

**Entrega - Capa de transporte - UDP**

**Práctica/Laboratorio de UDP**

Integrantes:

Gauto, Marianela

Rucci, Milagros

Sosa, Ester

## 1. Sobre el protocolo UDP

### a- ¿Cómo se denomina la PDU?

La PDU (Protocol Data Unit) correspondiente al protocolo UDP, se denomina datagrama.

### b- ¿Qué servicios o aplicaciones suelen utilizar este protocolo?

Normalmente la telefonía por Internet se ejecuta sobre UDP. Cada lado de una aplicación de telefonía por Internet necesita enviar datos a través de la red a una cierta velocidad mínima; esto será posible más probablemente con UDP que con TCP. Además, las aplicaciones de telefonía por Internet son tolerantes a las pérdidas de datos, por lo que no necesitan el servicio de transferencia de datos fiable proporcionado por TCP. Los protocolos DNS y TFTP (protocolo utilizado para la transferencia de archivos) se ejecuta sobre UDP.

### c- ¿Qué hace el protocolo UDP en relación al control de errores?

El protocolo UDP posee un campo llamado checksum (16 bits). Es una suma de verificación utilizada para comprobar si hay errores tanto en la cabecera como en los datos.

## 2. Interactuando con un servidor UDP utilizando hping3

### a- Conéctese al servidor redes.catedras.linti.unlp.edu.ar al port UDP/1235. ¿Qué respuesta obtuvo? ¿Qué puede inferir de dicho puerto?

Comando: `sudo hping3 -2 -p 1235 redes.catedras.linti.unlp.edu.ar`

```
redes@redes:~$ sudo hping3 -2 -p 1235 redes.catedras.linti.unlp.edu.ar
HPING redes.catedras.linti.unlp.edu.ar (eth0 163.10.20.136): udp mode set, 28 headers + 0 data bytes
ICMP Port Unreachable from ip=163.10.20.136 name=redes.catedras.linti.unlp.edu.ar
status=0 port=1788 seq=0
ICMP Port Unreachable from ip=163.10.20.136 name=redes.catedras.linti.unlp.edu.ar
status=0 port=1790 seq=2
ICMP Port Unreachable from ip=163.10.20.136 name=redes.catedras.linti.unlp.edu.ar
status=0 port=1792 seq=4
ICMP Port Unreachable from ip=163.10.20.136 name=redes.catedras.linti.unlp.edu.ar
status=0 port=1793 seq=5
ICMP Port Unreachable from ip=163.10.20.136 name=redes.catedras.linti.unlp.edu.ar
status=0 port=1794 seq=6
ICMP Port Unreachable from ip=163.10.20.136 name=redes.catedras.linti.unlp.edu.ar
status=0 port=1796 seq=8
ICMP Port Unreachable from ip=163.10.20.136 name=redes.catedras.linti.unlp.edu.ar
status=0 port=1797 seq=9
ICMP Port Unreachable from ip=163.10.20.136 name=redes.catedras.linti.unlp.edu.ar
status=0 port=1798 seq=10
^C
--- redes.catedras.linti.unlp.edu.ar hping statistic ---
11 packets transmitted, 8 packets received, 28% packet loss
round-trip min/avg/max = 15.9/229.7/1149.8 ms
```

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	163.10.20.136	UDP	42	Source port: 1777 Destination port: 1235
2	0.110488000	163.10.20.136	10.0.2.15	ICMP	70	Destination unreachable (Port unreachable)
3	0.113909000	10.0.2.15	163.10.0.67	DNS	97	Standard query 0x7841 PTR 136.20.10.163.in-addr.arpa
4	0.410458000	163.10.0.67	10.0.2.15	DNS	538	Standard query response 0x7841
5	0.412143000	10.0.2.15	163.10.10.61	DNS	97	Standard query 0x93ba PTR 136.20.10.163.in-addr.arpa
6	1.000771000	10.0.2.15	163.10.20.136	UDP	42	Source port: 1778 Destination port: 1235
7	1.212187000	10.0.2.15	163.10.0.65	DNS	97	Standard query 0xdba4 PTR 136.20.10.163.in-addr.arpa
8	1.419487000	163.10.0.65	10.0.2.15	DNS	176	Standard query response 0xdba4 PTR redes.catedras.linti.unlp.edu.ar
9	2.002863000	10.0.2.15	163.10.20.136	UDP	42	Source port: 1779 Destination port: 1235
10	2.019164000	163.10.20.136	10.0.2.15	ICMP	70	Destination unreachable (Port unreachable)
11	3.004716000	10.0.2.15	163.10.20.136	UDP	42	Source port: 1780 Destination port: 1235
12	3.022569000	163.10.20.136	10.0.2.15	ICMP	70	Destination unreachable (Port unreachable)
13	4.005668000	10.0.2.15	163.10.20.136	UDP	42	Source port: 1781 Destination port: 1235
14	4.020856000	163.10.20.136	10.0.2.15	ICMP	70	Destination unreachable (Port unreachable)
15	5.001515000	CadmusCo_fd:3c:8e	RealtekU_12:35:02	ARP	42	Who has 10.0.2.2? Tell 10.0.2.15
16	5.002750000	RealtekU_12:35:02	CadmusCo_fd:3c:8e	ARP	60	10.0.2.2 is at 52:54:00:12:35:02

La respuesta que obtuvimos es ICMP Port Unreachable. Podemos inferir que no podemos conectarnos con redes.catedras.linti.unlp.edu.ar en el puerto 1235.

