DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico I

Wiretapping

Teoría de las Comunicaciones

Integrante	LU	Correo electrónico
Fernández, Gonzalo	836/10	gpfernandezflorio@gmail.com
Aleman, Damián Eliel	377/10	damianealeman@gmail.com
Pizzagalli, Matías	257/12	matipizza@gmail.com

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

Facultad de Ciencias Exactas y Naturales

Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón I/Planta Baja)

Intendente Güiraldes 2610 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Fax: (++54 +11) 4576-3300

http://www.exactas.uba.ar

Índice

1. Introducción Teórica

2. Desarrollo

Para la implementación de las consignas pedidas se utilizó el lenguaje de programación `python`, tal como fue recomendado por la cátedra. Para acceder a la placa de red se utilizó el paquete `scapy` que provee funciones específicas para ello.

2.1. Primera consigna: capturando tráfico

2.1.1. Ejercicio 1

El código que implementa la herramienta que escucha pasivamente los paquetes Ethernet de la red es `e1.py` y se encuentra en el directorio `src`. Toma como parámetro opcional un entero que se traduce en la cantidad de segundos que va a permanecer activo. El valor por defecto es 10 segundos. La función `main` del script utiliza la función `sniff` del paquete `scapy` pasándole como parámetro de `timeout` el parámetro ingresado (o 10 si no se ingresó ninguno) y como parámetro de `prn` la función `monitor_callback` que toma un paquete de red y lo imprime mediante un llamado a la función `show`. La forma de ejecutarlo es

```
$ sudo python e1.py [TIMEOUT]
```

Notar que para ejecutarlo se necesitan permisos de administrador ya que la función `sniff` de `scapy` necesita permisos para acceder a la placa de red.

2.1.2. Ejercicio 2

El código que implementa la herramienta para calcular la entropía de la fuente S en la red local es `e2.py` y se encuentra en el directorio `src`. Este programa es una modificación del anterior, `e1.py`. La principal modificación es que los paquetes obtenidos por el llamado a `sniff` ahora se almacenan para operar sobre ellos luego. Se anula el parámetro `prn` pero se conserva el `timeout` (el cuál sigue siendo un parámetro opcional del programa). Una vez recibidos todos los paquetes, se obtiene el tipo de cada uno mediante el atributo `type`. En este punto encontramos que no todos los paquetes capturados poseen dicho atributo, así que atrapamos una excepción al leer el tipo. En caso de saltar la excepción, consideramos que el paquete tiene tipo $0x0000$ y lo imprimimos llamando a `monitor_callback`. A través de la función de mapeo `map_number_to_name` convertimos a una cadena el valor del tipo del paquete. Esta función utiliza un diccionario basado en la siguiente tabla[1]:

[1] https://en.wikipedia.org/wiki/EtherType

0x0800	IPv4		0x8847	MPLS Unicast		0x88CD	SERCOS III
0x0806	ARP		0x8848	MPLS Multicast		0x88E1	HomePlug AV
0x0842	WakeOn LAN		0x8863	PPPoE Discovery		0x88E3	MRP
0x22F3	IETF TRILL		0x8864	PPPoE Session		0x88E5	MAC security
0x6003	DECnet		0x8870	Jumbo		0x88E7	PBB
0x8035	RARP		0x887B	HomePlug 1.0		0x88F7	PTP
0x809B	Ethertalk		0x888E	802.1X		0x8902	CFM
0x80F3	AARP		0x8892	PROFINET		0x8906	FCoE
0x8100	802.1Q		0x889A	SCSI		0x8914	FCoE Init
0x8137	IPX		0x88A2	ATA		0x8915	RoCE
0x8204	QNX Qnet		0x88A4	EtherCAT		0x891D	TTE
0x86DD	IPv6		0x88A8	802.1ad		0x892F	HSR
0x8808	EFC		0x88AB	Powerlink		0x9000	ECTP
0x8819	CobraNet		0x88CC	LLDP			

Luego se utiliza la función `Counter` de `python` para generar el diccionario `cantidades` cuyas claves son los tipos de protocolos y sus respectivos valores son la cantidad de paquetes de tal tipo. A partir de este diccionario (el cual se imprime para verificación) se calcula la probabilidad de cada tipo. Finalmente se calcula la entropía de la fuente utilizando la probabilidad de cada tipo y se la imprime. La forma de ejecutarlo es:

```
$ sudo python e2.py [TIMEOUT]
```

2.1.3. Ejercicio 3

El código que implementa la herramientade distinción de nodos (hosts) de la red, basada únicamente en paquetes que utilizan el protocolo ARP es `e3.py` y se encuentra en el directorio `src`. El criterio para la diferenciación de los nodos que se encuentran en la red es ...

