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

Esta somera introducción ...

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. Implementación (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

Para poder distinguir los nodos de un red, se nos ocurrio proponer como posibles fuentes de información la IP destino y la IP origen de los paquetes ARP. Nos parece interesante medir ambas fuentes de información propuestas para luego poder analizarlas y llegar a distinguir cuales son los nodos que mas envian y mas reciben paquetes y de esta forma comprender la topologia de la red en cuestion.

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`. Luego, en lugar de obtener el tipo de cada paquete, que ya sabemos que todos son ARP, lo que se obtiene es la IP origen y la IP destino de cada paquete. El resto del código es idéntico al de `e2.py`. La forma de ejecutarlo es:

El código que implementa la herramienta de 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`.

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

2.2. Experimentación (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 las tres fuentes. De esta forma se

pueden comparar los 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 paquetes 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`, la IP origen y la IP destino (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

2.2.1.1. Medición cableada de 10 minutos

Este experimento consiste en ... Notebook conectada por cable ethernet a un router en una red hogareña durante 10 minutos. La IP de la notebook es 192.168.1.114 La IP del router es 192.168.1.1 Se sabe además que hay otro router conectado con la IP 192.168.1.2

2.2.1.2. Medición inalámbrica de 10 minutos

Este experimento consiste en ...

2.2.2. Experimento 2: Red pública

2.2.2.1. Medición inalámbrica de 10 minutos

Este experimento consiste en ...

2.2.2.2. Medición inalámbrica de 60 minutos

Este experimento consistie en ...

2.2.3. Experimento 3: Red laboral

2.2.3.1. Medición inalámbrica de 60 minutos

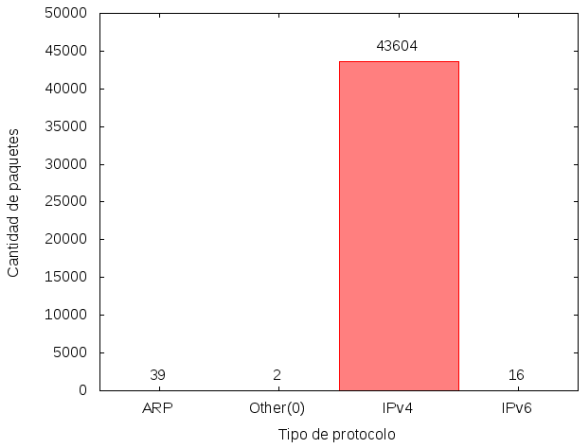
Este experimento consistió en dos mediciones de una hora dos días distintos sobre una misma red.

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.