**b- Conéctese al servidor redes.catedras.linti.unlp.edu.ar al port UDP/1234 utilizando el puerto 5000 como puerto origen. ¿Qué respuesta obtuvo? ¿Qué puede inferir de dicho puerto? Deberá realizar y entregar una captura de tráfico para cada uno de los ítems anteriores.**

```
redes@redes:~$ sudo hping3 -2 -us 5000 -p 1234 redes.catedras.linti.unlp.edu.ar
HPING redes.catedras.linti.unlp.edu.ar (eth0 163.10.20.136): udp mode set, 28 headers + 0 data bytes
^C
--- redes.catedras.linti.unlp.edu.ar hping statistic ---
8 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	163.10.0.67	DNS	103	Standard query 0x337d A redes.catedras.linti.unlp.edu.ar
2	0.026277000	163.10.0.67	10.0.2.15	DNS	386	Standard query response 0x337d
3	0.027514000	10.0.2.15	163.10.5.66	DNS	103	Standard query 0x3e86 A redes.catedras.linti.unlp.edu.ar
4	0.043107000	163.10.5.66	10.0.2.15	DNS	212	Standard query response 0x3e86 A 163.10.20.136
5	0.045309000	10.0.2.15	163.10.20.136	UDP	42	Source port: 5000 Destination port: 1234
6	1.045780000	10.0.2.15	163.10.20.136	UDP	42	Source port: 5001 Destination port: 1234[Malformed Packet]
7	2.048052000	10.0.2.15	163.10.20.136	UDP	42	Source port: 5002 Destination port: 1234
8	3.049015000	10.0.2.15	163.10.20.136	UDP	42	Source port: 5003 Destination port: 1234
9	4.052695000	10.0.2.15	163.10.20.136	UDP	42	Source port: 5004 Destination port: 1234
10	5.006192000	CadmusCo_fd:3c:8e	RealtekU_12:35:02	ARP	42	Who has 10.0.2.2? Tell 10.0.2.15
11	5.006920000	RealtekU_12:35:02	CadmusCo_fd:3c:8e	ARP	60	10.0.2.2 is at 52:54:00:12:35:02
12	5.053464000	10.0.2.15	163.10.20.136	UDP	42	Source port: 5005 Destination port: 1234
13	6.054120000	10.0.2.15	163.10.20.136	UDP	42	Source port: 5006 Destination port: 1234
14	7.055729000	10.0.2.15	163.10.20.136	UDP	42	Source port: 5007 Destination port: 1234
15	8.056455000	10.0.2.15	163.10.20.136	UDP	42	Source port: 5008 Destination port: 1234
16	9.057013000	10.0.2.15	163.10.20.136	UDP	42	Source port: 5009 Destination port: 1234
17	10.057933000	10.0.2.15	163.10.20.136	UDP	42	Source port: 5010 Destination port: 1234
18	11.058511000	10.0.2.15	163.10.20.136	UDP	42	Source port: 5011 Destination port: 1234

Se transmitieron todos los paquetes y no se recibió nada (ya que UDP no tiene mensajes de retorno por reconocimiento).

### **3. Utilizando el lenguaje de programación que desee**

**a- Implemente un servidor UDP que dado un string recibido por el cliente devuelva dicho string en uppercase.**

```
# -*- coding: utf-8 -*-
import socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server_socket.bind(("localhost", 12000))
while True:
    message, address = server_socket.recvfrom(1024)
    upperText = message.upper()
    print address
    print upperText
    server_socket.sendto(upperText, address)
server_socket.close()
```

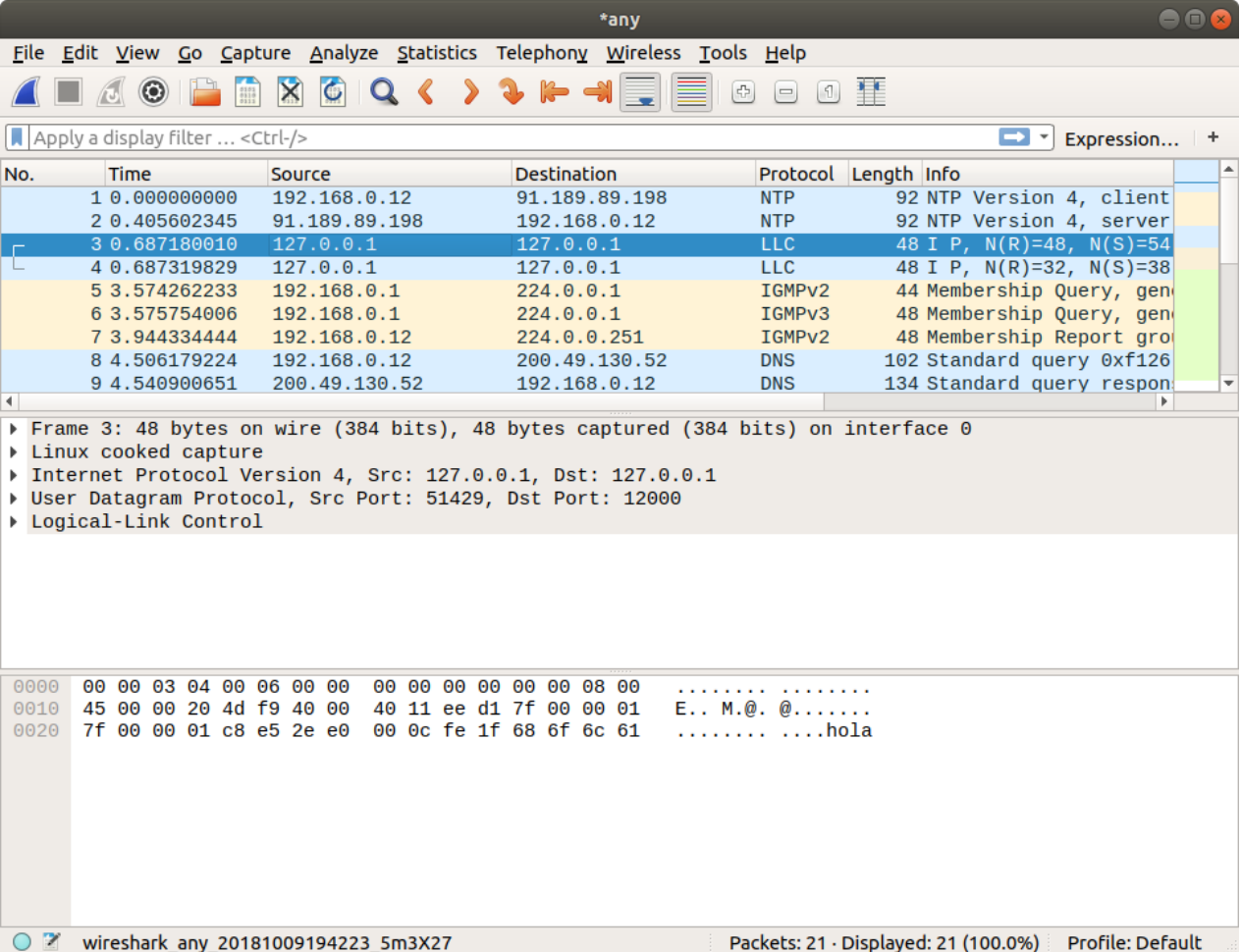
**b- Implemente un cliente UDP que utilice dicho servidor.**