Una vez más, implementar este programa consistió en modificar el anterior, ya que la única diferencia entre ambas consignas es el atributo de cada paquete utilizado como símbolo de la fuente. Al llamado a la función `sniff` se le agregó el parámetro `filter=‘‘arp’’` para que sólo se examinen los paquetes `ARP`. En lugar de obtener el tipo de cada paquete, lo que se obtiene es El resto del código es idéntico al de `e2.py`. La forma de ejecutarlo es:

```
$ sudo python e3.py [TIMEOUT]
```

2.2. Segunda consigna: gráficos y análisis

Para la parte de experimentación se implementó otro script de python que realiza una medición y sobre esa medición calcula la entropía para ambas fuentes. De esta forma se pueden comparar los dos análisis realizados sobre una misma muestra, cosa que no habríamos podido si ejecutábamos primero un script y luego el otro. El programa correspondiente es `sniffer.py` y se encuentra en el directorio `src`.

La mayor parte del código es idéntica a la de `e2.py` y `e3.py` combinados. Sin embargo, también se aplicaron algunas optimizaciones. Por ejemplo, ya no se almacenan los paque-

tes capturados por `sniff`, sino que se procesan a medida que se capturan. Para ello, se modificó la función `monitor_callback` de forma que obtenga el tipo (tal como se hizo en `e2.py`) y, en caso de ser un paquete `ARP`, (tal como se hizo en `e3.py`) de cada paquete a medida que son capturados. Es por esto que se volvió a la versión original de `sniff` pasándole como parámetro `prn=monitor_callback`. Tras finalizar la escucha, se generan los diccionarios, se imprimen y se calculan las entropías. Además, se escribe a un archivo los valores de cada diccionario para poder ser graficados como histogramas con `gnuplot`. La forma de ejecutarlo es:

```
$ sudo python sniffer.py FILE_PREFIX [TIMEOUT]
```

El parámetro de `timeout` sigue siendo opcional (10 segundos por defecto). Los archivos de salida se componen del prefijo `FILE_PREFIX` pasado como parámetro obligatorio y las cadenas `Protocolos`, `IpsSrcArp` o `IpsDstArp`, según corresponda. Estos archivos se guardan en la carpeta `mediciones`.

2.2.1. Experimento 1: Red hogareña, cableada, 10 mintos

Este experimento consiste en ...

2.2.2. Experimento 2: Red hogareña, inalámbrica, 10 mintos

Este experimento consiste en ...

2.2.3. Experimento 3: Red pública, inalámbrica, 10 minutos

Este experimento consiste en ...

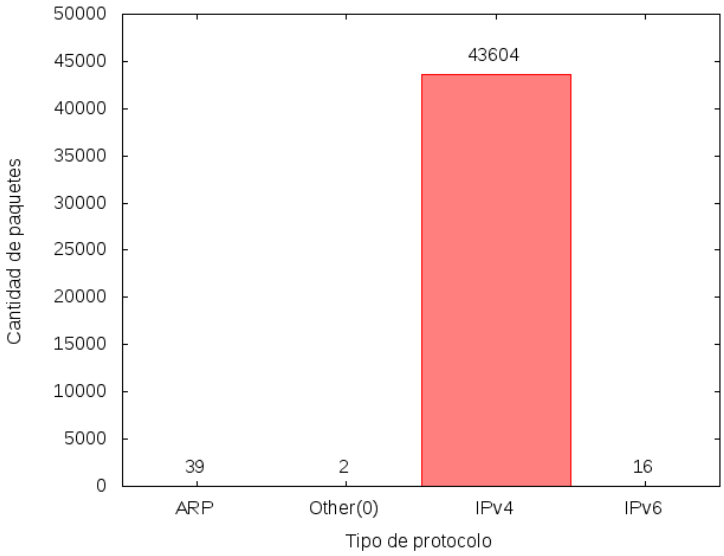
2.2.4. Experimento 4: Red pública, inalámbrica, 60 minutos