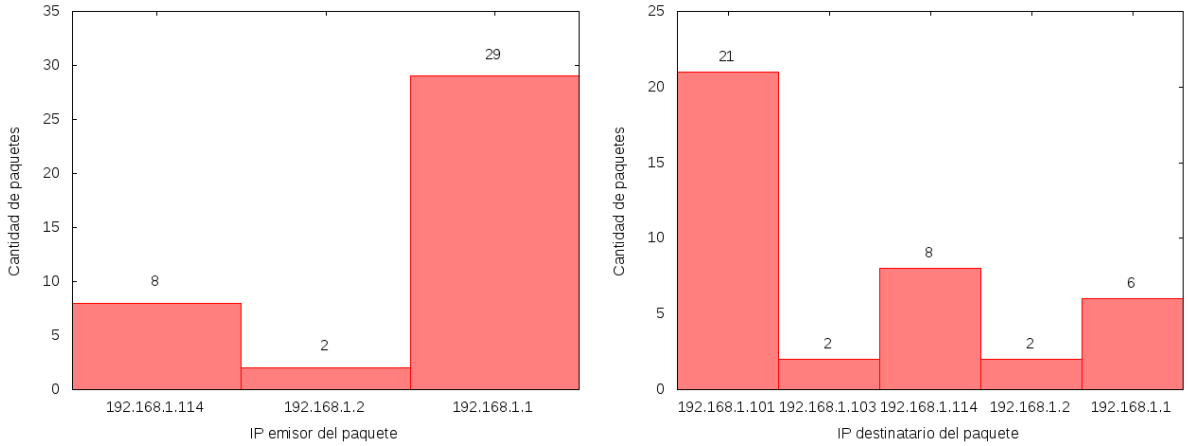
3.1. Experimento 1: Red hogareña

3.1.1. Medición cableada



En este gráfico podemos observar que ARP se usa muy poco en relación con IP. Imaginamos que es porque hay pocos nodos en la red y se aprende rápido la configuración de la red. En cuanto a la diferencia entre los distintas versiones de IP, nuestra hipótesis es que en general vamos a encontrar más paquetes IPv4 que IPv6 ya que todavía no se migró a la versión 6 del protocolo IP. Dicha hipótesis se ve confirmada en este gráfico ya que la relación entre ambas versiones del protocolo es aproximadamente 2725 a 1 a favor de la versión 4. Es claro que el tipo $0x0800$ correspondiente al protocolo IPv4, es un símbolo distinguido de la fuente **S** ya que es el que más aparece, es decir que es el que tiene mayor probabilidad. El resto de los símbolos tienen frecuencias similares y mucho menores a la de $0x0800$.

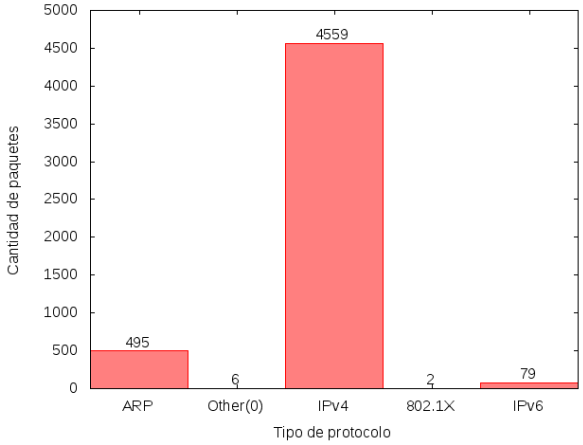
Vemos que para este experimento no tiene sentido considerar la IP del emisor de un paquete para identificar a los nodos de la red ya que sólo tres direcciones (la de la computadora que está ejecutando el experimento y las de los dos routers) son visibles. Esto tiene



sentido al considerar que la computadora que está ejecutando el experimento está conectada al router principal (con dirección **192.168.1.1**) a través de un cable ethernet, por lo que los paquetes emitidos por otros hosts no deberían llegar hasta este equipo. En cuanto a los destinatarios de los paquetes, vemos que el router no representa un símbolo distinguido como sí lo hacía en la fuente de basada en emisores.

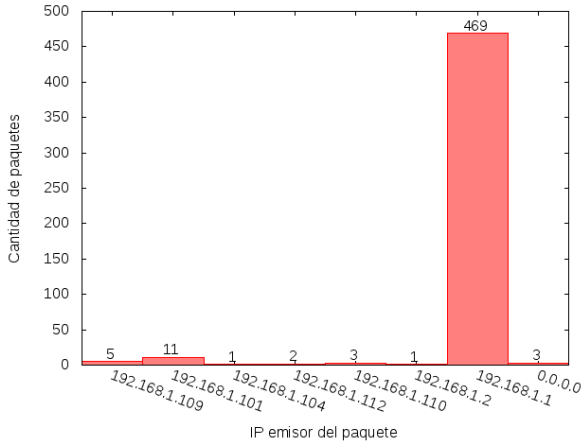
Entropía de los protocolos	0.0157726691884
Entropía de las IPs origen de ARP	1.00639070966
Entropía de las IPs destino de ARP	1.80467296546