```
# -*- coding: utf-8 -*-
import socket
import sys

address = ("localhost", 12000)
client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

while (1):
    message_client = raw_input("Ingrese el mensaje: ")
    client_socket.sendto(message_client, address)
    message_server, address = client_socket.recvfrom(1024)
    print "Respuesta del servidor: ", message_server
```

c- Establezca una conexión entre el cliente y el servidor y capture tráfico con Wireshark. Analice los campos del datagrama UDP.



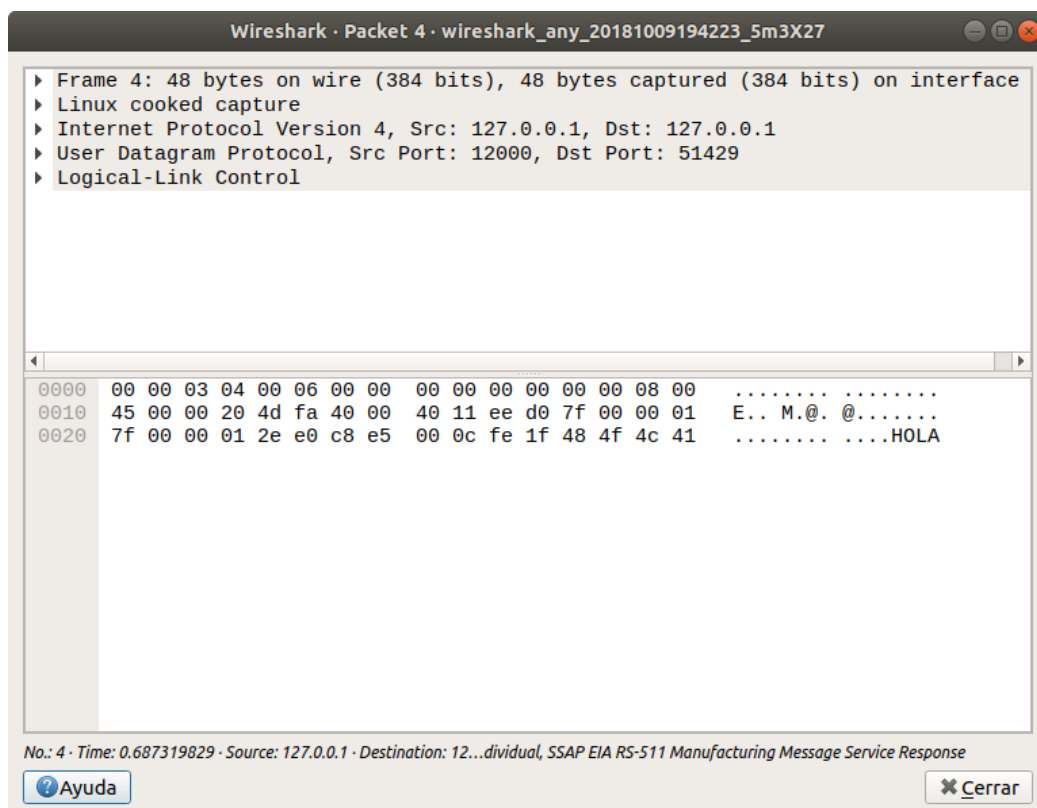
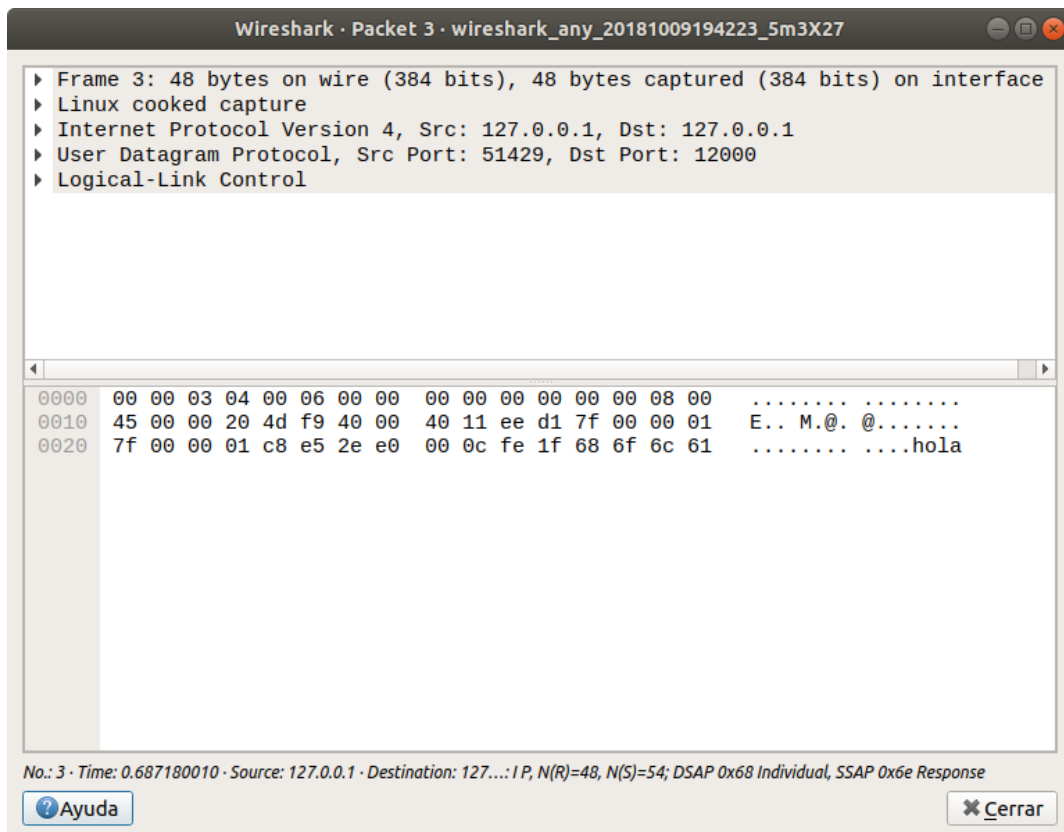
Wireshark interface showing a packet capture of an NTP client-server exchange. The packet list shows packets 1-9. Packet 3 is selected, showing details for Ethernet II, Internet Protocol Version 4, User Datagram Protocol, and Logical-Link Control. The packet bytes pane shows the raw data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.0.12	91.189.89.198	NTP	92	NTP Version 4, client
2	0.405602345	91.189.89.198	192.168.0.12	NTP	92	NTP Version 4, server
3	0.687180010	127.0.0.1	127.0.0.1	LLC	48	I P, N(R)=48, N(S)=54
4	0.687319829	127.0.0.1	127.0.0.1	LLC	48	I P, N(R)=32, N(S)=38
5	3.574262233	192.168.0.1	224.0.0.1	IGMPv2	44	Membership Query, gen
6	3.575754006	192.168.0.1	224.0.0.1	IGMPv3	48	Membership Query, gen
7	3.944334444	192.168.0.12	224.0.0.251	IGMPv2	48	Membership Report gro
8	4.506179224	192.168.0.12	200.49.130.52	DNS	102	Standard query 0xf126
9	4.540900651	200.49.130.52	192.168.0.12	DNS	134	Standard query respon

Frame 3: 48 bytes on wire (384 bits), 48 bytes captured (384 bits) on interface 0  
Linux cooked capture  
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1  
User Datagram Protocol, Src Port: 51429, Dst Port: 12000  
Logical-Link Control