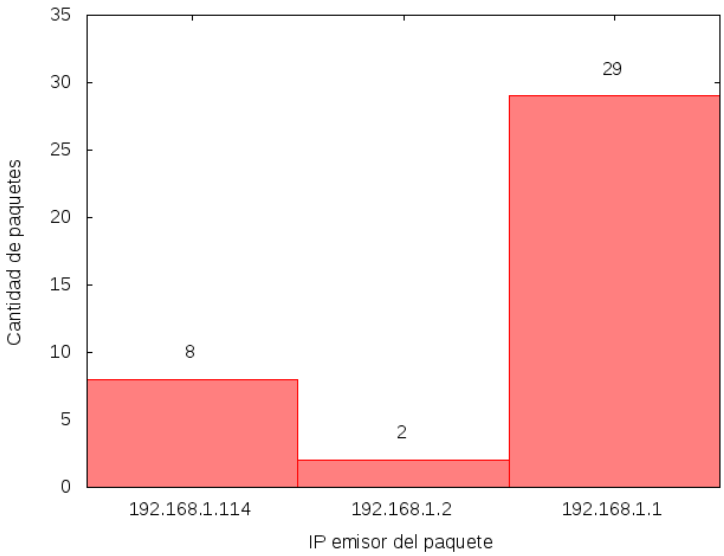
Este experimento consiste en ...

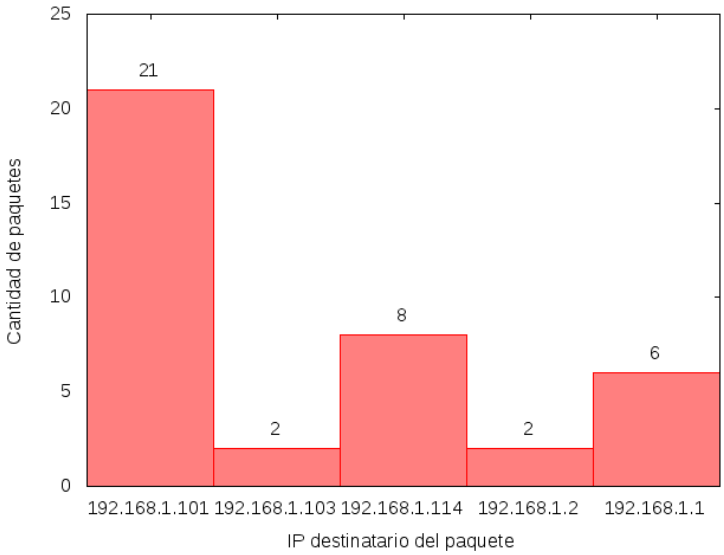
3. Resultados

En la carpeta `mediciones` se encuentran los archivos de salida correspondientes a cada experimento. Por cada uno, se adjunta un archivo `README.txt` con la información conocida sobre la red en cuestión y otro arhivo `exceptions.txt` que describe los paquetes sin tipo capturados. Los nombres de los experimentos denotan el tipo de red, el tipo de conexión y la duración del experimento en minutos.

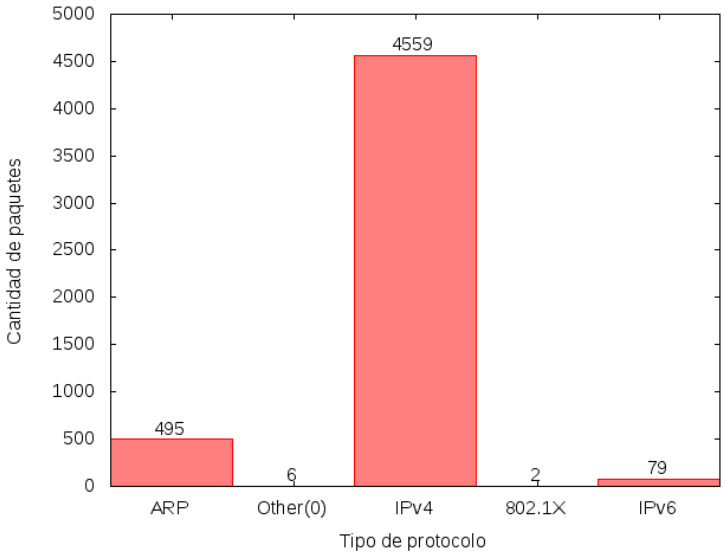
3.0.5. Experimento 1: Red hogareña, cableada, 10 mintos (`home-eth-10`)

