

Redes de Computadoras

# Proyecto II

## Creación e Implementación de un Protocolo de la Capa de Aplicación

---

### Objetivo

El objetivo del proyecto es crear un protocolo de red desde cero, con toda la implementación del envío de mensajes necesario para interactuar con un servidor mediante una aplicación de Pokemon.

Las aplicaciones simulan el safari pokemon en consola, donde se pueden atrapar pokémon (151 originales) arrojando piedras o carnada para mejorar probabilidades de captura o bajar probabilidades de huida de los pokemon.

Al capturar los pokémon se transfiere la imagen correspondiente del servidor a el directorio de almacenamiento para simular el sistema de almacenamiento de los juegos.

Fue desarrollado en **C** completamente, y probado en sistemas **Fedora** y **Debian**, las instrucciones de instalación son sencillas y solo se necesita de un archivo para esto, se necesitan permisos de super usuario para instalar las páginas del manual de **Linux**.

Una vez instalado, se tiene acceso al código fuente y a su documentación completa.

---

---

## Diseño del Protocolo

### Estados

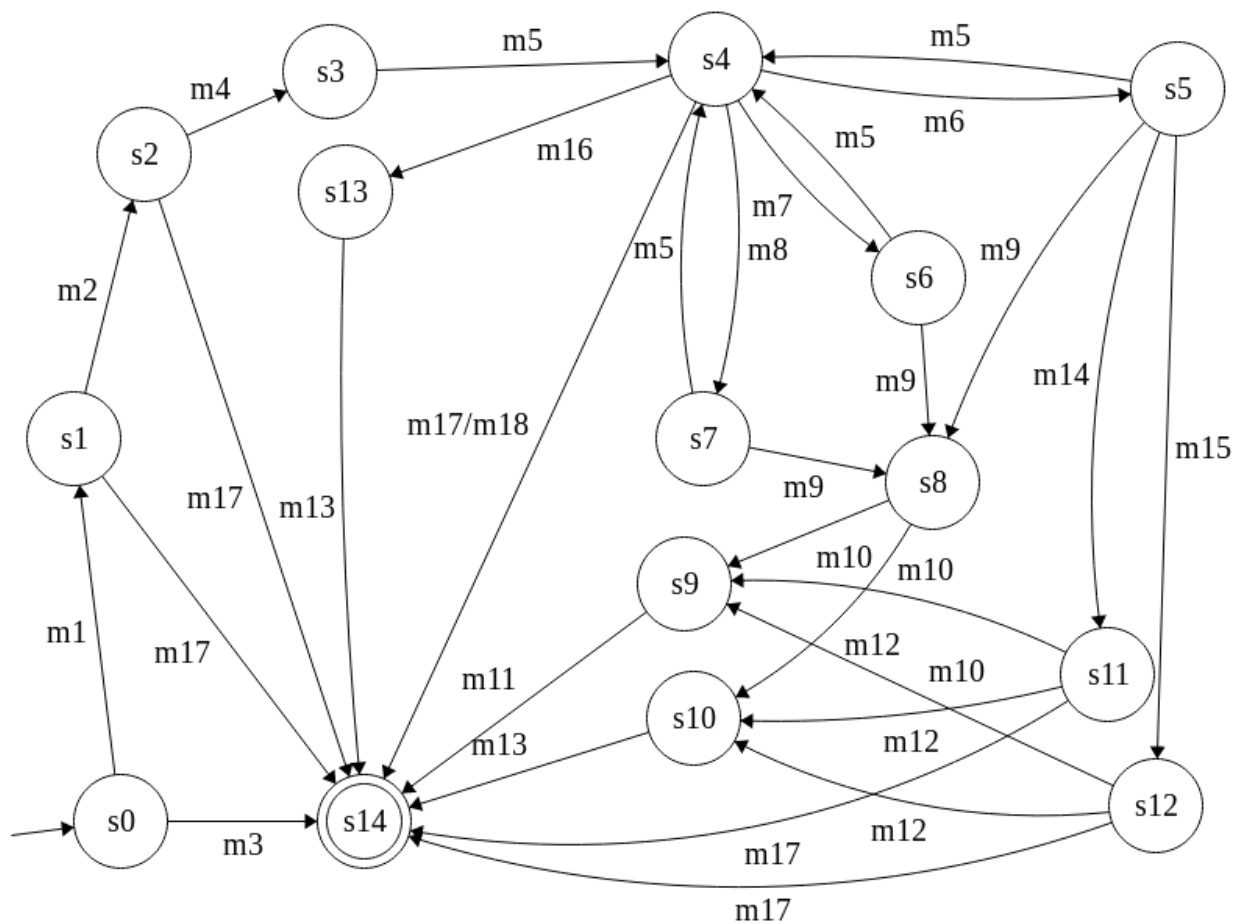
Estado	Descripción
s0	Estado inicial, esperando conexión entrante
s1	Se verifica la existencia del usuario de una solicitud
s2	Se verificó que el usuario existe, se establece conexión
s3	Se solicita un pokémon a capturar
s4	Se obtiene el pokemon y se espera a las acciones del cliente
s5	Intento de captura, se reduce el número de pokebolas y se espera respuesta (escape, captura, número insuficiente de pokebolas o continuación de ciclo)
s6	Se arroja, piedra, se espera respuesta de escape o continuación de ciclo
s7	Se arroja carnada, se espera respuesta de escape o continuación de ciclo
s8	El pokémon escapa, se espera acción del cliente
s9	El cliente quiere ver su pokédex, se obtienen datos de envío.
s10	El cliente no desea ver el pokedex, se prepara el término de sesión.
s11	El pokémon se ha capturado, se espera acción del cliente.
s12	Intentos de captura agotados, se espera acción del cliente.
s13	Se escapa del encuentro, se prepara el término de la sesión.
s14	Cierre de conexión

---

## Códigos

Código	Descripción
<i>m1</i> (15 - <b>TRY_USER</b> )	Cliente - Verificación de nombre de usuario (1 byte código, <b>n</b> bytes nombre)
<i>m2</i> (20 - <b>USER</b> )	Servidor - Confirma que el usuario existe (1 byte)
<i>m3</i> (41 - <b>NO_USER</b> )	Servidor - No existe el usuario (1 byte)
<i>m4</i> (10 - <b>ASK_POK</b> )	Cliente - Solicita un pokémon a capturar (1 byte)
<i>m5</i> (21 - <b>ENCOUNTER</b> )	Servidor - Estado de captura (Pokémon y número de pokebolas) (1 byte código, 1 byte id_pokemon, 1 byte num_pkbll))
<i>m6</i> (11 - <b>BALL</b> )	Cliente - Intento de captura con pokebola (1 byte)
<i>m7</i> (12 - <b>ROCK</b> )	Cliente - Arroja piedra (+ probabilidad de atrapar/huida) (1 byte)
<i>m8</i> (13 - <b>BAIT</b> )	Cliente - Arroja carnada (- probabilidad de atrapar/huida) (1 byte)
<i>m9</i> (22 - <b>ESCAPE</b> )	Servidor - El pokemon escapa (1 byte)
<i>m10</i> (30 - <b>OK</b> )	Cliente - Desea ver pokédex (1 byte)
<i>m11</i> (25 - <b>POKEDEX</b> )	Servidor - Regresa el pokedex del usuario (1 byte código, <b>n</b> bytes pokédex)
<i>m12</i> (31 - <b>NO</b> )	Cliente - No desea ver pokédex (1 byte)
<i>m13</i> (32 - <b>END_SESSION</b> )	Servidor/Cliente - Término de sesión (1 byte)
<i>m14</i> (23 - <b>SUCCESS</b> )	Servidor - Se captura pokémon (1 byte código, <b>n</b> bytes img_pokemon)

<i>m15</i> (24 - <b>NO_TRIES</b> )	Servidor - Pokebolas agotadas (1 byte)
<i>m16</i> (14 - <b>RUN</b> )	Cliente - Se escapa del encuentro (1 byte)
<i>m17</i> (42 - <b>ERROR_CODE</b> )	Cliente/Servidor - Código no esperado (Error) (1 byte)
<i>m18</i> (40 - <b>TIMEOUT</b> )	Cliente/Servidor - Se acaba el tiempo de espera, se corta conexión. (1 byte)



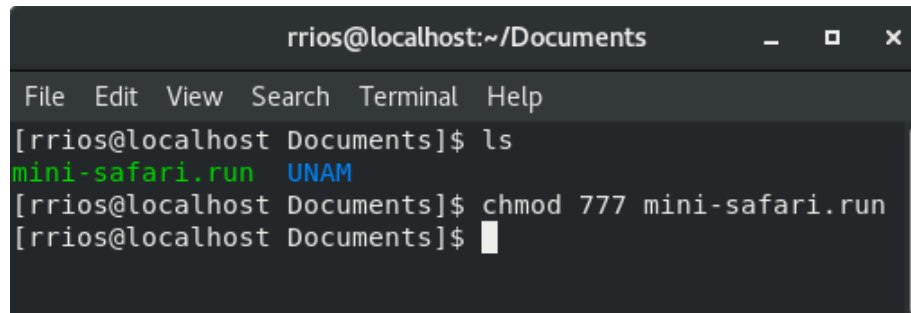
FSM del protocolo de capa de aplicación

---

## Uso

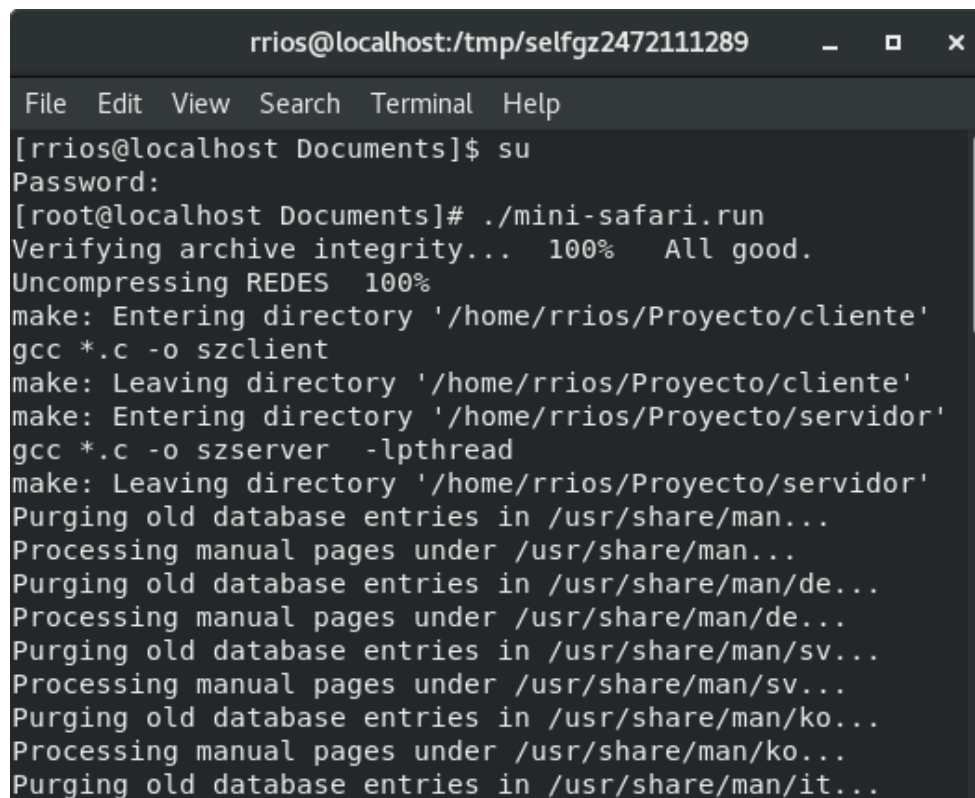
### a) Instalación (Cliente/Servidor)

Accedemos a la ubicación del archivo **mini-safari.run** y le asignamos permisos generales de uso con **chmod 777 mini-safari.run**

A terminal window titled 'rrios@localhost:~/Documents' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the following commands and output:

```
[rrios@localhost Documents]$ ls
mini-safari.run  UNAM
[rrios@localhost Documents]$ chmod 777 mini-safari.run
[rrios@localhost Documents]$
```

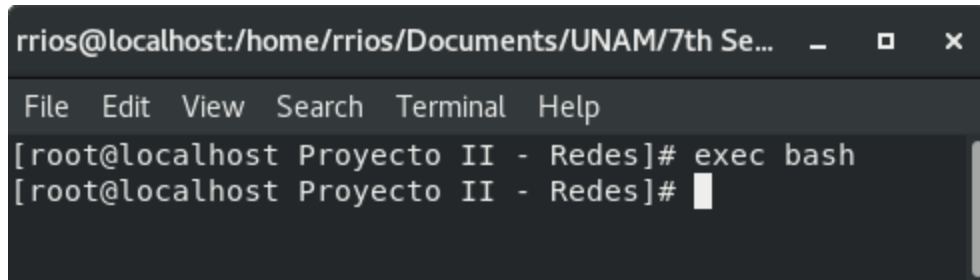
Ahora necesitamos ejecutar 2 veces el archivo, una vez en modo **su**

A terminal window titled 'rrios@localhost:/tmp/selfgz2472111289' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the execution of 'mini-safari.run' as root:

```
[rrios@localhost Documents]$ su
Password:
[root@localhost Documents]# ./mini-safari.run
Verifying archive integrity... 100% All good.
Uncompressing REDES 100%
make: Entering directory '/home/rrios/Proyecto/cliente'
gcc *.c -o szclient
make: Leaving directory '/home/rrios/Proyecto/cliente'
make: Entering directory '/home/rrios/Proyecto/servidor'
gcc *.c -o szserver -lpthread
make: Leaving directory '/home/rrios/Proyecto/servidor'
Purging old database entries in /usr/share/man...
Processing manual pages under /usr/share/man...
Purging old database entries in /usr/share/man/de...
Processing manual pages under /usr/share/man/de...
Purging old database entries in /usr/share/man/sv...
Processing manual pages under /usr/share/man/sv...
Purging old database entries in /usr/share/man/ko...
Processing manual pages under /usr/share/man/ko...
Purging old database entries in /usr/share/man/it...
```

---

Seguido del comando **exec bash**

A terminal window with a dark background. The title bar shows the user 'rrios' at 'localhost' in the directory '/home/rrios/Documents/UNAM/7th Se...'. The menu bar includes 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal content shows a root prompt '[root@localhost Proyecto II - Redes]#' followed by the command 'exec bash'. The next line shows the prompt has changed to '[root@localhost Proyecto II - Redes]#' with a white cursor block.