```
0000 00 00 03 04 00 06 00 00 00 00 00 00 00 08 00 .....  
0010 45 00 00 20 4d f9 40 00 40 11 ee d1 7f 00 00 01 E.. M.@. @.....  
0020 7f 00 00 01 c8 e5 2e e0 00 0c fe 1f 68 6f 6c 61 .....hola
```

wireshark\_any\_20181009194223\_5m3X27      Packets: 21 · Displayed: 21 (100.0%)      Profile: Default



En las capturas se puede visualizar cuando el cliente envía el datagrama desde localhost puerto 51429 hacia localhost puerto 12000 (definido por el servidor cuando configura la función bind), el mensaje “hola” y recibe la respuesta del servidor del texto en mayúsculas.

d- Utilizando el cliente recientemente desarrollado, conéctese al servidor `redes.catedras.linti.unlp.edu.ar` al puerto UDP 1233 utilizando como puerto origen el `.port 5000`. Envíe como dato el nombre del grupo. Por ejemplo, si su grupo es el grupo A, debería enviar como dato “Grupo A” sin las comillas. ¿Qué mensaje obtuvo?

```
# -*- coding: utf-8 -*-
```

```
import socket
```

```
import sys
```

```
port = 1233
```

```
address = "redes.catedras.linti.unlp.edu.ar"
```

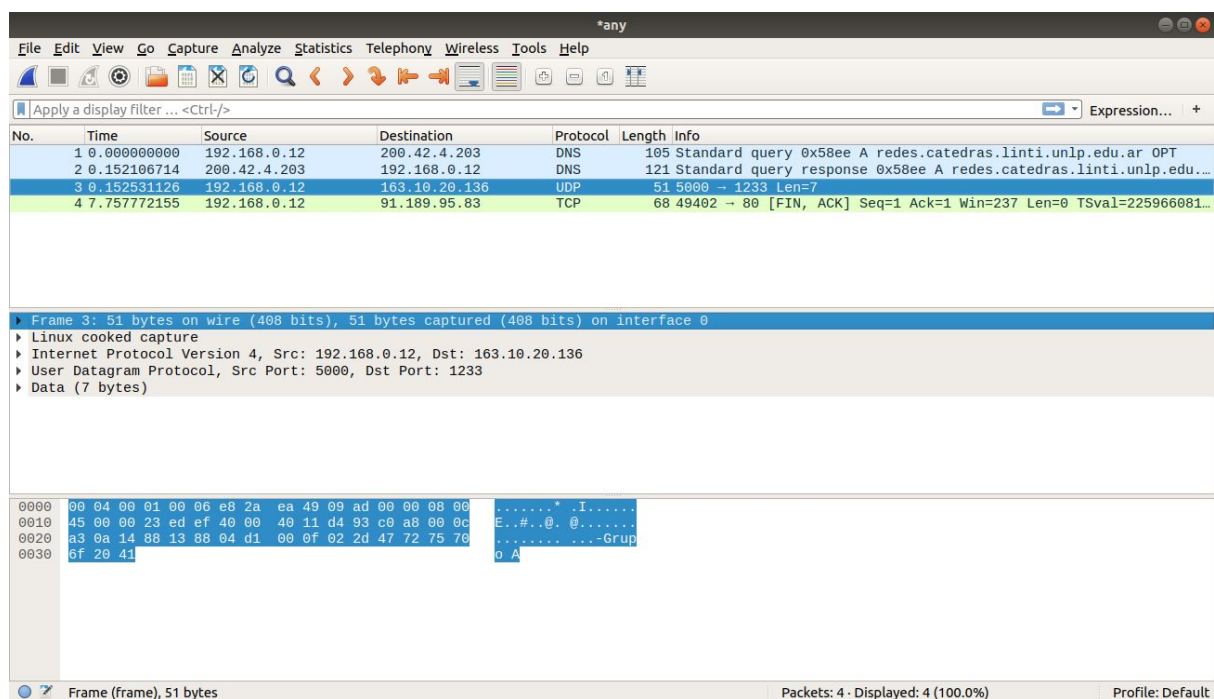
```
client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

```
client_socket.bind(("", 5000))
```

```
client_socket.sendto('Grupo A', (address, port))
```

```
message_server, address = client_socket.recvfrom(4096)
```

```
print "Respuesta del servidor: ", message_server
```



Podemos observar que el paquete se envía por el puerto origen 5000, a la ip 163.18.28.136 que es de `redes.catedras.linti.unlp.edu.ar` puerto 1233, pero no recibimos respuesta del servidor y como el protocolo



En la siguiente captura podemos observar con el comando netstat -nul (network udp listening) que el puerto 5000 de UDP está en estado escuchando y acepta conexión de cualquier IP.

```

echu@sosa: ~
Archivo Editar Ver Buscar Terminal Ayuda
echu@sosa:~$ netstat -nul
Conexiones activas de Internet (solo servidores)
Proto Recib Enviad Dirección local Dirección remota Estado
udp 4608 0 127.0.0.53:53 0.0.0.0:*
udp 10752 0 0.0.0.0:68 0.0.0.0:*
udp 0 0 0.0.0.0:37112 0.0.0.0:*
udp 0 0 0.0.0.0:631 0.0.0.0:*
udp 0 0 0.0.0.0:5000 0.0.0.0:*
udp 46592 0 0.0.0.0:5353 0.0.0.0:*
udp6 0 0 :::37131 :::*
udp6 18432 0 :::5353 :::*

```

Con el siguiente comando podemos observar también que el servidor está escuchando en el puerto 1233.

```

echu@sosa: ~
Archivo Editar Ver Buscar Terminal Ayuda
echu@sosa:~$ sudo nmap -sU -p 1233 redes.catedras.linti.unledu.ar -Pn
[sudo] contraseña para echu:

Starting Nmap 7.60 ( https://nmap.org ) at 2018-10-09 21:05 -03
Failed to resolve "redes.catedras.linti.unledu.ar".
WARNING: No targets were specified, so 0 hosts scanned.
Nmap done: 0 IP addresses (0 hosts up) scanned in 0.60 seconds
echu@sosa:~$

```

Se consulta a la cátedra si había algún problema con el servidor, nos comentan que probemos nuevamente, probamos y ahora si obtenemos respuesta.

The image shows a Wireshark packet capture interface. The top menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. Below the menu is a toolbar with various icons for packet capture and analysis. The main pane displays a list of captured packets with columns for No., Time, Source, Destination, Protocol, Length, and Info. Packet 11 is highlighted, showing a UDP packet from 192.168.0.12 to 163.10.20.136 on port 5000 to 1233. The packet details pane shows the Ethernet II header, Internet Protocol Version 4 header, and User Datagram Protocol header. The packet bytes pane shows the raw data in hexadecimal and ASCII.