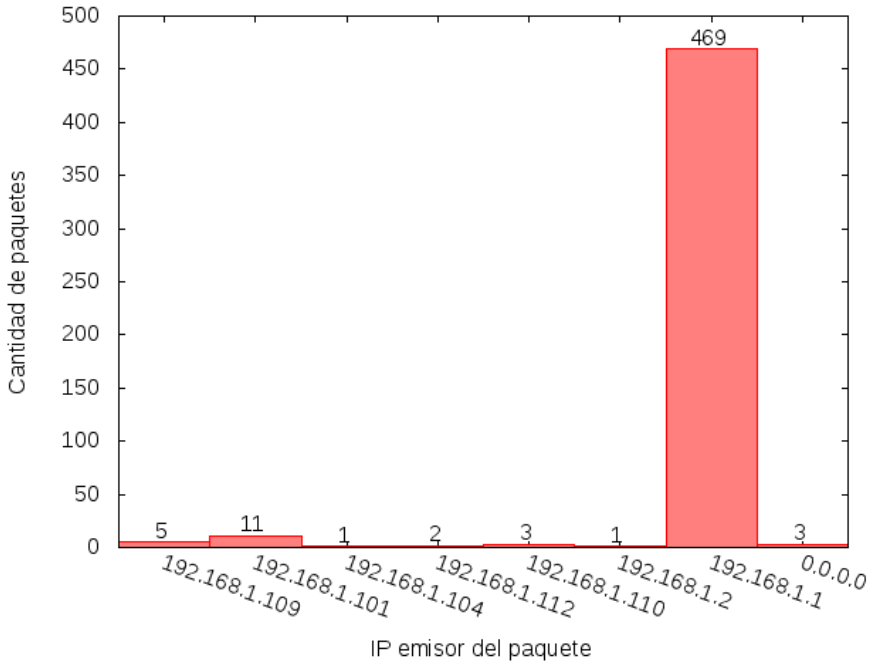


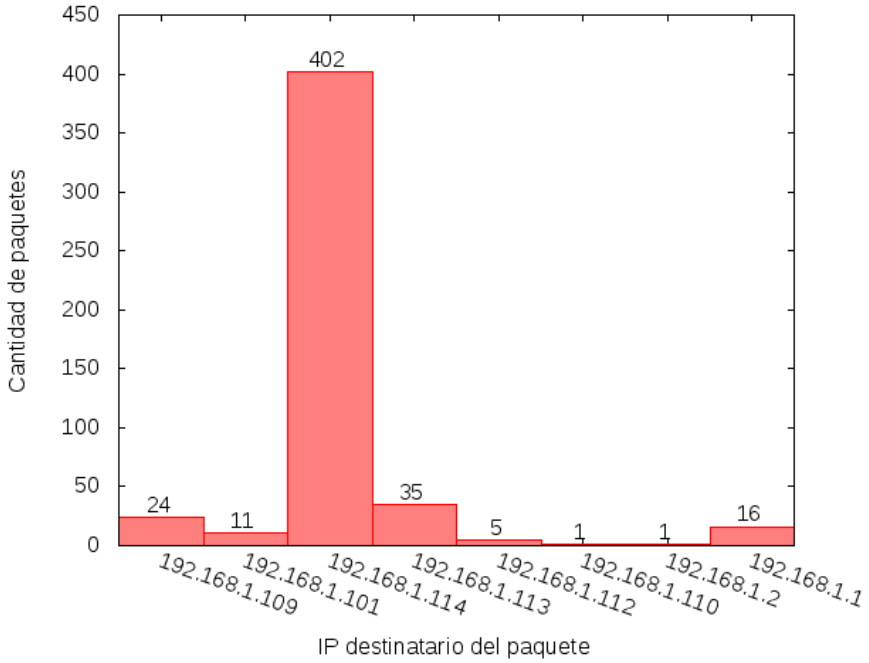




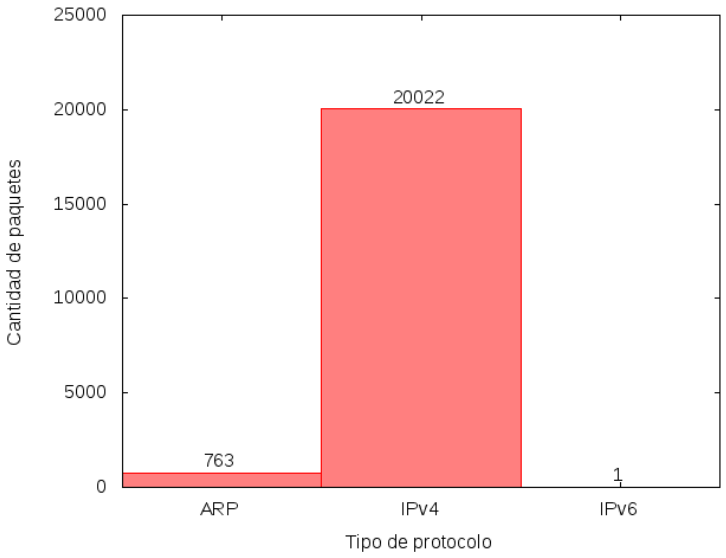
3.0.6. Experimento 2: Red hogareña, inalámbrica, 10 mintos (`home-wfi-10`)