Análogamente en modo *normal*, esto se hace por que se necesitan permisos de **su** para copiar los manuales a las carpetas necesarias de **man**, y se cambia la variable de entorno **\$PATH** para que las aplicaciones sean accedidas globalmente, sin embargo al estar en modo **su** el archivo donde se actualiza no es el mismo del usuario normal.

Si no se ejecuta en modo *normal* (donde algunas operaciones ya no podrán ser realizadas, pero no son relevantes en ese punto), las aplicaciones solo podrán ser ejecutadas como **su**.

Esta primer instalación fue hecha en **Fedora 28** y la carpeta en donde están instaladas las aplicaciones es: **/home/\$USER/Proyecto**, para saber que valor tiene **\$USER** en su sistema, solo se necesita ejecutar el comando **echo \$USER**.

Una vez que termina la instalación, se puede acceder globalmente (en modo **su**) a cualquiera de las dos aplicaciones, y también a los manuales (**man**) en cualquier modo con los comandos **man szclient** y **man szserver**:

```
rrios@localhost:~  
File Edit View Search Terminal Help  
man(6)                                szclient man page                                man(6)  
  
NAME  
    szclient - Inicia una cliente para conexion a szserver  
  
SYNOPSIS  
    szclient [USUARIO] [DIRECCION IP DE SERVIDOR] [PUERTO: 9999]  
  
DESCRIPTION  
    szclient es un pequeño programa que permite la simulacion de el poke-safari en con-  
    sola.  
  
OPTIONS  
    No Hay opciones adicionales  
  
SEE ALSO  
    szserver(6)  
  
BUGS  
    No hay conocidos  
  
AUTHOR  
    (diegofavela@ciencias.unam.mx,      777coatl@ciencias.unam.mx,      fernandofong@cien-  
    cias.unam.mx)  
  
1.0                                15 May 2018                                man(6)  
Manual page szclient(6) line 1/26 (END) (press h for help or q to quit)
```

```
rrios@localhost:~  
File Edit View Search Terminal Help  
man(6)                                szserver man page                                man(6)  
  
NAME  
    szserver - Inicia un servidor para los szclient  
  
SYNOPSIS  
    szserver  
  
DESCRIPTION  
    szserver es un programa que actua como servidor para una aplicacion que simula el  
    mini-safari Pokémon  
  
OPTIONS  
    No Hay opciones adicionales  
  
SEE ALSO  
    szclient (6)  
  
BUGS  
    No hay conocidos  
  
AUTHOR  
    (diegofavela@ciencias.unam.mx,      777coatl@ciencias.unam.mx,      fernandofong@cien-  
    cias.unam.mx)  
  
1.0                                15 May 2018                                man(6)  
Manual page szserver(6) line 1 (press h for help or q to quit)
```

---

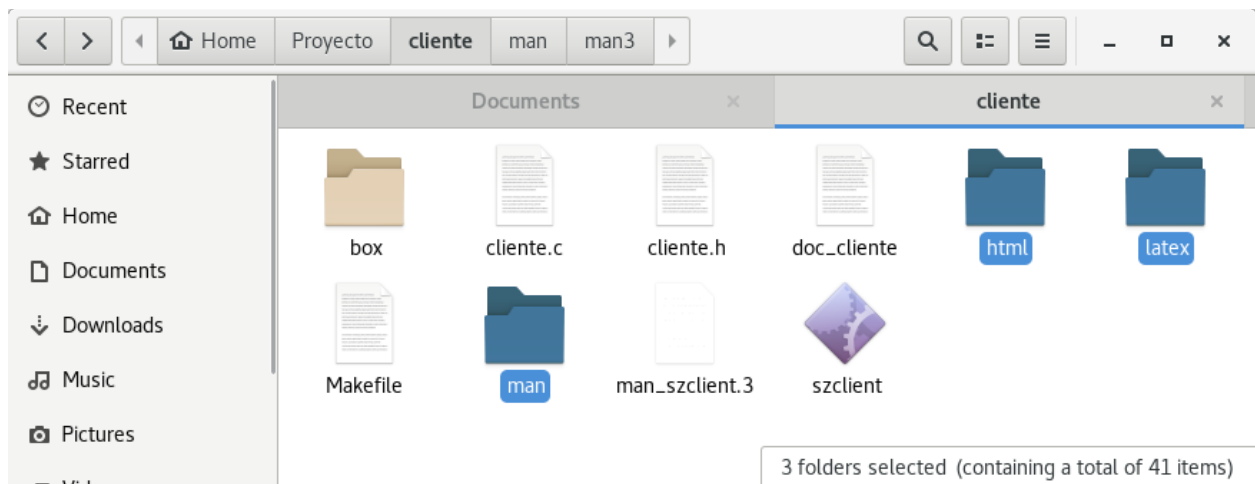
## b) Generación de Documentación (*opcional*)

La carpeta de instalación siempre es **/home/<usuario>/Proyecto**, dentro de esta carpeta se encuentra el desinstalador **uninstall.sh** y dos carpetas mas, correspondientes a cada aplicación: *cliente* y *servidor*.

En cada una de estas carpetas existe un *Makefile* que permite compilar, generar documentación y limpiar la carpeta (sin programas, solo código fuente), la documentación puede generarse en 3 diferentes formatos: *HTML*, *LaTeX* y *MAN*, pero es necesario tener instalado el programa **doxygen**, las instrucciones de instalación se encuentran [aquí](#).

Para obtener la documentación solo debemos irnos a la carpeta deseada y ejecutar el comando **make doc**, esto generará 3 carpetas mas llamadas: *html*, *latex* y *man*.

En la carpeta *html* se encuentra un archivo llamado *index.html*, el cual puede ser visualizado en un navegador y obtener la información requerida de las funciones de las aplicaciones, en la carpeta *latex* se puede generar un *PDF* con el archivo *Makefile* interno, pero se necesita tener una distribución bien equipada de *LaTeX* para una generación exitosa con el comando **make**, por último la carpeta de *man* tiene una subcarpeta en donde se encuentran archivos con extension .3, estos archivos se pueden visualizar desde consola con el siguiente comando: **man ./<archivo>.3**





```
rrios@localhost:~/Documents/UNAM/7th Semester/Proyecto II - Redes/Proyecto/cliente
File Edit View Search Terminal Help
[rrios@localhost cliente]$ make doc
doxygen doc_cliente
Searching for include files...
Searching for example files...
Searching for images...
Searching for dot files...
Searching for msc files...
Searching for dia files...
Searching for files to exclude
Searching INPUT for files to process...
Searching for files in directory /home/rrios/Documents/UNAM/7th Semester/Proyecto II - Redes/Proyecto/cliente
Reading and parsing tag files
Parsing files
Preprocessing /home/rrios/Documents/UNAM/7th Semester/Proyecto II - Redes/Proyecto/cliente/cliente.c...
Parsing file /home/rrios/Documents/UNAM/7th Semester/Proyecto II - Redes/Proyecto/cliente/cliente.c...
Preprocessing /home/rrios/Documents/UNAM/7th Semester/Proyecto II - Redes/Proyecto/cliente/cliente.h...
Parsing file /home/rrios/Documents/UNAM/7th Semester/Proyecto II - Redes/Proyecto/cliente/cliente.h...
Building group list...
Building directory list...
Building namespace list...
Building file list...
Building class list...
Associating documentation with classes...
Computing nesting relations for classes...
Building example list...
Searching for enumerations...
Searching for documented typedefs...
Searching for members imported via using declarations...
Searching for included using directives...
Searching for documented variables...
Building interface member list...
Building member list...
Searching for friends...
Searching for documented defines...
Computing class inheritance relations...
Computing class usage relations...
Flushing cached template relations that have become invalid...
Computing class relations...
Add enum values to enums...
Searching for member function documentation...
Creating members for template instances...
Building page list...
Search for main page...
Computing page relations...
Determining the scope of groups...
Sorting lists...
Freeing entry tree
```

### c) Ejecución

Los programas pueden ser ejecutados en el mismo equipo, solo tenemos que conocer la dirección **IP** del que funcionara como servidor, en este ejemplo se usarán dos máquinas virtuales en **VMWare** y el dispositivo físico para ver las capacidades referentes a hilos de ejecución.

Una máquina virtual será el servidor, mientras la otra y el equipo físico servirán como clientes, la configuración de las máquinas virtuales es la siguiente:

La red Virtual que usamos tiene la siguiente configuración:

The screenshot shows the 'Virtual Network Editor' window. It contains a table with network configurations and a detailed configuration section for 'vmnet7'.

Name	Type	External Connection	Host Connection	DHCP	Subnet IP Address
vmnet0	bridged	auto-bridging	—	—	—
vmnet1	host-only	none	vmnet1	yes	192.168.149.0
vmnet7	host-only	none	vmnet7	no	192.168.1.0
vmnet8	NAT	NAT	vmnet8	yes	192.168.168.0

Buttons: + Add Network... - Remove Network

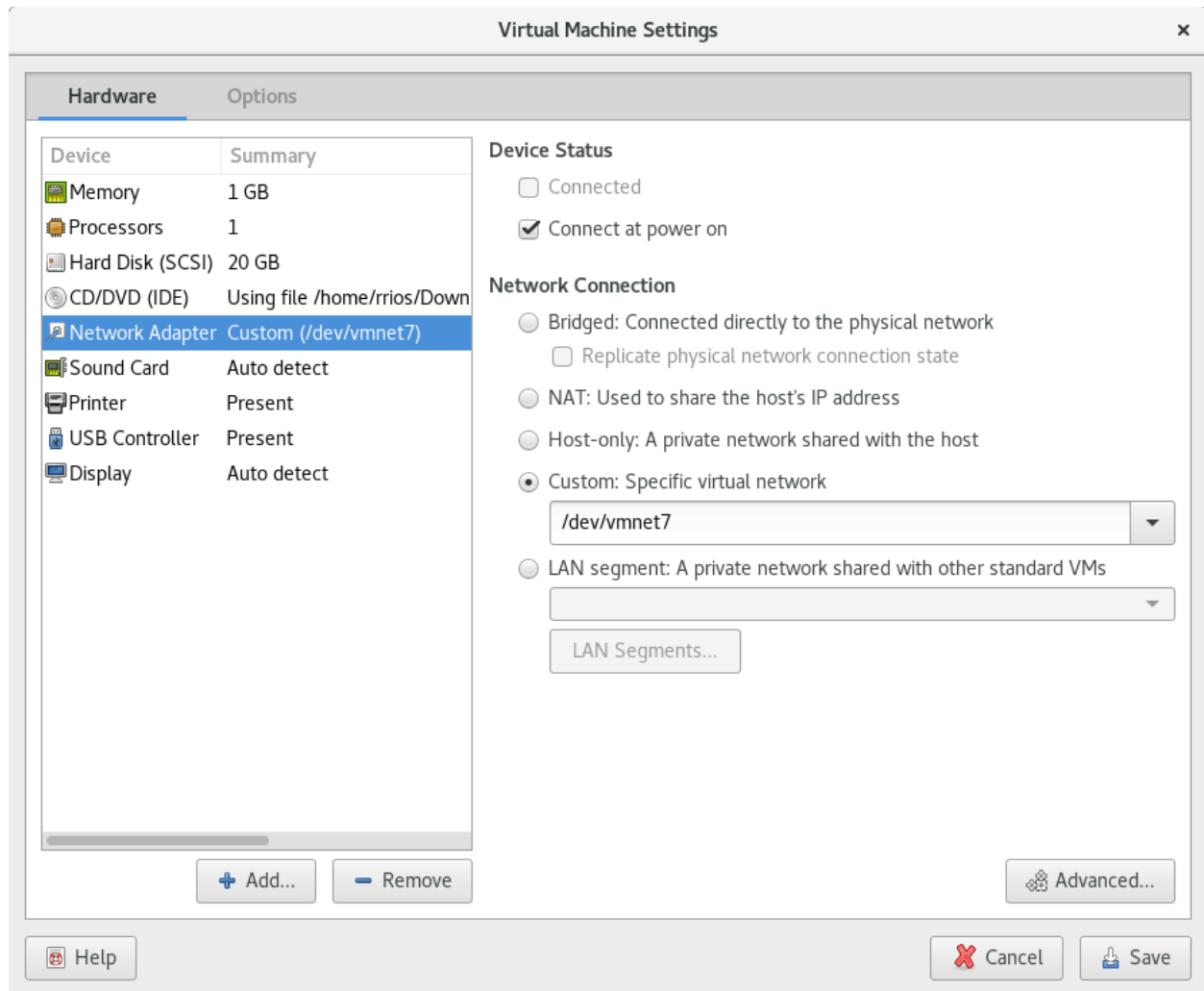
**vmnet7**

- ☐ Bridged (connect VMs directly to the external network)  
Bridged to: Automatic Automatic Settings...
- ☐ NAT (share host's IP address with VMs) NAT Settings...
- ☒ Host-only (connect VMs internally in a private network)
- ☐ Use local DHCP service to distribute IP addresses to VMs
- ☒ Connect a host virtual adapter (vmnet7) to this network

Subnet IP: 192.168. 1 . 0 Subnet mask: 255.255.255. 0  
💡 Leave blank to automatically select an unused subnet IP.