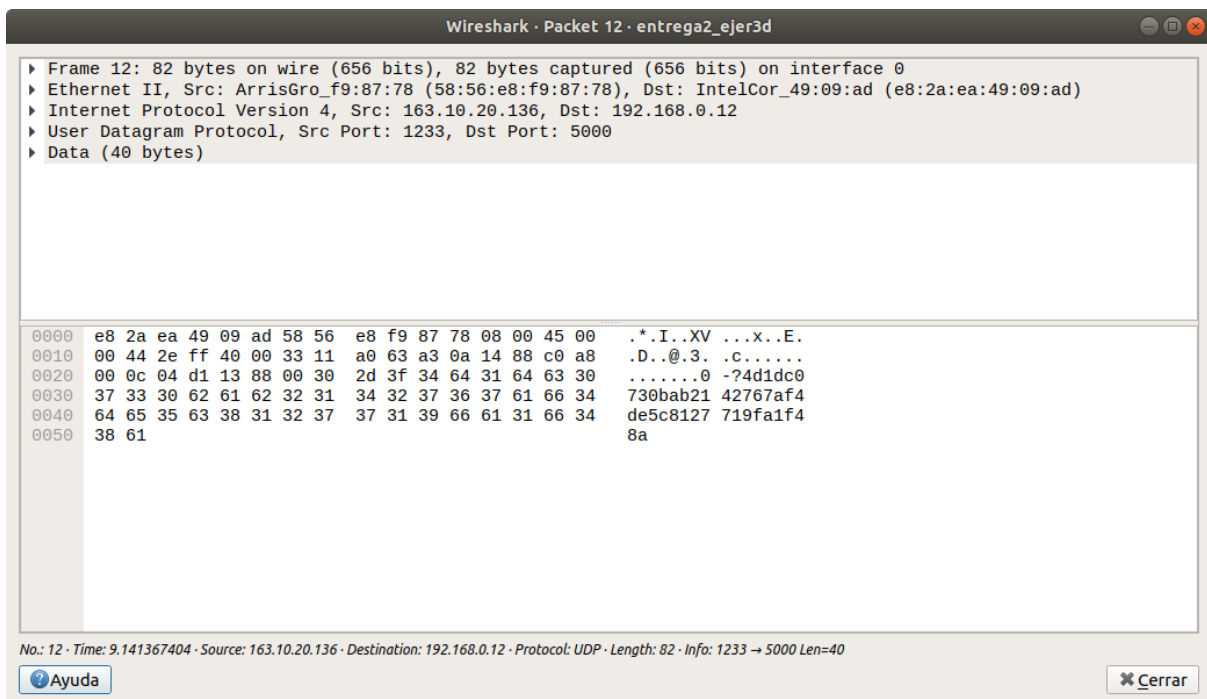
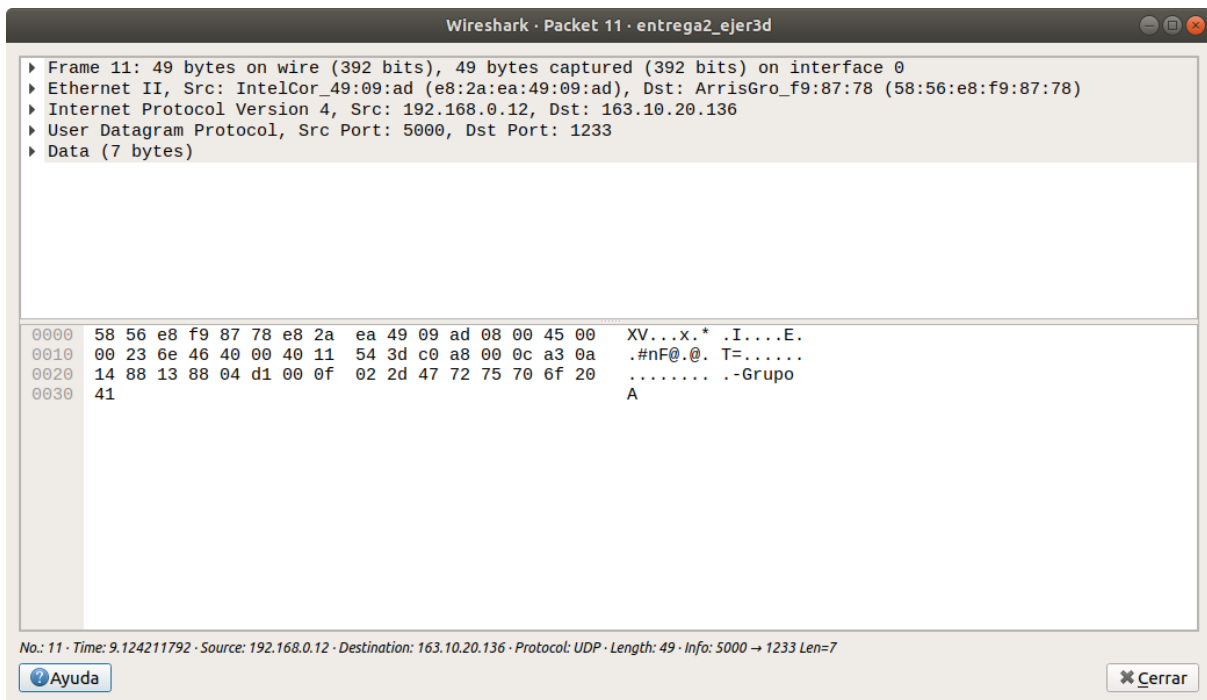
No.	Time	Source	Destination	Protocol	Length	Info
10	6.451325858	AsustekC_98:72:ba	Broadcast	ARP	60	Who has 192.168.0.3? Tell 192.168.0.14
11	9.124211752	192.168.0.12	163.10.20.136	UDP	49	5000 → 1233 Len=7
12	9.141367404	163.10.20.136	192.168.0.12	UDP	82	1233 → 5000 Len=40
13	10.782852845	192.168.0.12	158.85.224.179	TLSv1.2	199	Application Data
14	11.071609811	158.85.224.179	192.168.0.12	TCP	66	443 → 45212 [ACK] Seq=1 Ack=134 Win=5 Len=0 TSval=100369576
15	11.072955659	64.233.186.189	192.168.0.12	TLSv1.2	125	Application Data
16	11.072981253	192.168.0.12	64.233.186.189	TCP	66	52924 → 443 [ACK] Seq=1 Ack=60 Win=1444 Len=0 TSval=1492393
17	11.877161416	158.85.224.179	192.168.0.12	TLSv1.2	176	Application Data
18	11.877181622	192.168.0.12	158.85.224.179	TCP	66	45212 → 443 [ACK] Seq=134 Ack=111 Win=1444 Len=0 TSval=9714