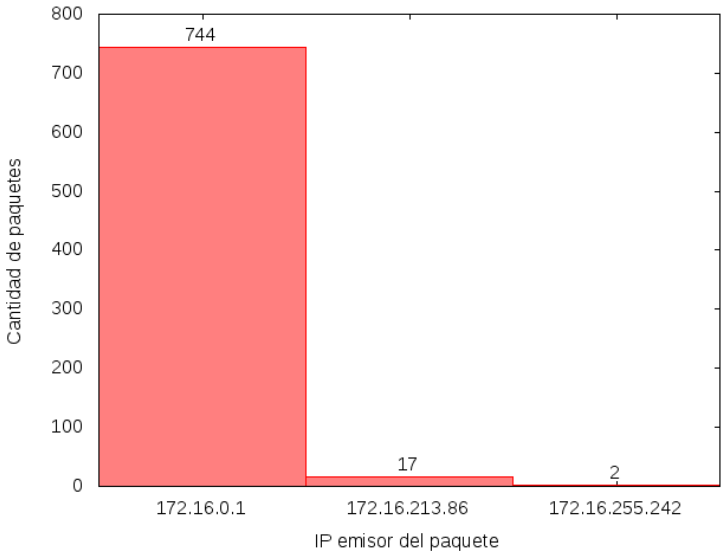


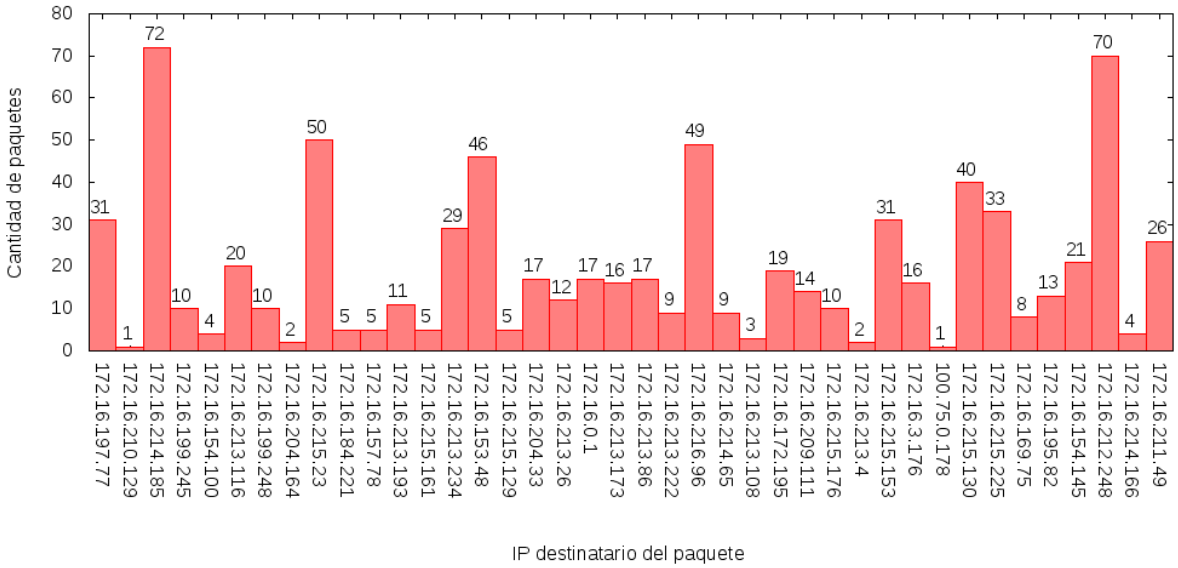




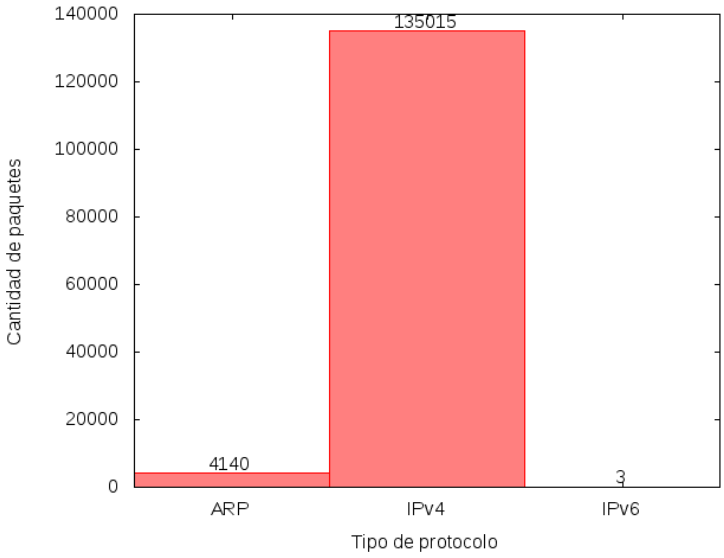
3.0.7. Experimento 3: Red pública, inalámbrica, 10 minutos