Buttons: Help Cancel Save

Las máquinas virtuales usan esta configuración de red:



Ambas usan **Debian 9**, mientras que el equipo físico usa **Fedora 28**.

Ahora pasaremos a ver el funcionamiento de los programas y algunos casos de interés, las direcciones **IP** de los dispositivos son:

Cliente A (Máquina Virtual):

```
wwhite@debian2: ~/Documents/Proyecto II - Redes/Proyecto/servidor
File Edit View Search Terminal Help
wwhite@debian2:~/Documents/Proyecto II - Redes/Proyecto/servidor$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:d9:16:0e brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.150/24 brd 192.168.1.255 scope global ens33
        valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fed9:160e/64 scope link
        valid_lft forever preferred_lft forever
wwhite@debian2:~/Documents/Proyecto II - Redes/Proyecto/servidor$
```

Cliente B (Equipo Físico):

```
rrios@localhost:~/Documents/UNAM/7th Semester/Proyecto II - Redes/Proyecto/cliente
File Edit View Search Terminal Help
    inet 192.168.149.1/24 brd 192.168.149.255 scope global vmnet1
        valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:fec0:1/64 scope link
        valid_lft forever preferred_lft forever
4: vmnet7: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN group default qlen 1000
    link/ether 00:50:56:c0:00:07 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.1/24 brd 192.168.1.255 scope global vmnet7
        valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:fec0:7/64 scope link
        valid_lft forever preferred_lft forever
5: vmnet8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN group default qlen 1000
    link/ether 00:50:56:c0:00:08 brd ff:ff:ff:ff:ff:ff
    inet 192.168.168.1/24 brd 192.168.168.255 scope global vmnet8
        valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:fec0:8/64 scope link
        valid_lft forever preferred_lft forever
6: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default qlen 1000
    link/ether 52:54:00:4a:48:70 brd ff:ff:ff:ff:ff:ff
```

Servidor (Máquina Virtual):

```
tmontana@debian1: ~/Documents/Proyecto II - Redes/Proyecto/cliente
File Edit View Search Terminal Help
tmontana@debian1:~/Documents/Proyecto II - Redes/Proyecto/cliente$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:c7:18:4a brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.200/24 brd 192.168.1.255 scope global ens33
        valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fec7:184a/64 scope link
        valid_lft forever preferred_lft forever
tmontana@debian1:~/Documents/Proyecto II - Redes/Proyecto/cliente$
```

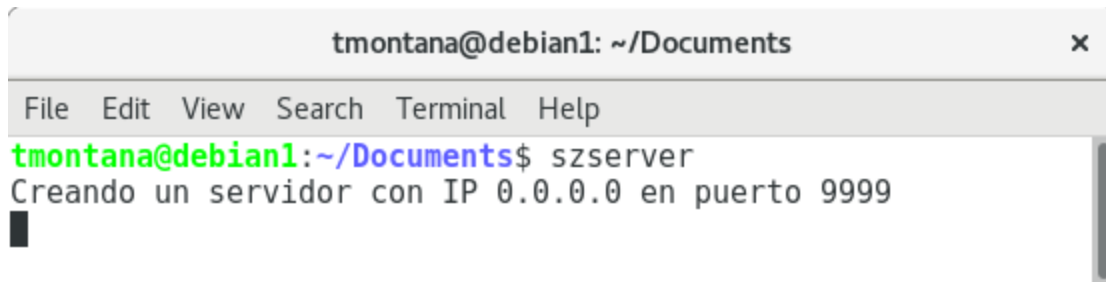
Se realiza la instalación de la misma manera que en el equipo físico en las máquinas, aquí es **crucial** la instalación adicional en modo normal ya que **Debian** no traduce a `$USER` de la misma manera que **Fedora** en modo **su**, (**Fedora**: `# echo $USER -> usuario normal`, **Debian**: `# echo $USER -> root`).

Dado los permisos de escritura y lectura, si no se realiza esto, el cliente y el servidor **no** podrán transmitirse paquetes exitosamente.

```
tmontana@debian1: ~/Documents
File Edit View Search Terminal Help
tmontana@debian1:~/Documents$ ./mini-safari.run
Verifying archive integrity... 100% All good.
Uncompressing REDES 100%
make: Entering directory '/home/tmontana/Proyecto/cliente'
gcc *.c -o szclient
make: Leaving directory '/home/tmontana/Proyecto/cliente'
make: Entering directory '/home/tmontana/Proyecto/servidor'
gcc *.c -o szserver -lpthread
make: Leaving directory '/home/tmontana/Proyecto/servidor'
cp: cannot create regular file '/usr/share/man/man6/szclient.6.gz': Permission denied
cp: cannot create regular file '/usr/share/man/man6/szserver.6.gz': Permission denied
0 man subdirectories contained newer manual pages.
0 manual pages were added.
0 stray cats were added.
0 old database entries were purged.
tmontana@debian1:~/Documents$ exec bash
tmontana@debian1:~/Documents$
```

---

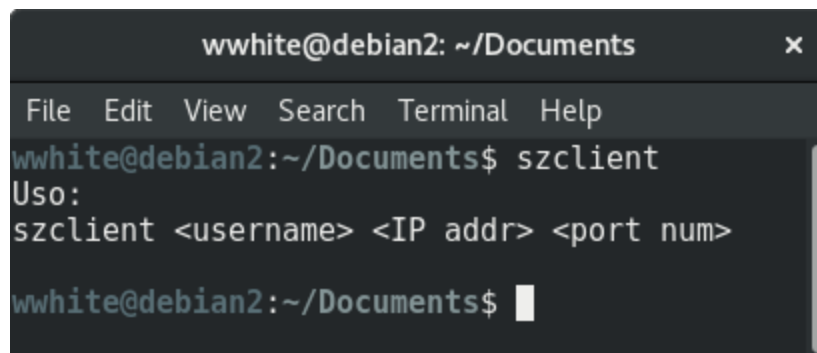
Ahora compilamos y ejecutamos el servidor:



```
tmontana@debian1: ~/Documents
File Edit View Search Terminal Help
tmontana@debian1:~/Documents$ szserver
Creando un servidor con IP 0.0.0.0 en puerto 9999
```

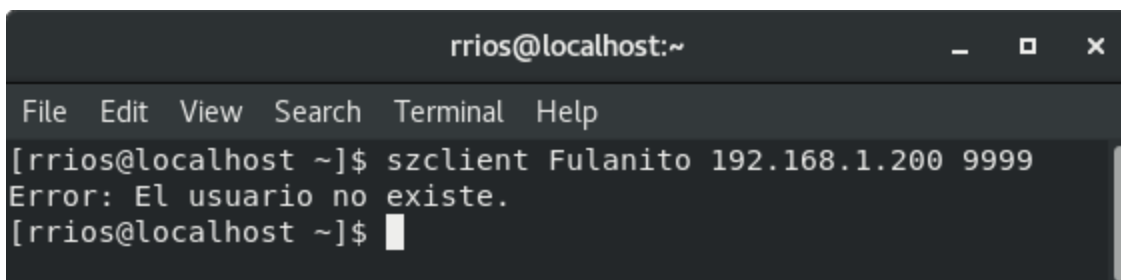
El servidor ahora nos muestra su estado, la **IP** es 0.0.0.0 por que esta usando la local, ahora veamos que pasa cuando intentamos conectarnos con los clientes de manera correcta e incorrecta...

Si no ponemos ningún dato, nos enseña el mensaje de uso:

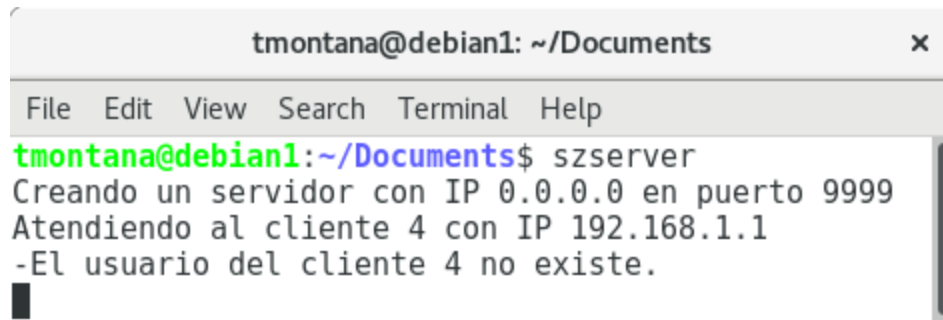


```
wwhite@debian2: ~/Documents
File Edit View Search Terminal Help
wwhite@debian2:~/Documents$ szclient
Uso:
szclient <username> <IP addr> <port num>
wwhite@debian2:~/Documents$
```

Si ponemos un usuario que no existe:

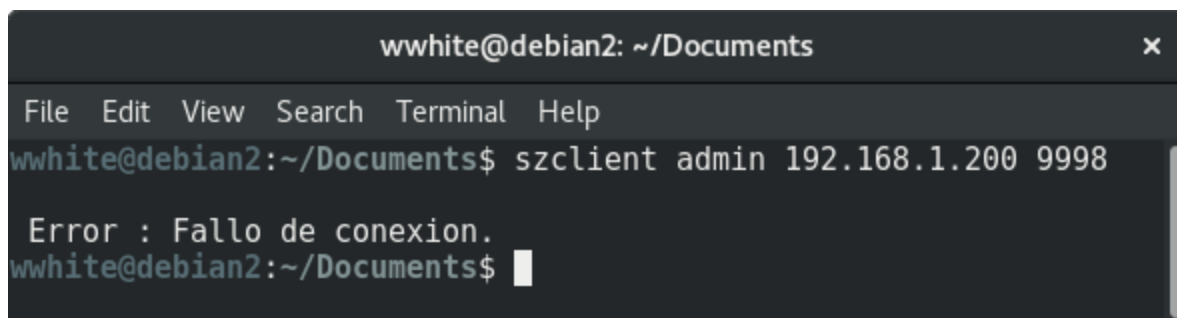


```
rrios@localhost:~
File Edit View Search Terminal Help
[rrios@localhost ~]$ szclient Fulanito 192.168.1.200 9999
Error: El usuario no existe.
[rrios@localhost ~]$
```



```
tmontana@debian1: ~/Documents
File Edit View Search Terminal Help
tmontana@debian1:~/Documents$ szserver
Creando un servidor con IP 0.0.0.0 en puerto 9999
Atendiendo al cliente 4 con IP 192.168.1.1
-El usuario del cliente 4 no existe.
```

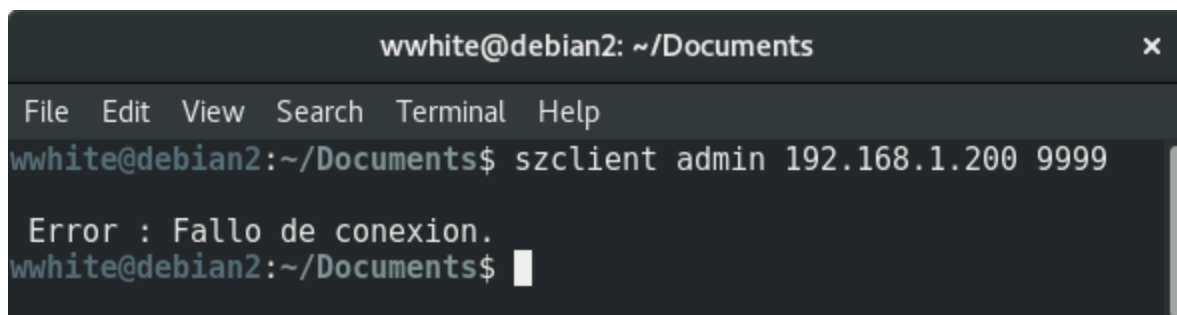
Si ponemos un puerto incorrecto:



```
wwhite@debian2: ~/Documents
File Edit View Search Terminal Help
wwhite@debian2:~/Documents$ szclient admin 192.168.1.200 9998

Error : Fallo de conexion.
wwhite@debian2:~/Documents$
```

Si en la dirección ingresada, no existe un servidor escuchando (después de unos segundos):

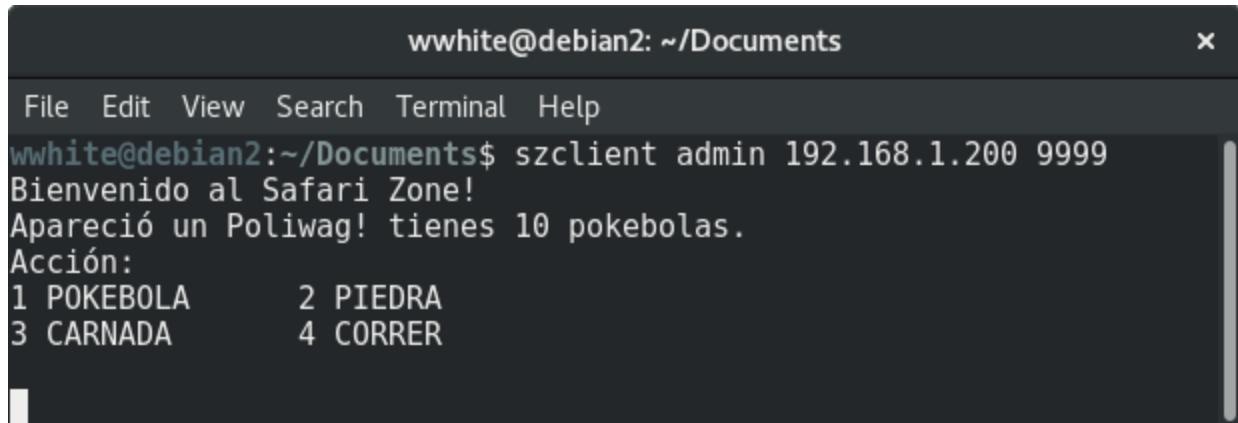


```
wwhite@debian2: ~/Documents
File Edit View Search Terminal Help
wwhite@debian2:~/Documents$ szclient admin 192.168.1.200 9999

Error : Fallo de conexion.
wwhite@debian2:~/Documents$
```

