian: ~/practica/sock-1.1
root@debian:~/practica/sock-1.1# exit
cerrar sesión
edy@debian:~$ mkdir practica
edy@debian:~$ cd practica
edy@debian:~/practica$ cp sock-1.1.tar.tar '/home/edy/Descargas/sock-1.1.tar.tar'
cp: no se puede efectuar 'stat' sobre 'sock-1.1.tar.tar': No existe el fichero o el directorio
edy@debian:~/practica$ tar '/home/edy/Descargas/sock-1.1.tar.tar'
tar: La opción antigua 'g' requiere un argumento
Pruebe 'tar --help' o 'tar --usage' para más información.
edy@debian:~/practica$ tar xvf '/home/edy/Descargas/sock-1.1.tar.tar'
sock-1.1/
sock-1.1/ChangeLog
sock-1.1/Makefile.in
sock-1.1/config.h.in
sock-1.1/configure
sock-1.1/configure.in
sock-1.1/install-sh
sock-1.1/sock.c
sock-1.1/README
sock-1.1/sock.1
sock-1.1/sock.lsm
```

```
Actividades Terminal 25 de oct 18:27
Carpeta pe

edy@debian: ~/practica/sock-1.1
sock-1.1/debian/copyright
sock-1.1/debian/rules
edy@debian:~/practica$ cd sock-1.1
edy@debian:~/practica/sock-1.1$ ./configure
creating cache ./config.cache
checking for gcc... gcc
checking whether the C compiler (gcc ) works... yes
checking whether the C compiler (gcc ) is a cross-compiler... no
checking whether we are using GNU C... yes
checking whether gcc accepts -g... yes
checking whether warnings should be enabled... yes
checking for a BSD compatible install... /usr/bin/install -c
checking for gethostbyname in -lresolv... yes
checking for socket in -lsocket... no
checking for gethostbyname in -lnsl... yes
checking how to run the C preprocessor... gcc -E
checking for ANSI C header files... yes
checking for pid_t... yes
checking return type of signal handlers... void
updating cache ./config.cache
creating ./config.status
creating Makefile
creating config.h
edy@debian:~/practica/sock-1.1$ make
```

```
actividades Terminal 25 de oct 18:27

checking how to run the C preprocessor... gcc -E
checking for ANSI C header files... yes
checking for pid_t... yes
checking return type of signal handlers... void
updating cache ./config.cache
creating ./config.status
creating Makefile
creating config.h
edy@debian:~/practica/sock-1.1$ make
gcc -g -O2 -Wall -W -Wno-parentheses -Wstrict-prototypes -Wno-unused -lnsl -lsolv sock.c -o sock
sock.c: In function 'main':
sock.c:461:37: warning: pointer targets in passing argument 3 of 'accept' differ
in signedness [-Wpointer-sign]
  461 |     int ns = accept(sk, sa_incoming, &l);
      |                                ~^
      |                                |
      |                                int *
In file included from sock.c:18:
/usr/include/x86_64-linux-gnu/sys/socket.h:233:28: note: expected 'socklen_t * restrict' {aka 'unsigned int * restrict'} but argument is of type 'int *'
  233 |     socklen_t *__restrict __addr_len);
edy@debian:~/practica/sock-1.1$ ./sock -e www.google.com:80
```

```
debian [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Actividades Terminal 25 de oct 18:41

edy@debian:~/practica$ cd practica
edy@debian:~/practica$ cd sock-1.1
edy@debian:~/practica/sock-1.1$ ./sock
./sock: connect: Connection refused
edy@debian:~/practica/sock-1.1$ ./sock -l
./sock: connect: Connection refused
Listening mode
edy@debian:~/practica/sock-1.1$ ./sock -d
Daemon mode (listen only) -- process multiple connections
edy@debian:~/practica/sock-1.1$ ./sock -e
Terminate as soon as EOF is seen in any direction
edy@debian:~/practica/sock-1.1$ ./sock -n
Avoid reverse DNS lookups
edy@debian:~/practica/sock-1.1$ ./sock :22
SSH-2.0-OpenSSH_8.4p1 Debian-5
Hola mundo
Invalid SSH identification string.
edy@debian:~/practica/sock-1.1$ ./sock :22
SSH-2.0-OpenSSH_8.4p1 Debian-5
^C
edy@debian:~/practica/sock-1.1$ ./sock .le :1030
./sock: .le: port number required
edy@debian:~/practica/sock-1.1$ ./sock .le :1030
./sock: .le: port number required
edy@debian:~/practica/sock-1.1$ ./sock .le :1050
./sock: .le: port number required
edy@debian:~/practica/sock-1.1$ ./sock -le :1050
Hola mundo
```