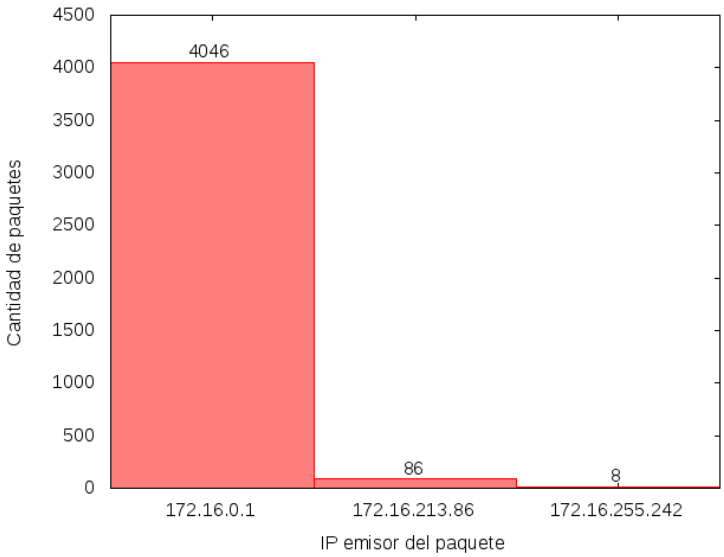


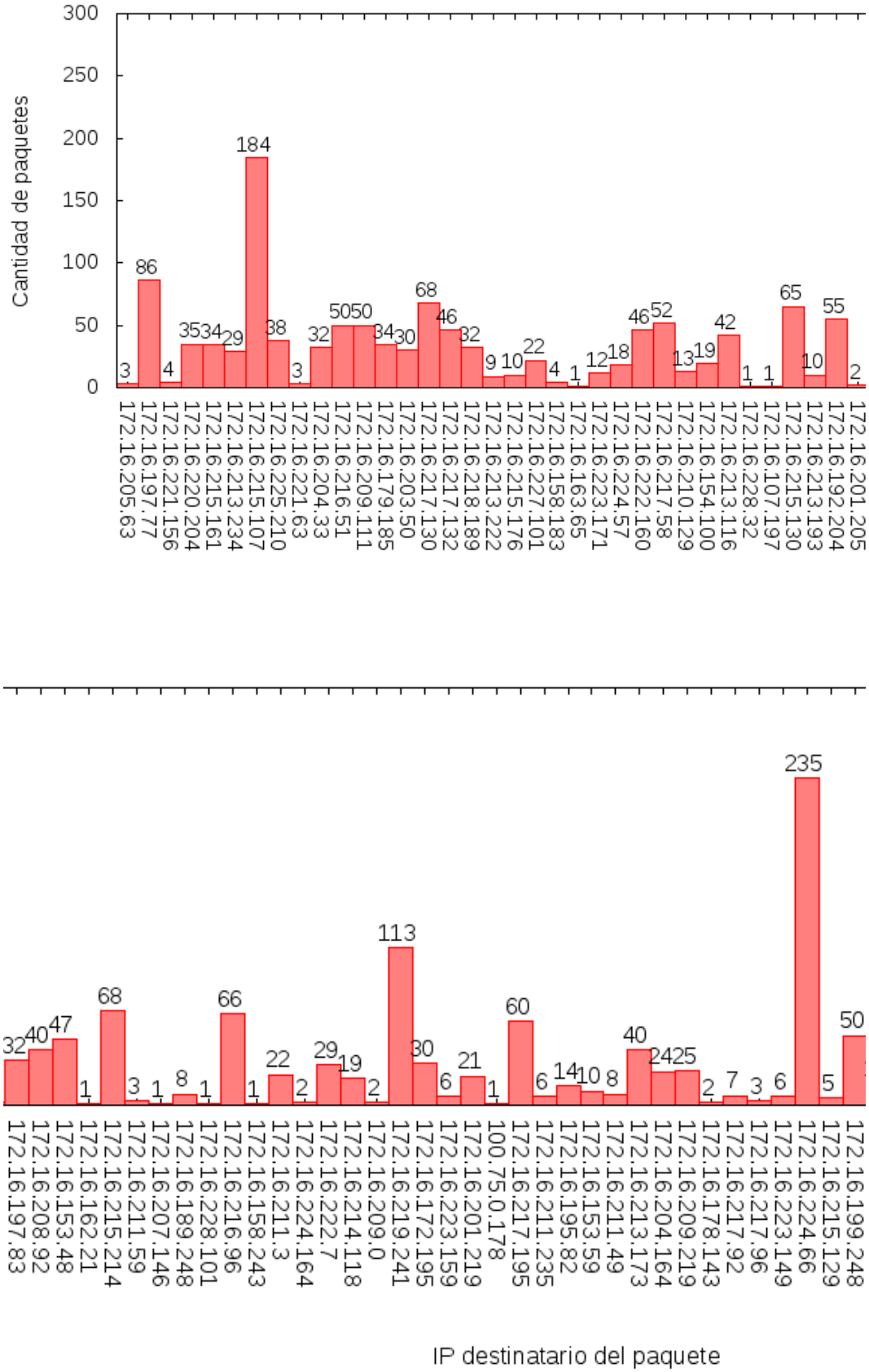

800
700
600
500
400
300
200
100
0
Cantidad de paquetes
744
17
2
172.16.0.1
172.16.213.86
172.16.255.242
IP emisor del paquete