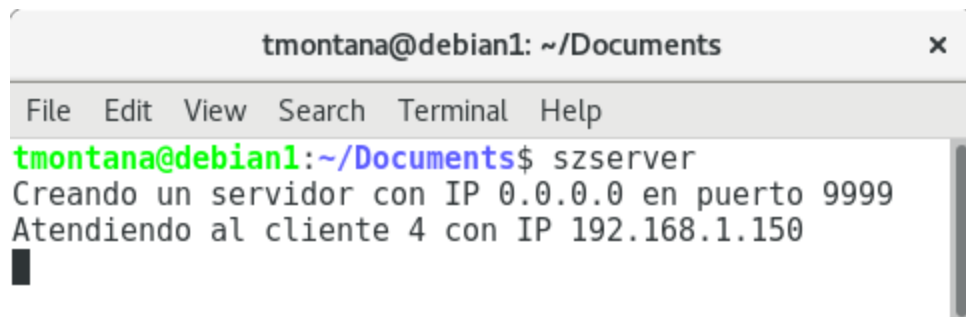
---

Ahora veamos ingresos correctos:



```
wwhite@debian2: ~/Documents
File Edit View Search Terminal Help
wwhite@debian2:~/Documents$ szclient admin 192.168.1.200 9999
Bienvenido al Safari Zone!
Apareció un Poliwag! tienes 10 pokebolas.
Acción:
1 POKEBOLA      2 PIEDRA
3 CARNADA       4 CORRER
```

Una vez que se realiza la conexión, se muestra el menú de eventos, y del lado del servidor podemos ver el número asignado al cliente y su dirección **IP**.



```
tmontana@debian1: ~/Documents
File Edit View Search Terminal Help
tmontana@debian1:~/Documents$ szserver
Creando un servidor con IP 0.0.0.0 en puerto 9999
Atendiendo al cliente 4 con IP 192.168.1.150
```



Ahora una interacción:

```
wwhite@debian2: ~/Documents
File Edit View Search Terminal Help
wwhite@debian2:~/Documents$ szclient admin 192.168.1.200 9999
Bienvenido al Safari Zone!
Apareció un Poliwag! tienes 10 pokebolas.
Acción:
1 POKEBOLA      2 PIEDRA
3 CARNADA       4 CORRER

1
Poliwag está observando cuidadosamente, tienes 9 pokebolas.
Acción:
1 POKEBOLA      2 PIEDRA
3 CARNADA       4 CORRER

2
Poliwag está observando cuidadosamente, tienes 9 pokebolas.
Acción:
1 POKEBOLA      2 PIEDRA
3 CARNADA       4 CORRER

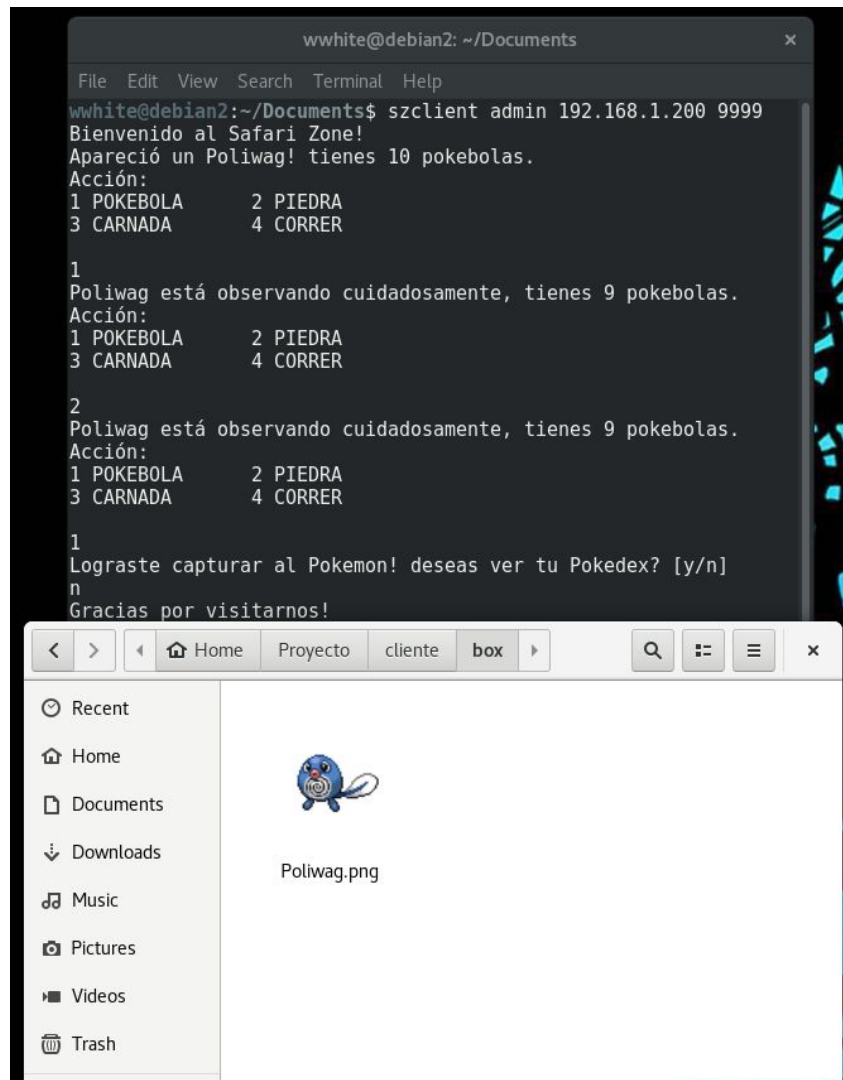
1
Lograste capturar al Pokemon! deseas ver tu Pokedex? [y/n]
n
Gracias por visitarnos!
wwhite@debian2:~/Documents$
```

Cuando se captura un pokémon, se pregunta si se quiere ver el pokédex, en cualquier caso, la conexión termina después de eso y se muestran mensajes correspondientes del lado del cliente y del lado del servidor:

```
tmontana@debian1: ~/Documents
File Edit View Search Terminal Help
tmontana@debian1:~/Documents$ szserver
Creando un servidor con IP 0.0.0.0 en puerto 9999
Atendiendo al cliente 4 con IP 192.168.1.150
Desconectandose del cliente 4.
```

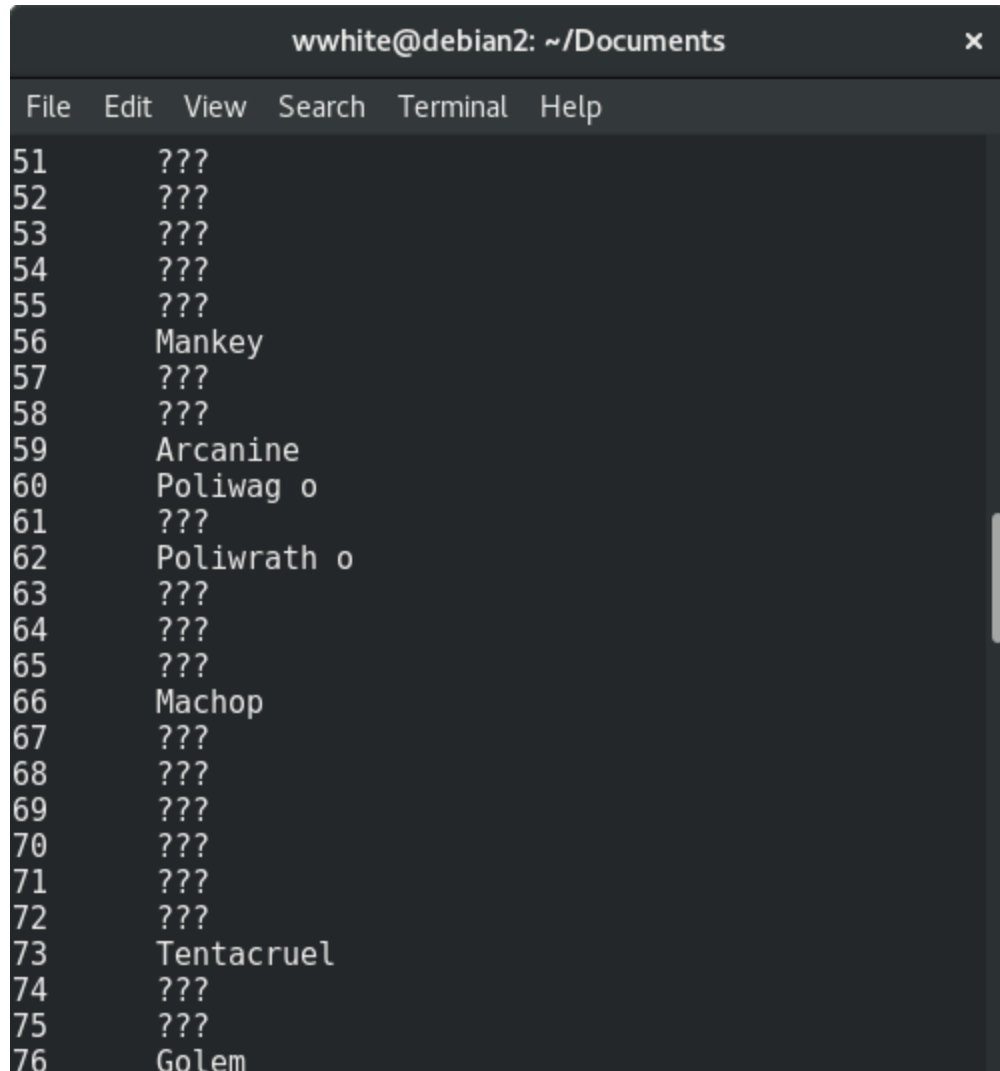
Podemos observar que al principio arrojamus una pokebola y fallo al capturar, entonces se nos da otra oportunidad por que el pokémon no escapa y se reduce nuestro número de pokebolas, después arrojamus una piedra, lo cual aumenta la posibilidad de que el pokemon escape, pero también aumenta la probabilidad de que sea capturado, arrojar carnada disminuye la probabilidad de que el pokemon huya, pero de la misma manera disminuye la probabilidad de que sea capturado.

La segunda pokebola fue un éxito, por lo que nos pregunta si queremos ver el pokedex, otra cosa que se debe notar es que cuando se atrapa un pokemon, el servidor manda la imagen correspondiente y se guarda de nuestro lado en el directorio **/home/\$USER/Proyecto/cliente/box/** como se muestra:



---

Si elegimos mostrar el pokedex, nos sale algo asi:

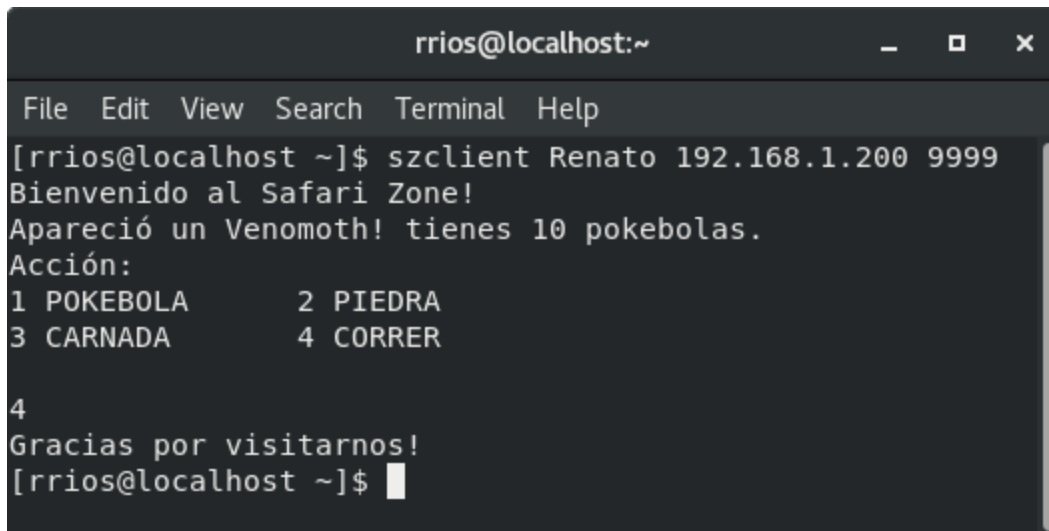


```
wwhite@debian2: ~/Documents
File Edit View Search Terminal Help
51      ???
52      ???
53      ???
54      ???
55      ???
56      Mankey
57      ???
58      ???
59      Arcanine
60      Poliwag o
61      ???
62      Poliwrath o
63      ???
64      ???
65      ???
66      Machop
67      ???
68      ???
69      ???
70      ???
71      ???
72      ???
73      Tentacruel
74      ???
75      ???
76      Golem
```

Observamos que el pokemon que acabamos de atrapar tiene un pequeño círculo al lado, eso indica que lo hemos visto y lo hemos atrapado, en el caso que no tengan ese círculo, es que solo los hemos visto, pero fallamos en atraparlo, y por último los que tienen ??? son los que nunca hemos visto. De igual manera el servidor termina la conexión después de enseñar el pokedex.