```
debian [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Actividades Terminal 25 de oct 18:42
edy@debian: ~/practica/sock-1.1
edy@debian:~$ cd practica
edy@debian:~/practica$ cd sock-1.1
edy@debian:~/practica/sock-1.1$ ./sock
./sock: connect: Connection refused
edy@debian:~/practica/sock-1.1$ ./sock -d
-d Daemon mode (listen only) -- process multiple connections
-e Terminate as soon as EOF is seen in any direction
-n Avoid reverse DNS lookups
edy@debian:~/practica/sock-1.1$ ./sock :22
SSH-2.0-OpenSSH_8.4p1 Debian-5
./sock: connect: Connection refused
Invalid SSH identification string.
Hola mundo
hola desde servidor
edy@debian:~/practica/sock-1.1$
edy@debian:~/practica/sock-1.1$ ./sock :22
SSH-2.0-OpenSSH_8.4p1 Debian-5
^C
edy@debian:~/practica/sock-1.1$ ./sock .le :1030
./sock: .le: port number required
edy@debian:~/practica/sock-1.1$ ./sock .le :1030
./sock: .le: port number required
edy@debian:~/practica/sock-1.1$ ./sock .le :1050
./sock: .le: port number required
edy@debian:~/practica/sock-1.1$ ./sock -le :1050
Hola mundo
hola desde servidor
```

```
debian [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Actividades Terminal 25 de oct 18:44
edy@debian: ~/practica/sock-1.1
Hola mundo
hola desde servidor
edy@debian:~/practica/sock-1.1$ ./sock -e :1050
./sock: connect: Connection refused
edy@debian:~/practica/sock-1.1$ ./sock -e :1030
ChangeLog
config.cache
config.h
config.h.in
config.log
config.status
configure
configure.in
debian
install-sh
Makefile
Makefile.in
README
sock
sock.1
sock.c
sock.lsm
edy@debian:~/practica/sock-1.1$
edy@debian:~/practica/sock-1.1$ ./sock -l :1030 -d ls
^[[A^[[B^[[B
```

The screenshot shows a terminal window titled "debian [Corriendo] - Oracle VM VirtualBox". The terminal is running a shell with the prompt "edy@debian: ~/practica/sock-1.1". The directory listing shows files like config.cache, config.h, config.h.in, config.log, config.status, configure, configure.in, debian, install-sh, Makefile, Makefile.in, README, sock, sock.1, sock.c, and sock.lsm. The user enters the command `./sock -u :1030`, which results in a "bash: ./sock: No existe el fichero o el directorio" error. The user then enters `date | ./sock -u :1030`, which outputs "hola de nuevo" and the date "lun 25 oct 2021 18:49:21 CEST".

```
config.cache
config.h
config.h.in
config.log
config.status
configure
configure.in
debian
install-sh
Makefile
Makefile.in
README
sock
sock.1
sock.c
sock.lsm
edy@debian:~/practica/sock-1.1$ ^C
edy@debian:~/practica/sock-1.1$ ./sock -u :1030
bash: ./sock: No existe el fichero o el directorio
edy@debian:~/practica/sock-1.1$ ./sock -u :1030
hola
^C
edy@debian:~/practica/sock-1.1$ date | ./sock -u :1030
edy@debian:~/practica/sock-1.1$
edy@debian:~/practica/sock-1.1$ ./sock -u :1030
hola
hola de nuevo
lun 25 oct 2021 18:49:21 CEST
```

## Bibliografía

ULPGS.(2007).Sistemas operativos. Consultado el:25/10/2021 Recuperado de:<http://sopa.dis.ulpgc.es/ii-dso/leclinux/ipc/sockets/sockets.pdf>

Erik.(2021).Ejemplos TCP y UDP Consultado el:25/10/2021 Recuperado de:<https://www.it-swarm-es.com/es/tcp/cuales-son-ejemplos-de-tcp-y-udp-en-la-vida-real/971487241/>

Yazid.h.(2021).Diferencias TCP y UDP Consultado el:25/10/2021 Recuperado de:[https://nanopdf.com/download/diferencias-entre-tcp-y-udp-el-protocolo-udp-udp-es-un\\_pdf](https://nanopdf.com/download/diferencias-entre-tcp-y-udp-el-protocolo-udp-udp-es-un_pdf)