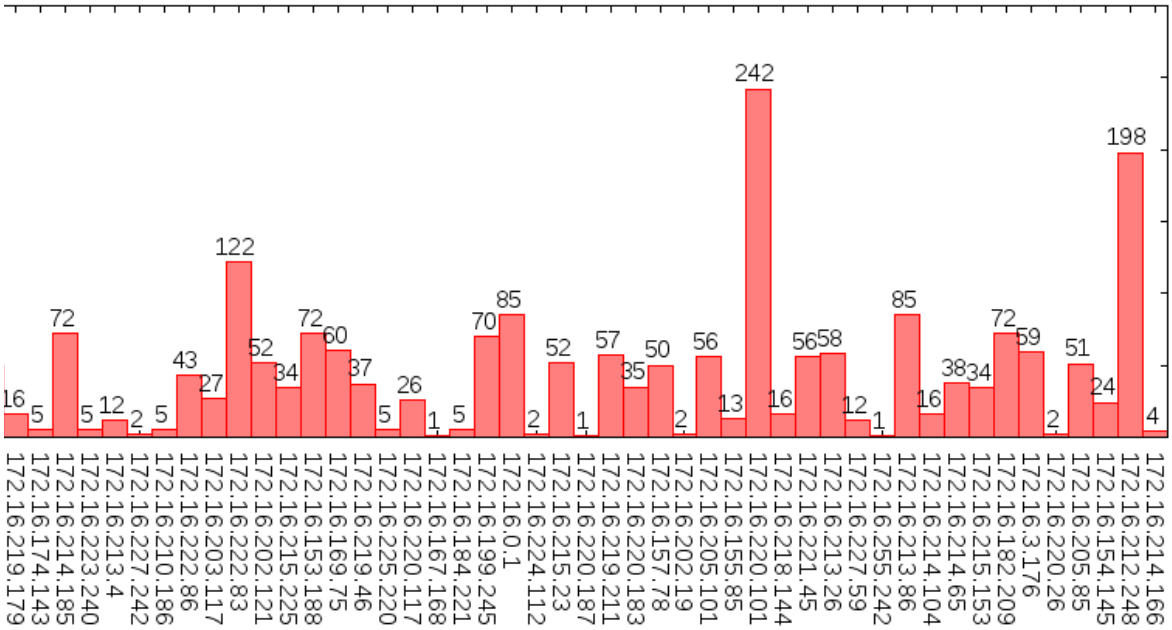

80
70
60
50
40
30
20
10
0
Cantidad de paquetes
31
1
72
10
4
20
10
2
50
5
5
11
5
29
46
5
17
12
17
16
17
9
49
9
3
19
14
10
2
31
16
1
40
33
8
13
21
70
4
26
172.16.197.77
172.16.210.129
172.16.214.185
172.16.199.245
172.16.154.100
172.16.213.116
172.16.199.248
172.16.204.164
172.16.215.23
172.16.184.221
172.16.157.78
172.16.213.193
172.16.215.161
172.16.213.234
172.16.153.48
172.16.215.129
172.16.204.33
172.16.213.26
172.16.0.1
172.16.213.173
172.16.213.86
172.16.213.222
172.16.216.96
172.16.214.65
172.16.213.108
172.16.172.195
172.16.209.111
172.16.215.176
172.16.213.4
172.16.215.153
172.16.3.176
100.75.0.178
172.16.215.130
172.16.215.225
172.16.169.75
172.16.195.82
172.16.154.145
172.16.212.248
172.16.214.166
172.16.211.49
IP destinatario del paquete

3.0.8. Experimento 4: Red pública, inalámbrica, 60 minutos






Cantidad de paquetes
300
250
200
150
100
50
0
3 86 4 35 34 29 184 38 3 32 50 50 34 30 68 46 32 9 10 22 4 1 12 18 46 52 13 19 42 1 1 65 10 55 2
172.16.205.63
172.16.197.77
172.16.221.156
172.16.220.204
172.16.215.161
172.16.213.234
172.16.215.107
172.16.225.210
172.16.221.63