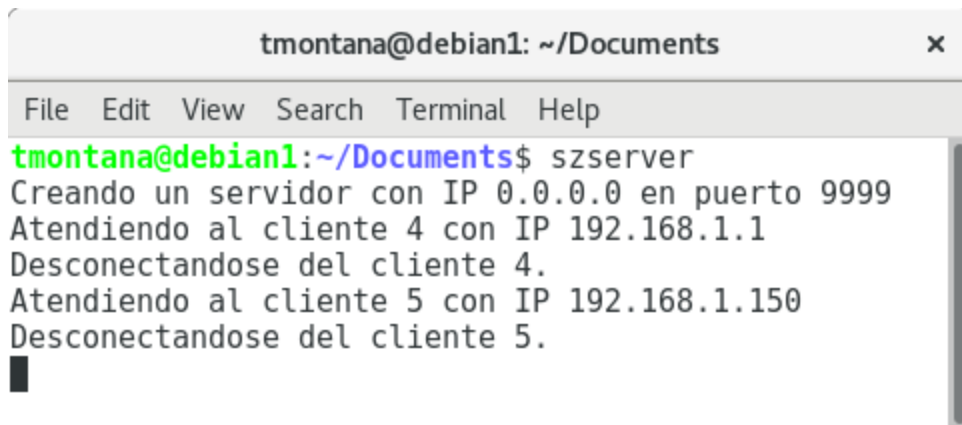
---

El otro evento que podemos ver es correr del encuentro, fallar la captura y terminar la conexión:



```
rrios@localhost:~  
File Edit View Search Terminal Help  
[rrios@localhost ~]$ szclient Renato 192.168.1.200 9999  
Bienvenido al Safari Zone!  
Apareció un Venomoth! tienes 10 pokebolas.  
Acción:  
1 POKEBOLA      2 PIEDRA  
3 CARNADA       4 CORRER  
  
4  
Gracias por visitarnos!  
[rrios@localhost ~]$
```

En este punto se conectaron un el cliente *B* y después de que termino se conecto el cliente *A*.



```
tmontana@debian1: ~/Documents  
File Edit View Search Terminal Help  
tmontana@debian1:~/Documents$ szserver  
Creando un servidor con IP 0.0.0.0 en puerto 9999  
Atendiendo al cliente 4 con IP 192.168.1.1  
Desconectandose del cliente 4.  
Atendiendo al cliente 5 con IP 192.168.1.150  
Desconectandose del cliente 5.  
█
```

Podemos ver como a cada conexión le asigna un número de cliente, aun cuando anteriormente vengan de la misma **IP**.

---

También podemos encontrarnos con situaciones en donde el pokemon escapa:

```
rrios@localhost:~  
File Edit View Search Terminal Help  
[rrios@localhost ~]$ szclient Renato 192.168.1.200 9999  
Bienvenido al Safari Zone!  
Apareció un Krabby! tienes 10 pokebolas.  
Acción:  
1 POKEBOLA          2 PIEDRA  
3 CARNADA           4 CORRER  
  
2  
Oh! Se escapó el Pokemon, deseas ver tu Pokedex? [y/n]  
n  
Gracias por visitarnos!  
[rrios@localhost ~]$
```

Por último veamos que pasa cuando no hay ningún tipo de comunicación de parte del cliente una vez conectado, al implementarse *timeouts*, el servidor corta la conexión después de cierto tiempo de inactividad, aproximadamente 1 minuto:

```
tmontana@debian1: ~/Documents  
File Edit View Search Terminal Help  
tmontana@debian1:~/Documents$ szserver  
Creando un servidor con IP 0.0.0.0 en puerto 9999  
Atendiendo al cliente 4 con IP 192.168.1.150  
-No hubo respuesta del cliente 4.  
■
```

Y si el cliente intenta alguna acción después de esto:

```
wwhite@debian2: ~/Documents  
File Edit View Search Terminal Help  
wwhite@debian2:~/Documents$ szclient admin 192.168.1.200 9999  
Bienvenido al Safari Zone!  
Apareció un Kadabra! tienes 10 pokebolas.  
Acción:  
1 POKEBOLA          2 PIEDRA  
3 CARNADA           4 CORRER  
  
1  
Error: Se desconectó el servidor.  
wwhite@debian2:~/Documents$
```

Por supuesto que el programa admite tener varias conexiones al mismo tiempo gracias a el uso de hilos, como se puede ver en estas capturas:

```
wwhite@debian2: ~/Documents
File Edit View Search Terminal Help
wwhite@debian2:~/Documents$ szclient Joan 192.168.1.200 9999
Bienvenido al Safari Zone!
Apareció un Slowbro! tienes 10 pokebolas.
Acción:
1 POKEBOLA      2 PIEDRA
3 CARNADA       4 CORRER

4
Gracias por visitarnos!
wwhite@debian2:~/Documents$
```

```
rrios@localhost:~
File Edit View Search Terminal Help
[rrios@localhost ~]$ szclient Anna 192.168.1.200 9999
Bienvenido al Safari Zone!
Apareció un Clefairy! tienes 10 pokebolas.
Acción:
1 POKEBOLA      2 PIEDRA
3 CARNADA       4 CORRER

2
Clefairy está observando cuidadosamente, tienes 10 pokebolas.
Acción:
1 POKEBOLA      2 PIEDRA
3 CARNADA       4 CORRER

1
Oh! Se escapó el Pokemon, deseas ver tu Pokedex? [y/n]
n
Gracias por visitarnos!
[rrios@localhost ~]$
```

```
tmontana@debian1: ~/Documents
File Edit View Search Terminal Help
tmontana@debian1:~/Documents$ szserver
Creando un servidor con IP 0.0.0.0 en puerto 9999
Atendiendo al cliente 4 con IP 192.168.1.1
Atendiendo al cliente 5 con IP 192.168.1.150
Desconectandose del cliente 5.
Desconectandose del cliente 4.
```

El cliente B se conecta primero, seguido del A, pero el A termina antes sus eventos y le sigue el B, no hay interrupciones entre los dos por que cada uno recibe un hilo para utilizar.

A continuación se muestra una captura de **Wireshark** donde se conecta el cliente A (rojo) y después el B (azul), cada uno con su respectivo *3-way handshake*.

\*vmnet7

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp

Expression...

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.1.150	192.168.1.200	TCP	74	38356 → 9999 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2038778 TSecr=0 WS=128
2	0.000259695	192.168.1.200	192.168.1.150	TCP	74	9999 → 38356 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=2060325 TSecr=2038778 WS=128
3	0.000360277	192.168.1.150	192.168.1.200	TCP	66	38356 → 9999 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=2038778 TSecr=2060325
4	0.000443316	192.168.1.150	192.168.1.200	TCP	72	38356 → 9999 [PSH, ACK] Seq=1 Ack=1 Win=29312 Len=6 TSval=2038778 TSecr=2060325
5	0.000492464	192.168.1.200	192.168.1.150	TCP	66	9999 → 38356 [ACK] Seq=1 Ack=7 Win=29056 Len=0 TSval=2060325 TSecr=2038778
6	0.001314731	192.168.1.200	192.168.1.150	TCP	67	9999 → 38356 [PSH, ACK] Seq=1 Ack=7 Win=29056 Len=1 TSval=2060326 TSecr=2038778
7	0.001449960	192.168.1.150	192.168.1.200	TCP	66	38356 → 9999 [ACK] Seq=7 Ack=2 Win=29312 Len=0 TSval=2038778 TSecr=2060326
8	0.001515626	192.168.1.150	192.168.1.200	TCP	67	38356 → 9999 [PSH, ACK] Seq=7 Ack=2 Win=29312 Len=1 TSval=2038778 TSecr=2060326
9	0.001600863	192.168.1.200	192.168.1.150	TCP	69	9999 → 38356 [PSH, ACK] Seq=2 Ack=8 Win=29056 Len=3 TSval=2060326 TSecr=2038778
10	0.043179749	192.168.1.150	192.168.1.200	TCP	66	38356 → 9999 [ACK] Seq=8 Ack=5 Win=29312 Len=0 TSval=2038789 TSecr=2060326
15	6.349413945	192.168.1.1	192.168.1.200	TCP	74	34176 → 9999 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=1390524832 TSecr=0 WS=128
16	6.349608762	192.168.1.200	192.168.1.1	TCP	74	9999 → 34176 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=2061913 TSecr=1390524832 WS=128
17	6.349643593	192.168.1.1	192.168.1.200	TCP	66	34176 → 9999 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=1390524832 TSecr=2061913
18	6.349665753	192.168.1.1	192.168.1.200	TCP	72	34176 → 9999 [PSH, ACK] Seq=1 Ack=1 Win=29312 Len=6 TSval=1390524832 TSecr=2061913
19	6.349721001	192.168.1.200	192.168.1.1	TCP	66	9999 → 34176 [ACK] Seq=1 Ack=7 Win=29056 Len=0 TSval=2061913 TSecr=1390524832
20	6.350148389	192.168.1.200	192.168.1.1	TCP	67	9999 → 34176 [PSH, ACK] Seq=1 Ack=7 Win=29056 Len=1 TSval=2061913 TSecr=1390524832
21	6.350165155	192.168.1.1	192.168.1.200	TCP	66	34176 → 9999 [ACK] Seq=7 Ack=2 Win=29312 Len=0 TSval=1390524833 TSecr=2061913
22	6.350192368	192.168.1.1	192.168.1.200	TCP	67	34176 → 9999 [PSH, ACK] Seq=7 Ack=2 Win=29312 Len=1 TSval=1390524833 TSecr=2061913
23	6.350254527	192.168.1.200	192.168.1.1	TCP	69	9999 → 34176 [PSH, ACK] Seq=2 Ack=8 Win=29056 Len=3 TSval=2061913 TSecr=1390524833
24	6.390472281	192.168.1.1	192.168.1.200	TCP	66	34176 → 9999 [ACK] Seq=8 Ack=5 Win=29312 Len=0 TSval=1390524873 TSecr=2061913
29	14.562555297	192.168.1.1	192.168.1.200	TCP	67	34176 → 9999 [PSH, ACK] Seq=8 Ack=5 Win=29312 Len=1 TSval=1390533045 TSecr=2061913
30	14.562933784	192.168.1.200	192.168.1.1	TCP	69	9999 → 34176 [PSH, ACK] Seq=5 Ack=9 Win=29056 Len=3 TSval=2063966 TSecr=1390533045
31	14.562987366	192.168.1.1	192.168.1.200	TCP	66	34176 → 9999 [ACK] Seq=9 Ack=8 Win=29312 Len=0 TSval=1390533046 TSecr=2063966
32	61.289055449	192.168.1.200	192.168.1.150	TCP	67	9999 → 38356 [PSH, ACK] Seq=5 Ack=8 Win=29056 Len=1 TSval=2075648 TSecr=2038789
33	61.289573124	192.168.1.150	192.168.1.200	TCP	66	38356 → 9999 [ACK] Seq=8 Ack=6 Win=29312 Len=0 TSval=2054100 TSecr=2075648
34	62.289669380	192.168.1.200	192.168.1.150	TCP	66	9999 → 38356 [FIN, ACK] Seq=6 Ack=8 Win=29056 Len=0 TSval=2075898 TSecr=2054100
35	62.331117042	192.168.1.150	192.168.1.200	TCP	66	38356 → 9999 [ACK] Seq=8 Ack=7 Win=29312 Len=0 TSval=2054361 TSecr=2075898
38	75.624840822	192.168.1.200	192.168.1.1	TCP	67	9999 → 34176 [PSH, ACK] Seq=8 Ack=9 Win=29056 Len=1 TSval=2079232 TSecr=1390533046
39	75.624871364	192.168.1.1	192.168.1.200	TCP	66	34176 → 9999 [ACK] Seq=9 Ack=9 Win=29312 Len=0 TSval=1390594108 TSecr=2079232
40	76.625189718	192.168.1.200	192.168.1.1	TCP	66	9999 → 34176 [FIN, ACK] Seq=9 Ack=9 Win=29056 Len=0 TSval=2079482 TSecr=1390594108
41	76.665455316	192.168.1.1	192.168.1.200	TCP	66	34176 → 9999 [ACK] Seq=9 Ack=10 Win=29312 Len=0 TSval=1390595148 TSecr=2079482
46	151.258318046	192.168.1.150	192.168.1.200	TCP	67	38356 → 9999 [PSH, ACK] Seq=8 Ack=7 Win=29312 Len=1 TSval=2076592 TSecr=2075898
47	151.258663683	192.168.1.200	192.168.1.150	TCP	60	9999 → 38356 [RST] Seq=7 Win=0 Len=0
52	157.970691344	192.168.1.1	192.168.1.200	TCP	67	34176 → 9999 [PSH, ACK] Seq=9 Ack=10 Win=29312 Len=1 TSval=1390676454 TSecr=2079482

--Header checksum: 0xb3af [validation disabled]

--[Header checksum status: Unverified]

--Source: 192.168.1.200

0000 00 50 56 c0 00 07 00 0c 29 c7 18 4a 08 00 45 00 PV.....)J..E..

0010 00 28 03 07 40 00 40 06 b3 af c0 a8 01 c8 c0 a8 ..@\_@.....

0020 01 01 27 0f 85 80 da 09 df 39 00 00 00 00 50 04 .....9.....P.