Frame 11: 49 bytes on wire (392 bits), 49 bytes captured (392 bits) on interface 0  
 Ethernet II, Src: IntelCor\_49:09:ad (e8:2a:ea:49:09:ad), Dst: ArrisGro\_f9:87:78 (58:56:e8:f9:87:78)  
 Internet Protocol Version 4, Src: 192.168.0.12, Dst: 163.10.20.136  
 User Datagram Protocol, Src Port: 5000, Dst Port: 1233  
 Data (7 bytes)

0000 58 56 e8 f9 87 78 e8 2a ea 49 09 ad 08 00 45 00 XV...x.\*.I....E.  
 0010 00 23 6e 46 40 00 40 11 54 3d c0 a8 00 0c a3 0a .#nF@.@.T=.....  
 0020 14 88 13 88 04 d1 00 0f 02 2d 47 72 75 70 6f 20 .....-Grupo  
 0030 41 A

Ready to load or capture Packets: 20 - Displayed: 20 (100.0%) Profile: Default





Ahora que el servidor nos responde recibimos del lado del servidor la respuesta de un conjunto de letras y números.

#### 4. Tamaño del payload en UDP

**a- ¿Cuál es la cantidad máxima de datos que se pueden enviar en un datagrama UDP? Modifique el servidor implementado anteriormente para mostrar dicho comportamiento.**

A los campos del puerto destino le sigue un campo obligatorio que indica el tamaño en bytes del datagrama UDP incluidos los datos.

La cantidad máxima de datos que se pueden enviar en un datagrama UDP es de 2 a la 16-1 = 65535 bytes. Incluyendo los 8 bytes del UDP header el máximo es 65527 bytes.

#### 5- Cálculo de checksum en UDP:

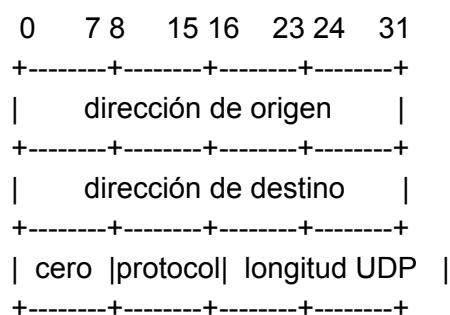
**a- ¿Cómo lo realiza? ¿Cómo es el algoritmo de chequeo? ¿Qué campos involucra?**

La suma de comprobación de UDP proporciona un mecanismo de detección de errores.

Se utiliza para determinar si los bits contenidos en el segmento UDP han sido alterados según se desplazaban desde el origen hasta el destino (por ejemplo, a causa de la existencia de ruido en los enlaces o mientras estaban almacenados en un router). UDP en el lado del emisor calcula el complemento a 1 de la suma de todas las palabras de 16 bits del segmento, acarreando cualquier desbordamiento obtenido durante la operación de suma sobre el bit de menor peso. Este resultado se almacena en el campo suma de comprobación del segmento UDP.

Según la RFC 768, el campo Suma de Control (Checksum) es el complemento a uno de 16 bits de la suma de los complementos a uno de las palabras de la combinación de una pseudo-cabecera construida con información de la cabecera IP, la cabecera UDP y los datos, y rellenada con octetos de valor cero en la parte final (si es necesario) hasta tener un múltiplo de dos octeto.

La pseudo-cabecera que imaginariamente antecede a la cabecera UDP contiene la dirección de origen, la dirección de destino, el protocolo y la longitud UDP. Esta información proporciona protección frente a datagramas mal encaminados. Este procedimiento de comprobación es el mismo que el utilizado en TCP.



Si la suma de control calculada es cero, se transmite como un campo de unos (el equivalente en la aritmética del complemento a uno). Un valor de la suma de control transmitido como un campo de ceros significa que el emisor no generó la suma de control (para depuración o para protocolos de más alto nivel a los que este campo les sea indiferente).

Ejemplo cálculo de una suma de comprobación:

Tenemos las siguientes tres palabras de 16 bits:

0110011001100000

0101010101010101

1000111100001100

La suma de las dos primeras palabras de 16 bits es:

0110011001100000

0101010101010101

-----

1011101110110101

Sumando la tercera palabra a la suma anterior, obtenemos,

1011101110110101

1000111100001100

-----

0100101011000010

Se observa que en esta última suma se produce un desbordamiento, el cual se acarrea sobre el bit de menor peso. El complemento a 1 se obtiene convirtiendo todos los 0 en 1 y todos los 1 en 0. Por tanto, el complemento a 1 de la suma 0100101011000010 es 1011010100111101, que es la suma de comprobación. En el receptor, las cuatro palabras de 16 bits se suman, incluyendo la suma de comprobación. Si no se han introducido errores en el paquete, entonces la suma en el receptor tiene que ser 1111111111111111. Si uno de los bits es un 0, entonces sabemos que el paquete contiene errores.

**b- Utilizando el emulador CORE: Descargue de la plataforma la topologia-udp.imn. Desde n1, conéctese utilizando nc con n2 al port 8000 y capture tráfico utilizando Wireshark1. Verifique el valor del campo checksum del datagrama UDP recibido y compárelo con el cálculo realizado manualmente por Ud.**

En n2 necesitamos previamente abrir el puerto 8000 utilizando el comando: ncat -l -u 8000, ahora el servidor está a la escucha.

Luego desde n1 utilizamos el comando nc 10.0.0.11 8000 (ip a la que nos queremos conectar de n2 y el puerto).