172.16.204.33
172.16.216.51
172.16.209.111
172.16.179.185
172.16.203.50
172.16.217.130
172.16.217.132
172.16.218.189
172.16.213.222
172.16.215.176
172.16.227.101
172.16.158.183
172.16.163.65
172.16.223.171
172.16.224.57
172.16.222.160
172.16.217.58
172.16.210.129
172.16.154.100
172.16.213.116
172.16.228.32
172.16.107.197
172.16.215.130
172.16.213.193
172.16.192.204
172.16.201.205
32 40 47 1 68 3 1 8 1 66 1 22 2 29 19 2 113 30 6 21 1 60 6 14 10 8 40 24 25 2 7 3 6 235 5 50
172.16.197.83
172.16.208.92
172.16.153.48
172.16.162.21
172.16.215.214
172.16.211.59
172.16.207.146
172.16.189.248
172.16.228.101
172.16.216.96
172.16.158.243
172.16.211.3
172.16.224.164
172.16.222.7
172.16.214.118
172.16.209.0
172.16.219.241
172.16.172.195
172.16.223.159
172.16.201.219
100.75.0.178
172.16.217.195
172.16.211.235
172.16.195.82
172.16.153.59
172.16.211.49
172.16.213.173
172.16.204.164
172.16.209.219
172.16.178.143
172.16.217.92
172.16.217.96
172.16.223.149
172.16.224.66
172.16.215.129
172.16.199.248
IP destinatario del paquete



4. Conclusiones

En base a los experimentos realizados, podemos concluír que ...

5. Apéndice

5.1. Enunciado

Referencias