Destination (ip.dst), 4 bytes

Packets: 57 · Displayed: 35 (61.4%)

Profile: Default

Se dejó que corriera el tiempo para que el servidor termina la conexión por *timeout*, aunque el cliente B lanzó una pokebola antes de eso. Ahora las capturas correspondientes a esta interacción (Cliente A, Cliente B y Servidor):

```
wwhite@debian2: ~/Documents
File Edit View Search Terminal Help
wwhite@debian2:~/Documents$ szclient Joan 192.168.1.200 9999
Bienvenido al Safari Zone!
Apareció un Raticate! tienes 10 pokebolas.
Acción:
1 POKEBOLA      2 PIEDRA
3 CARNADA       4 CORRER

1
Error: Se desconectó el servidor.
wwhite@debian2:~/Documents$
```

```
rrios@localhost:~
File Edit View Search Terminal Help
[rrios@localhost ~]$ szclient Anna 192.168.1.200 9999
Bienvenido al Safari Zone!
Apareció un Seel! tienes 10 pokebolas.
Acción:
1 POKEBOLA      2 PIEDRA
3 CARNADA       4 CORRER

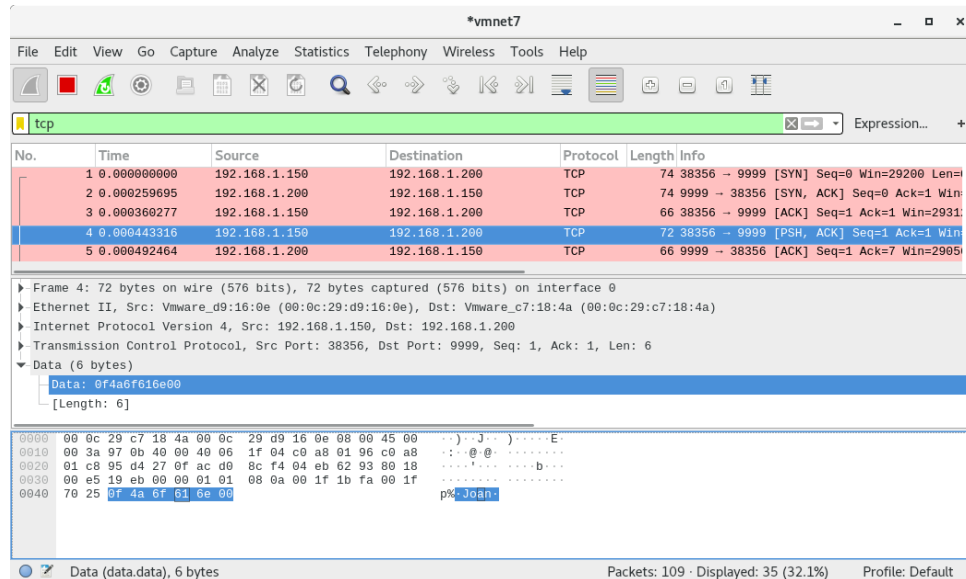
1
Seel está observando cuidadosamente, tienes 9 pokebolas.
Acción:
1 POKEBOLA      2 PIEDRA
3 CARNADA       4 CORRER

4
Error: Se desconectó el servidor.
[rrios@localhost ~]$
```

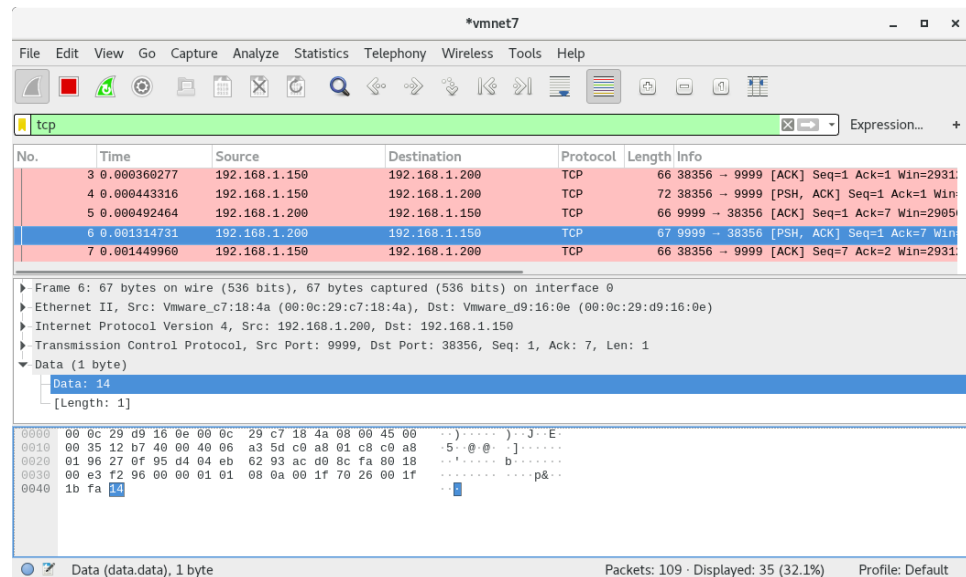
```
tmontana@debian1: ~/Documents
File Edit View Search Terminal Help
tmontana@debian1:~/Documents$ szserver
Creando un servidor con IP 0.0.0.0 en puerto 9999
Atendiendo al cliente 4 con IP 192.168.1.150
Atendiendo al cliente 5 con IP 192.168.1.1
-No hubo respuesta del cliente 4.
-No hubo respuesta del cliente 5.
```



Analizando las transmisiones:

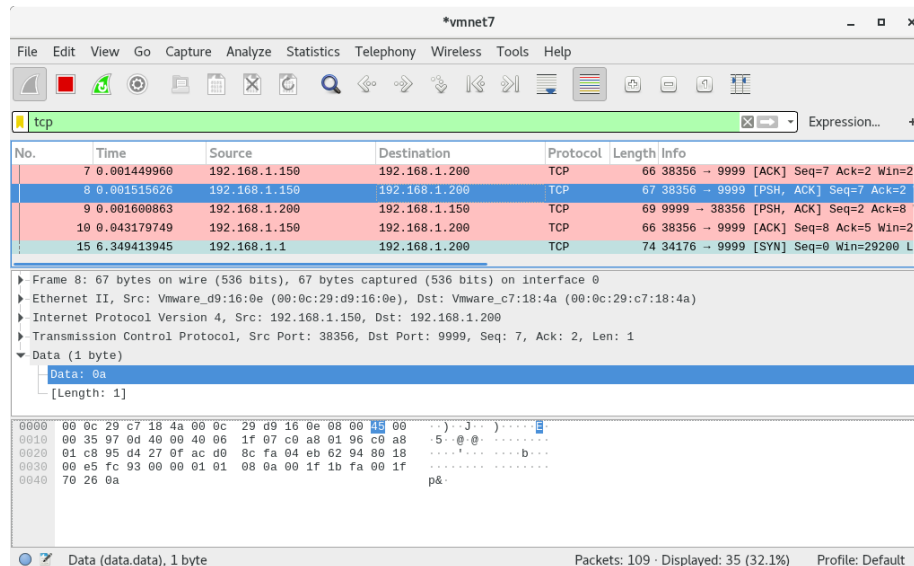


Justo después del 1er 3-way *handshake*, el cliente A manda un mensaje de 6 bytes conformado por el código de *TRY\_USER*(15 -> 0f en hexadecimal) seguido del nombre del usuario, en este caso *Joan*, despues sigue el ACK correspondiente del servidor, este mensaje es donde el cliente pregunta si el usuario existe.



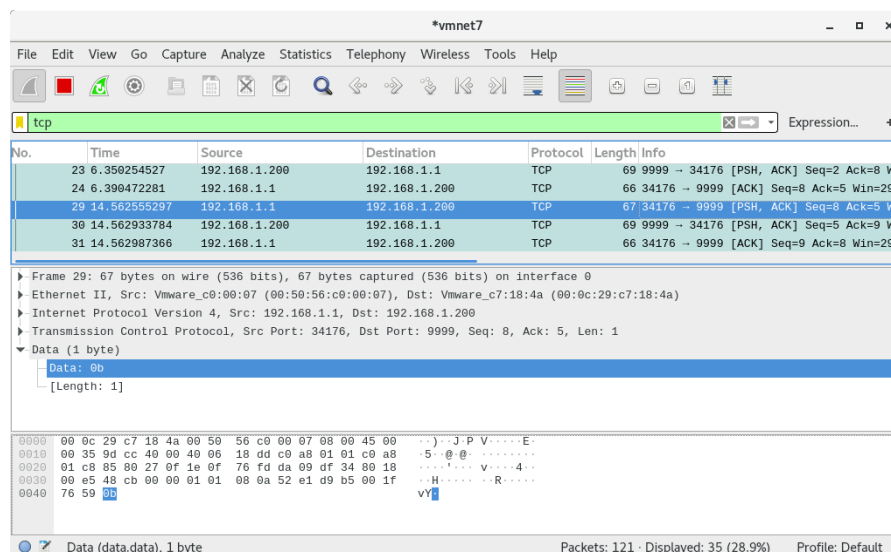
El servidor le responde con un código *USER*(20 -> 14 en hexadecimal) que quiere decir que si existe el usuario, y el usuario manda su ACK.

Después el cliente le pide un Pokémon con un código *ASK\_POK*(10 -> 0a en hexadecimal)



Y el servidor le responde con un código *ENCOUNTER*(21 -> 15 en hexadecimal) seguido del número del Pokémon menos uno (19 -> 13 en hexadecimal), esto por que los nombres de los pokemon estan almacenados en un arreglo (en este caso el número corresponde a *Raticate*), y por ultimo el numero de pokebolas (10 -> 0a en hexadecimal). Justo ahí el cliente B se conecta y el A ya no pide nada al servidor.

La comunicación inicial del cliente B es igual que la anterior, pero también realiza una acción después, tirar una pokebola con el código *BALL*(10 -> 0b en hexadecimal).



El servidor responde con un código *ENCOUNTER*(21 -> 15 en hexadecimal) seguido del número del pokémon que en este caso es un *Seel* (85 -> 55 en hexadecimal) pero ahora con (9 -> 09 hexadecimal) pokebolas. En ese momento el cliente deja de comunicarse.

No.	Time	Source	Destination	Protocol	Length	Info
23	6.350254527	192.168.1.200	192.168.1.1	TCP	69	9999 -> 34176 [PSH, ACK] Seq=2 Ack=8 Win=256 Len=3
24	6.390472281	192.168.1.1	192.168.1.200	TCP	66	34176 -> 9999 [ACK] Seq=8 Ack=5 Win=256 Len=0
29	14.562555297	192.168.1.1	192.168.1.200	TCP	67	34176 -> 9999 [PSH, ACK] Seq=8 Ack=5 Win=256 Len=3
30	14.562933784	192.168.1.200	192.168.1.1	TCP	69	9999 -> 34176 [PSH, ACK] Seq=5 Ack=9 Win=256 Len=3
31	14.562987366	192.168.1.1	192.168.1.200	TCP	66	34176 -> 9999 [ACK] Seq=9 Ack=8 Win=256 Len=0

Frame 30: 69 bytes on wire (552 bits), 69 bytes captured (552 bits) on interface 0  
 Ethernet II, Src: Vmware\_c7:18:4a (00:0c:29:c7:18:4a), Dst: Vmware\_c0:00:07 (00:50:56:c0:00:07)  
 Internet Protocol Version 4, Src: 192.168.1.200, Dst: 192.168.1.1  
 Transmission Control Protocol, Src Port: 9999, Dst Port: 34176, Seq: 5, Ack: 9, Len: 3  
 Data (3 bytes)  
 Data: 155509  
 [Length: 3]

0000 00 50 56 c0 00 07 00 0c 29 c7 18 4a 08 00 45 00 ..PV.....)..J..E-  
 0010 00 37 a9 6c 40 00 40 06 0d 3b c0 a8 01 c8 c0 a8 ..7.l@.@.:[.....  
 0020 01 01 27 0f 85 80 da 09 df 34 1e 0f 76 fe 80 18 .....4..v.....  
 0030 00 e3 2d 70 00 00 01 01 08 0a 00 1f 7e 5e 52 e1 .....p.....AR..  
 0040 d9 b5 15 55 09 .....U

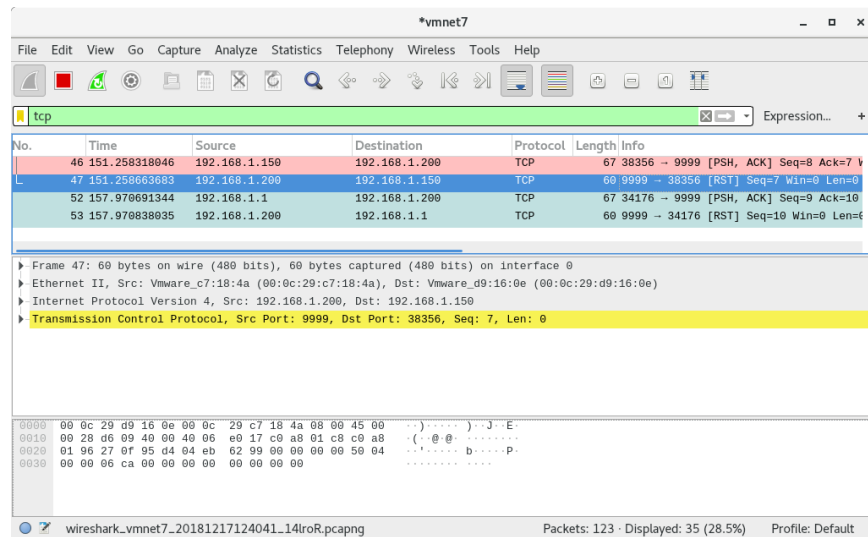
Pasado un tiempo, el servidor cierra la conexión de acuerdo a los *timeouts*, primero con el cliente A y después con el cliente B con un código *TIMEOUT*(40 -> 28 en hexadecimal) seguido por los *FIN* y *ACK* correspondientes.