Vamos a ver si mandamos un mensaje desde la consola de n1 llega a la consola de n2, de la misma manera que si mandamos de n2 un mensaje llega a n1.

entrega2\_ejer5bv2.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/> Expression...

Time	Source	Destination	Protocol	Length	Info
1 0.000000	10.0.0.11	10.0.0.10	UDP	49	8000 → 34064 Len=5
2 0.000070	10.0.0.11	10.0.0.10	UDP	49	8000 → 34064 Len=5
3 1.196220	10.0.2.15	216.58.222.46	TCP	56	34726 → 443 [ACK] Seq=1 Ack=1 Win=39704 Len=0
4 1.196443	10.0.2.15	216.58.222.35	TCP	56	48050 → 443 [ACK] Seq=1 Ack=1 Win=39704 Len=0
5 1.196525	10.0.2.15	216.58.222.46	TCP	56	34729 → 443 [ACK] Seq=1 Ack=1 Win=39704 Len=0
6 1.196629	10.0.2.15	216.58.222.33	TCP	56	57957 → 443 [ACK] Seq=1 Ack=1 Win=39704 Len=0
7 1.196774	216.58.222.46	10.0.2.15	TCP	62	[TCP ACKed unseen segment] 443 → 34726 [ACK] Seq=1 Ack=2 Win=...
8 1.196802	216.58.222.35	10.0.2.15	TCP	62	[TCP ACKed unseen segment] 443 → 48050 [ACK] Seq=1 Ack=2 Win=...
9 1.196808	216.58.222.46	10.0.2.15	TCP	62	[TCP ACKed unseen segment] 443 → 34729 [ACK] Seq=1 Ack=2 Win=...

Frame 1: 49 bytes on wire (392 bits), 49 bytes captured (392 bits) on interface 0

Linux cooked capture

Internet Protocol Version 4, Src: 10.0.0.11, Dst: 10.0.0.10

User Datagram Protocol, Src Port: 8000, Dst Port: 34064

Source Port: 8000

Destination Port: 34064

Length: 13

Checksum: 0x1433 [unverified]

[Checksum Status: Unverified]

[Stream index: 0]

Data (5 bytes)

```

0000 00 03 00 01 00 06 00 00 00 aa 00 01 00 00 08 00 .....
0010 45 00 00 21 f9 fc 40 00 40 11 2c bb 0a 00 00 0b E...!..@. @,....
0020 0a 00 00 0a 1f 40 85 10 00 0d 14 33 68 6f 6c 61 .....!..@. @,....
0030 0a

```

Frame (frame), 49 bytes

Packets: 34 · Displayed: 34 (100.0%) · Load time: 0:0.0 · Profile: Default

entrega2\_ejer5bv2.pcapng

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/> Expression...

Time	Source	Destination	Protocol	Length	Info
1 0.000000	10.0.0.11	10.0.0.10	UDP	49	8000 → 34064 Len=5
2 0.000070	10.0.0.11	10.0.0.10	UDP	49	8000 → 34064 Len=5
3 1.196220	10.0.2.15	216.58.222.46	TCP	56	34726 → 443 [ACK] Seq=1 Ack=1 Win=39704 Len=0
4 1.196443	10.0.2.15	216.58.222.35	TCP	56	48050 → 443 [ACK] Seq=1 Ack=1 Win=39704 Len=0
5 1.196525	10.0.2.15	216.58.222.46	TCP	56	34729 → 443 [ACK] Seq=1 Ack=1 Win=39704 Len=0
6 1.196629	10.0.2.15	216.58.222.33	TCP	56	57957 → 443 [ACK] Seq=1 Ack=1 Win=39704 Len=0
7 1.196774	216.58.222.46	10.0.2.15	TCP	62	[TCP ACKed unseen segment] 443 → 34726 [ACK] Seq=1 Ack=2 Win=...
8 1.196802	216.58.222.35	10.0.2.15	TCP	62	[TCP ACKed unseen segment] 443 → 48050 [ACK] Seq=1 Ack=2 Win=...
9 1.196808	216.58.222.46	10.0.2.15	TCP	62	[TCP ACKed unseen segment] 443 → 34729 [ACK] Seq=1 Ack=2 Win=...

Frame 2: 49 bytes on wire (392 bits), 49 bytes captured (392 bits) on interface 0

Linux cooked capture

Internet Protocol Version 4, Src: 10.0.0.11, Dst: 10.0.0.10

User Datagram Protocol, Src Port: 8000, Dst Port: 34064

Source Port: 8000

Destination Port: 34064

Length: 13

Checksum: 0x1433 [unverified]

[Checksum Status: Unverified]

[Stream index: 0]

Data (5 bytes)

```

0000 00 04 00 01 00 06 00 00 00 aa 00 01 00 00 08 00 .....
0010 45 00 00 21 f9 fc 40 00 40 11 2c bb 0a 00 00 0b E...!..@. @,....
0020 0a 00 00 0a 1f 40 85 10 00 0d 14 33 68 6f 6c 61 .....!..@. @,....
0030 0a

```

Details at: [http://www.wireshark.org/docs/wsug\\_html\\_unlinked/ChAdvChecksums.html](http://www.wireshark.org/docs/wsug_html_unlinked/ChAdvChecksums.html) (udp.checksum), 2 byte

Packets: 34 · Displayed: 34 (100.0%) · Load time: 0:0.0 · Profile: Default

Checksum emisor: 0x1433, si lo convertimos a binario nos queda 0001010000110011, su complemento es 1110101111001100.

Checksum receptor: 0x1433, si lo convertimos a binario nos queda 0001010000110011.

La suma de los ambos valores es:

1110101111001100

0001010000110011

-----

1111111111111111 → No se han introducido errores en el paquete