No.	Time	Source	Destination	Protocol	Length	Info
30	14.562933784	192.168.1.200	192.168.1.1	TCP	69	9999 -> 34176 [PSH, ACK] Seq=5 Ack=9 Win=256 Len=3
31	14.562987366	192.168.1.1	192.168.1.200	TCP	66	34176 -> 9999 [ACK] Seq=9 Ack=8 Win=256 Len=0
32	61.289055449	192.168.1.200	192.168.1.150	TCP	67	9999 -> 38356 [PSH, ACK] Seq=5 Ack=8 Win=256 Len=1
33	61.289573124	192.168.1.150	192.168.1.200	TCP	66	38356 -> 9999 [ACK] Seq=8 Ack=6 Win=256 Len=0
34	62.289669380	192.168.1.200	192.168.1.150	TCP	66	9999 -> 38356 [FIN, ACK] Seq=6 Ack=8 Win=0 Len=0

Frame 32: 67 bytes on wire (536 bits), 67 bytes captured (536 bits) on interface 0  
 Ethernet II, Src: Vmware\_c7:18:4a (00:0c:29:c7:18:4a), Dst: Vmware\_d9:16:0e (00:0c:29:d9:16:0e)  
 Internet Protocol Version 4, Src: 192.168.1.200, Dst: 192.168.1.150  
 Transmission Control Protocol, Src Port: 9999, Dst Port: 38356, Seq: 5, Ack: 8, Len: 1  
 Data (1 byte)  
 Data: 28  
 [Length: 1]

0000 00 0c 29 d9 16 0e 00 0c 29 c7 18 4a 08 00 45 00 .....J..E-  
 0010 00 35 12 b9 40 00 40 06 a3 5b c0 a8 01 c8 c0 a8 ..5..@.@.:[.....  
 0020 01 96 27 0f 95 d4 04 eb 62 97 ac d0 8c fb 80 18 .....b.....  
 0030 00 e3 a2 ac 00 00 01 01 08 0a 00 1f ac 00 00 1f .....  
 0040 1c 05 28 .....[

Los dos clientes se intentan comunicar después del cierre de conexión, pero se les regresa un *RST* (paquete de reseteo) que indica que la conexión ya esta cerrada.



The image shows a Wireshark capture window titled '\*vmnet7'. The filter is 'tcp'. The packet list shows four packets:

No.	Time	Source	Destination	Protocol	Length	Info
46	151.258318946	192.168.1.150	192.168.1.200	TCP	67	38356 → 9999 [PSH, ACK] Seq=8 Ack=7 Win=0 Len=0
47	151.258653883	192.168.1.200	192.168.1.150	TCP	60	9999 → 38356 [RST] Seq=7 Win=0 Len=0
52	157.970691344	192.168.1.1	192.168.1.200	TCP	67	34176 → 9999 [PSH, ACK] Seq=9 Ack=10 Win=0 Len=0
53	157.970838935	192.168.1.200	192.168.1.1	TCP	60	9999 → 34176 [RST] Seq=10 Win=0 Len=0

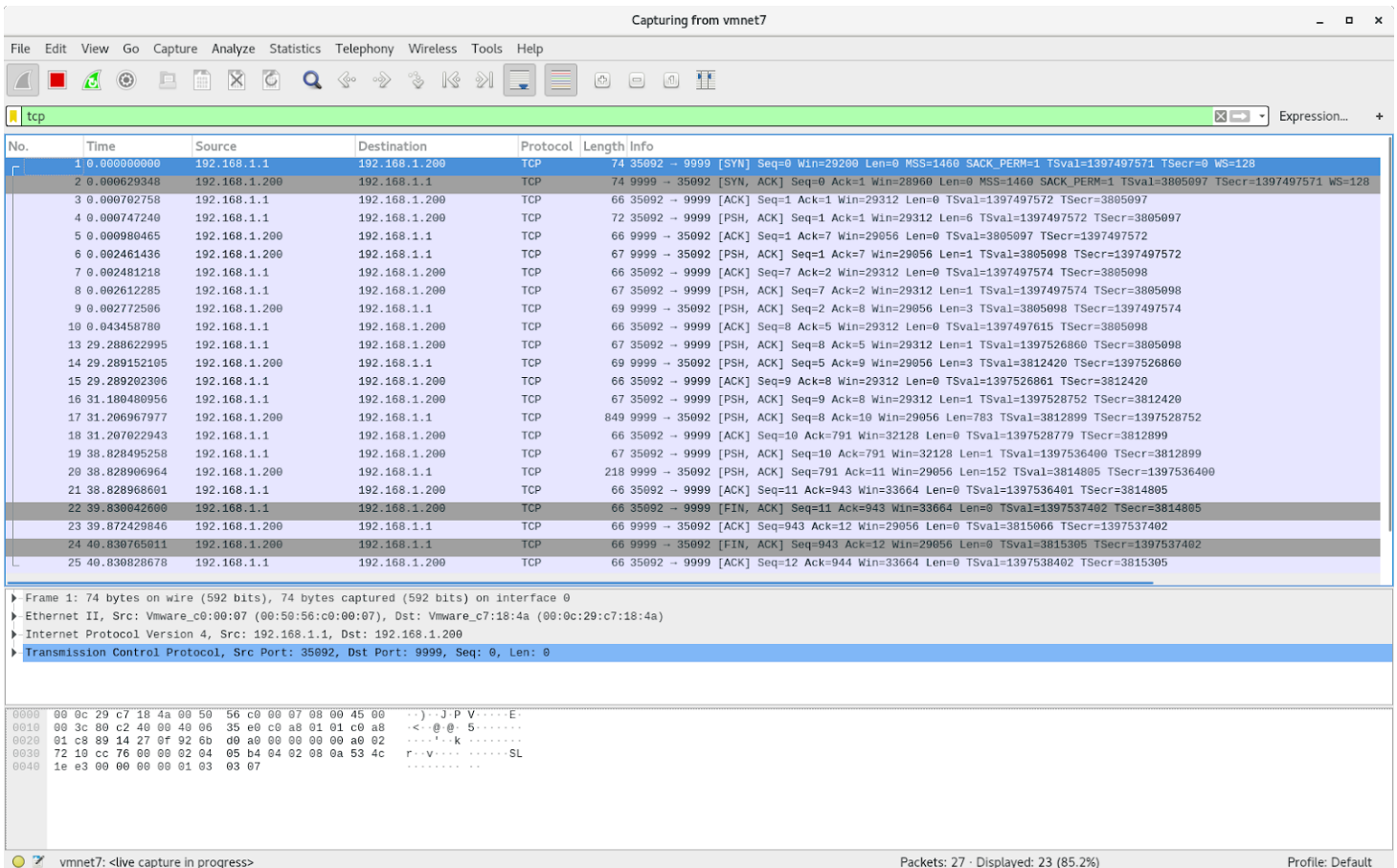
The packet details for packet 47 are expanded, showing:

- Frame 47: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
- Ethernet II, Src: Vmware\_c7:18:4a (00:0c:29:c7:18:4a), Dst: Vmware\_d9:16:0e (00:0c:29:d9:16:0e)
- Internet Protocol Version 4, Src: 192.168.1.200, Dst: 192.168.1.150
- Transmission Control Protocol, Src Port: 9999, Dst Port: 38356, Seq: 7, Len: 0

The packet bytes are shown in hexadecimal and ASCII:

```
0000 00 0c 29 d9 16 0e 00 0c 29 c7 18 4a 08 00 45 00  ..J.PV.....E.
0010 00 28 d6 09 40 00 00 e0 17 c0 a8 01 c8 c0 a8  ..<..@..5.....
0020 01 96 27 0f 95 d4 04 eb 62 99 00 00 00 00 50 04  .....k.....
0030 00 00 06 ca 00 00 00 00 00 00 00 00 00 00 00 00  r.v.....SL
0040
```

Por último esta otra interacción, donde se captura un pokémon y se muestra el pokedex:



The image shows a Wireshark capture window titled 'Capturing from vmnet7'. The filter is 'tcp'. The packet list shows 25 packets, all from 192.168.1.1 to 192.168.1.200. The first packet is a SYN packet, and the subsequent packets are ACKs. The packet details for packet 1 are expanded, showing:

- Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
- Ethernet II, Src: Vmware\_c0:00:07 (00:50:56:c0:00:07), Dst: Vmware\_c7:18:4a (00:0c:29:c7:18:4a)
- Internet Protocol Version 4, Src: 192.168.1.1, Dst: 192.168.1.200
- Transmission Control Protocol, Src Port: 35092, Dst Port: 9999, Seq: 0, Len: 0

The packet bytes are shown in hexadecimal and ASCII:

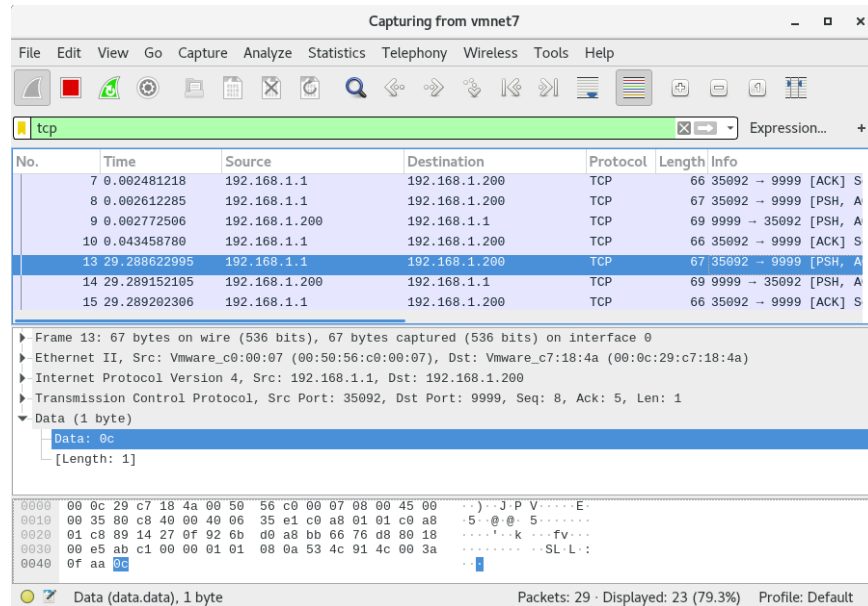
```
0000 00 0c 29 c7 18 4a 00 50 56 c0 00 07 08 00 45 00  ..J.PV.....E.
0010 00 3c 80 c2 40 00 40 06 35 e0 c0 a8 01 01 c0 a8  ..<..@..5.....
0020 01 c8 89 14 27 0f 92 60 d0 a0 00 00 00 00 a0 02  .....k.....
0030 72 10 cc 76 00 00 02 04 05 b4 04 02 08 0a 53 4c  r.v.....SL
0040 1e e3 00 00 00 00 01 03 03 07
```

Las capturas de esta interacción

```
rrios@localhost:~  
File Edit View Search Terminal Help  
[rrios@localhost ~]$ szclient Anna 192.168.1.200 9999  
Bienvenido al Safari Zone!  
Apareció un Gloom! tienes 10 pokebolas.  
Acción:  
1 POKEBOLA      2 PIEDRA  
3 CARNADA       4 CORRER  
  
2  
Gloom está observando cuidadosamente, tienes 10 pokebolas.  
Acción:  
1 POKEBOLA      2 PIEDRA  
3 CARNADA       4 CORRER  
  
1  
Lograste capturar al Pokemon! deseas ver tu Pokedex? [y/n]  
y  
1      Bulbasaur o  
2      Ivysaur o  
3      Venusaur o  
4      Charmander o  
5      Charmeleon o  
6      Charizard o  
7      Squirtle o  
8      Wartortle o  
9      Blastoise o  
10     Caterpie o  
11     Metapod o  
12     Butterfree o  
13     Weedle o  
14     Kakuna o  
15     Beedrill o  
16     Pidgey o  
17     Pidgeotto o  
18     Pidgeot o
```

```
tmontana@debian1: ~/Documents  
File Edit View Search Terminal Help  
tmontana@debian1:~/Documents$ szserver  
Creando un servidor con IP 0.0.0.0 en puerto 9999  
Atendiendo al cliente 4 con IP 192.168.1.1  
Desconectandose del cliente 4.  
█
```

Después de verificar al usuario, y pedir el pokemon correspondiente (*Gloom* **44 -1**) el cliente lanza una piedra con el código *ROCK*(**12** -> **0c** en hexadecimal), como no se escapa el servidor responde con el código *ENCOUNTER* correspondiente sin perder pokebolas.



Capturing from vmnet7

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp

No.	Time	Source	Destination	Protocol	Length	Info
7	0.002481218	192.168.1.1	192.168.1.200	TCP	66	35092 → 9999 [ACK] S
8	0.002612285	192.168.1.1	192.168.1.200	TCP	67	35092 → 9999 [PSH, A
9	0.002772506	192.168.1.200	192.168.1.1	TCP	69	9999 → 35092 [PSH, A
10	0.043458780	192.168.1.1	192.168.1.200	TCP	66	35092 → 9999 [ACK] S
13	29.288622995	192.168.1.1	192.168.1.200	TCP	67	35092 → 9999 [PSH, A
14	29.289152105	192.168.1.200	192.168.1.1	TCP	69	9999 → 35092 [PSH, A
15	29.289202306	192.168.1.1	192.168.1.200	TCP	66	35092 → 9999 [ACK] S

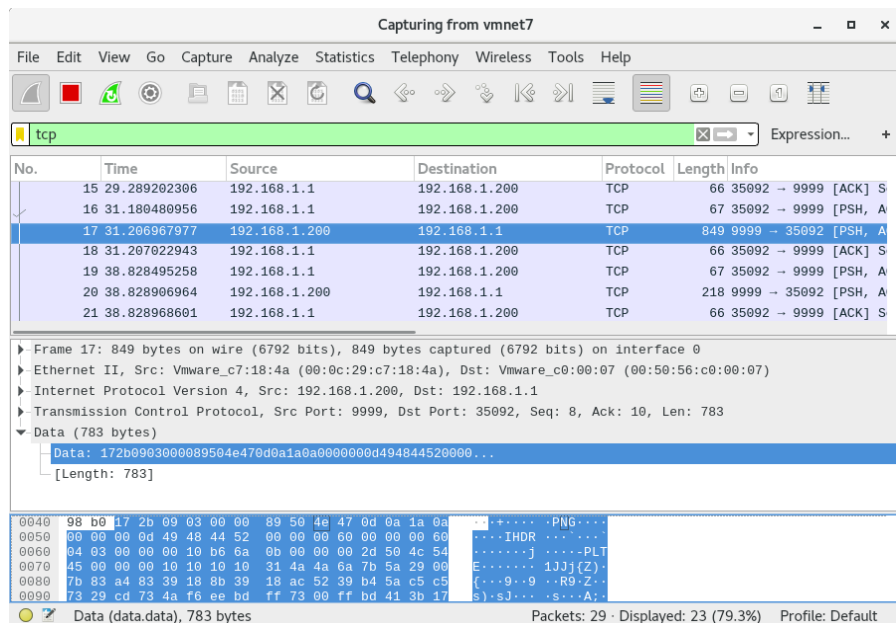
Frame 13: 67 bytes on wire (536 bits), 67 bytes captured (536 bits) on interface 0  
Ethernet II, Src: Vmware\_c0:00:07 (00:50:56:c0:00:07), Dst: Vmware\_c7:18:4a (00:0c:29:c7:18:4a)  
Internet Protocol Version 4, Src: 192.168.1.1, Dst: 192.168.1.200  
Transmission Control Protocol, Src Port: 35092, Dst Port: 9999, Seq: 8, Ack: 5, Len: 1  
Data (1 byte)  
Data: 0c  
[Length: 1]

0000 00 0c 29 c7 18 4a 00 50 56 c0 00 07 08 00 45 00 ... J P V ... E  
0010 00 35 80 c8 40 00 00 06 35 e1 c0 a8 01 01 c0 a8 ... 5 @ @ 5 ...  
0020 01 c8 89 14 27 0f 92 6b d0 a8 bb 66 76 d8 80 18 ... k ... f v ...  
0030 00 e5 ab c1 00 00 01 01 08 0a 53 4c 91 4c 00 3a ... SL L ...  
0040 0f aa 0c ...

Data (data.data), 1 byte

Packets: 29 · Displayed: 23 (79.3%) Profile: Default

Después el cliente lanza una pokebola, y en esta ocasión atrapa al pokemon, por lo que el servidor responde con un código *SUCCESS*(**23** -> **17** en hexadecimal) seguido de la imagen del pokemon correspondiente:



Capturing from vmnet7

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp

No.	Time	Source	Destination	Protocol	Length	Info
15	29.289202306	192.168.1.1	192.168.1.200	TCP	66	35092 → 9999 [ACK] S
16	31.180480956	192.168.1.1	192.168.1.200	TCP	67	35092 → 9999 [PSH, A
17	31.206967977	192.168.1.200	192.168.1.1	TCP	849	9999 → 35092 [PSH, A
18	31.207022943	192.168.1.1	192.168.1.200	TCP	66	35092 → 9999 [ACK] S
19	38.828495258	192.168.1.1	192.168.1.200	TCP	67	35092 → 9999 [PSH, A
20	38.828906964	192.168.1.200	192.168.1.1	TCP	218	9999 → 35092 [PSH, A
21	38.828968601	192.168.1.1	192.168.1.200	TCP	66	35092 → 9999 [ACK] S

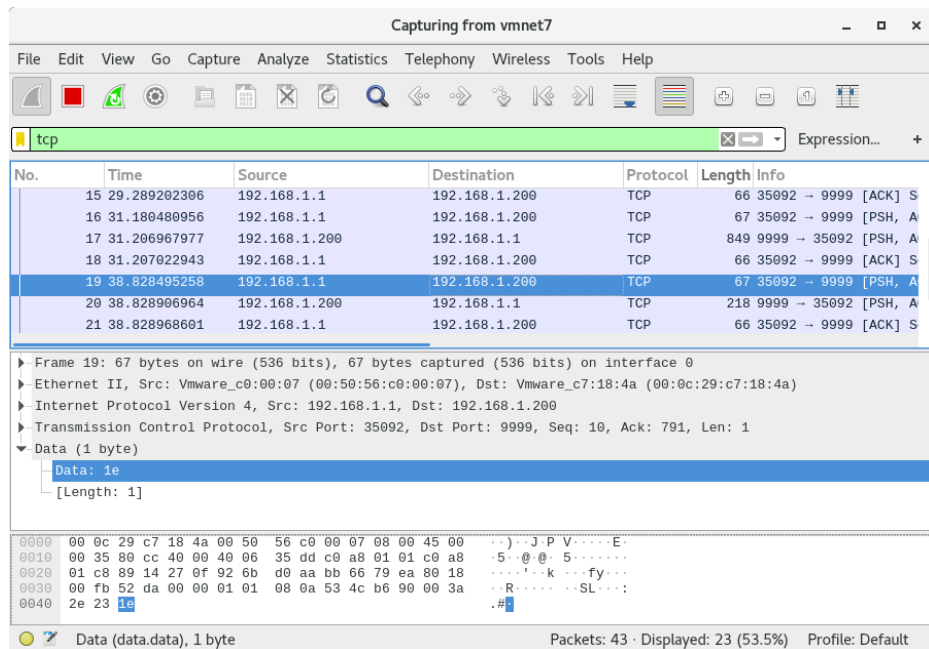
Frame 17: 849 bytes on wire (6792 bits), 849 bytes captured (6792 bits) on interface 0  
Ethernet II, Src: Vmware\_c7:18:4a (00:0c:29:c7:18:4a), Dst: Vmware\_c0:00:07 (00:50:56:c0:00:07)  
Internet Protocol Version 4, Src: 192.168.1.200, Dst: 192.168.1.1  
Transmission Control Protocol, Src Port: 9999, Dst Port: 35092, Seq: 8, Ack: 10, Len: 783  
Data (783 bytes)  
Data: 172b0903000089504e470d0a1a0a0000000d494844520000...  
[Length: 783]

0040 98 b6 17 2b 09 03 00 00 89 50 4e 47 0d 0a 1a 0a ... + ... PNG ...  
0050 00 00 00 0d 49 48 44 52 00 00 00 60 00 00 00 60 ... IHDR ...  
0060 04 03 00 00 00 10 b6 6a 0b 00 00 00 2d 50 4c 54 ... j ... PLT ...  
0070 45 00 00 00 10 10 10 10 31 4a 4a 6a 7b 5a 29 00 ... E ... 1JJ (Z) ...  
0080 7b 83 a4 83 39 18 8b 39 18 ac 52 39 b4 5a c5 c5 ... ( ... 9 ... R9 Z ...  
0090 73 29 cd 73 4a f6 ee bd ff 73 00 ff bd 41 3b 17 ... s) sJ ... s ... A ...

Data (data.data), 783 bytes

Packets: 29 · Displayed: 23 (79.3%) Profile: Default

Lo siguiente que hace el cliente, después de recibir la imagen es mandar su respuesta acerca de si mostrar la pokédex o no, en este caso se usa el código OK(30 -> 1e en hexadecimal) para indicar que si se quiere obtener:



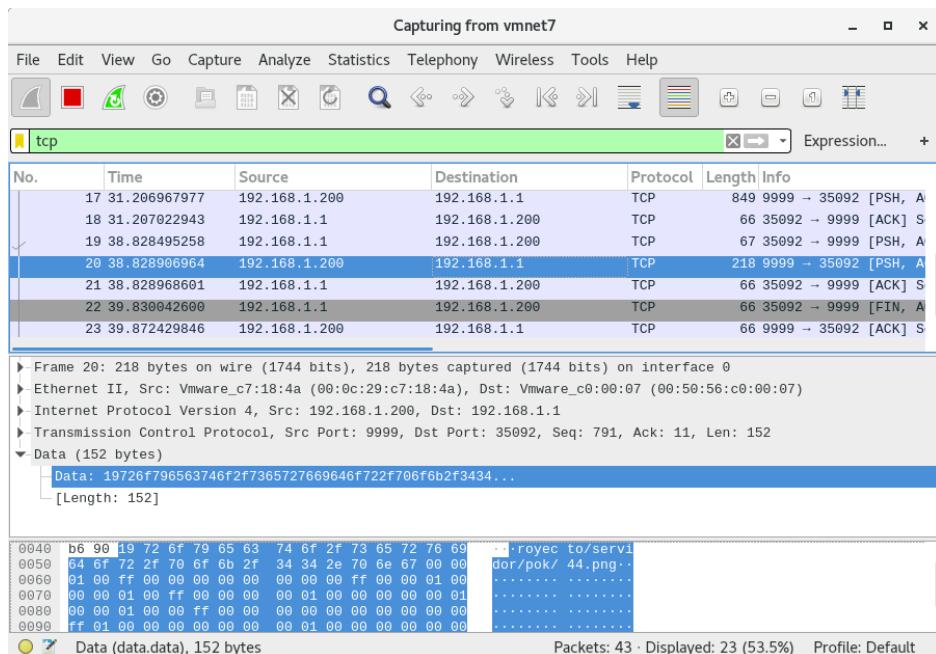
No.	Time	Source	Destination	Protocol	Length	Info
15	29.289202306	192.168.1.1	192.168.1.200	TCP	66	35092 → 9999 [ACK] S
16	31.180480956	192.168.1.1	192.168.1.200	TCP	67	35092 → 9999 [PSH, A
17	31.206967977	192.168.1.200	192.168.1.1	TCP	849	9999 → 35092 [PSH, A
18	31.207022943	192.168.1.1	192.168.1.200	TCP	66	35092 → 9999 [ACK] S
19	38.828495258	192.168.1.1	192.168.1.200	TCP	67	35092 → 9999 [PSH, A
20	38.828906964	192.168.1.200	192.168.1.1	TCP	218	9999 → 35092 [PSH, A
21	38.828968601	192.168.1.1	192.168.1.200	TCP	66	35092 → 9999 [ACK] S

Frame 19: 67 bytes on wire (536 bits), 67 bytes captured (536 bits) on interface 0  
Ethernet II, Src: Vmware\_c0:00:07 (00:50:56:c0:00:07), Dst: Vmware\_c7:18:4a (00:0c:29:c7:18:4a)  
Internet Protocol Version 4, Src: 192.168.1.1, Dst: 192.168.1.200  
Transmission Control Protocol, Src Port: 35092, Dst Port: 9999, Seq: 10, Ack: 791, Len: 1  
Data (1 byte)  
Data: 1e  
[Length: 1]

0000 00 0c 29 c7 18 4a 00 50 56 c0 00 07 08 00 45 00 ... J P V ... E  
0010 00 35 00 cc 40 00 06 35 dd c0 a8 01 01 c0 a8 ... 5 @ @ 5 ...  
0020 01 c8 89 14 27 0f 92 6b d0 aa bb 66 79 ea 80 18 ... k ... fy ...  
0030 00 fb 52 da 00 00 01 01 08 0a 53 4c b6 90 00 3a ... R ... SL ...  
0040 2e 23 1e ... #

Data (data.data), 1 byte      Packets: 43 · Displayed: 23 (53.5%)      Profile: Default

El servidor responde con el código *POKEDEX*(25 -> 19 en hexadecimal) seguido de los contenidos del archivo correspondiente al usuario y se cierra la conexión de ambos lados.



No.	Time	Source	Destination	Protocol	Length	Info
17	31.206967977	192.168.1.200	192.168.1.1	TCP	849	9999 → 35092 [PSH, A
18	31.207022943	192.168.1.1	192.168.1.200	TCP	66	35092 → 9999 [ACK] S
19	38.828495258	192.168.1.1	192.168.1.200	TCP	67	35092 → 9999 [PSH, A
20	38.828906964	192.168.1.200	192.168.1.1	TCP	218	9999 → 35092 [PSH, A
21	38.828968601	192.168.1.1	192.168.1.200	TCP	66	35092 → 9999 [ACK] S
22	39.830042600	192.168.1.1	192.168.1.200	TCP	66	35092 → 9999 [FIN, A
23	39.872429846	192.168.1.200	192.168.1.1	TCP	66	9999 → 35092 [ACK] S

Frame 20: 218 bytes on wire (1744 bits), 218 bytes captured (1744 bits) on interface 0  
Ethernet II, Src: Vmware\_c7:18:4a (00:0c:29:c7:18:4a), Dst: Vmware\_c0:00:07 (00:50:56:c0:00:07)  
Internet Protocol Version 4, Src: 192.168.1.200, Dst: 192.168.1.1  
Transmission Control Protocol, Src Port: 9999, Dst Port: 35092, Seq: 791, Ack: 11, Len: 152  
Data (152 bytes)  
Data: 19726f796563746f2f7365727669646f722f706f6b2f3434...  
[Length: 152]

0040 b6 90 19 72 6f 79 65 63 74 6f 2f 73 65 72 76 69 ... royec to/servi  
0050 54 6f 72 2f 70 6f 6b 2f 34 34 2e 70 6e 67 00 00 ... dor/pok/ 44.png  
0060 31 00 ff 00 00 00 00 00 00 00 ff 00 00 01 00 ...  
0070 00 00 01 00 ff 00 00 00 00 01 00 00 00 00 00 ...  
0080 00 00 01 00 ff 00 00 00 00 00 00 00 00 00 00 ...  
0090 ff 01 00 00 00 00 00 00 01 00 00 00 00 00 00 ...

Data (data.data), 152 bytes      Packets: 43 · Displayed: 23 (53.5%)      Profile: Default

---

## Repositorio

Repositorio: <https://github.com/M0rn1ngSt4r/Proyecto-Final---Redes>