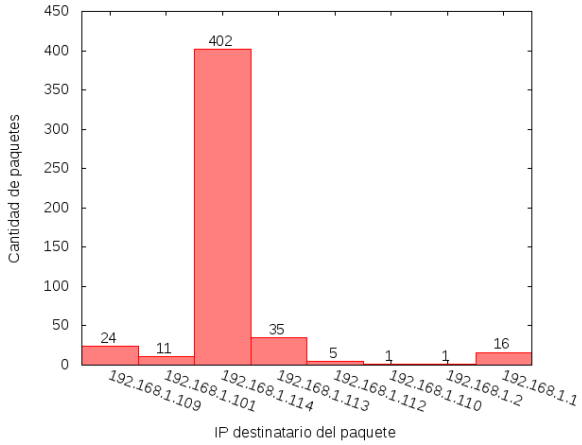
3.1.2. Medición inalámbrica



Otra vez se ve la superioridad de la versión 4 del protocolo IP, no sólo sobre la versión 6 del mismo protocolo, sino sobre el resto de los protocolos. Sin embargo, la proporción es mucho menor. Al comparar este gráfico con el del experimento anterior, notamos dos cambios significativos. Por un lado la cantidad de paquetes de tipo IPv4 es casi 10 veces menor. Por otro lado la cantidad de paquetes de tipo ARP y de tipo IPv6 se multiplicaron varias veces. Si consideramos que la duración fue de 10 minutos para ambos experimentos, llama la atención esta diferencia. Como se puede ver en la siguiente tabla, los paquetes Ipv4 pasaron de ocupar el 99.9 % de los paquetes capturados a ocupar el 88.7 %. Los de tipo IPv6 pasaron del 0.04 % al 1.54 % y los de tipo ARP pasaron del 0.09 % al 9.63 %.

Protocolo	Red cableada	Red Wifi
ARP	0.09 %	9.63 %
IPv4	99.87 %	88.68 %
IPv6	0.04 %	1.54 %
Otros	0.01 %	0.16 %



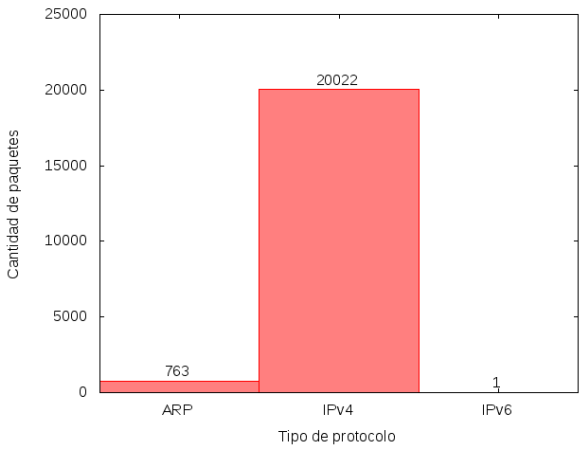


En cada uno de estos gráficos encontramos un nodo distinguido. Del lado de los emisores, la dirección IP distinguida corresponde al router principal de la red, mientras que el nodo al que más paquetes van dirigidos es el host que ejecuta el experimento. El resto de las direcciones mantiene una baja frecuencia de envío y recepción de paquetes. Otro punto a notar es la aparición de la dirección **0.0.0.0** como emisor de 3 paquetes.

Entropía de los protocolos	0.5871665762
Entropía de las IPs origen de ARP	0.420338347182
Entropía de las IPs destino de ARP	1.11098133367

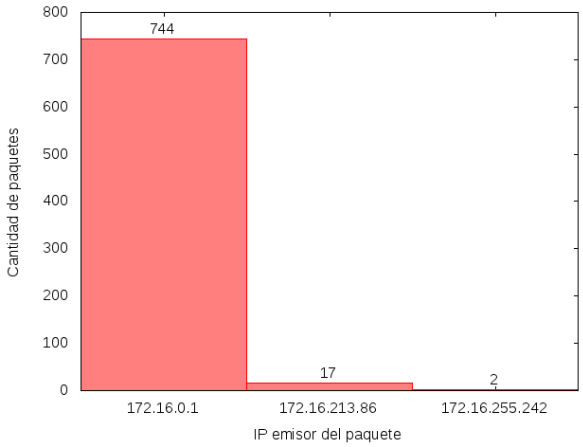
3.2. Experimento 2: Red pública

3.2.1. Medición 10 minutos

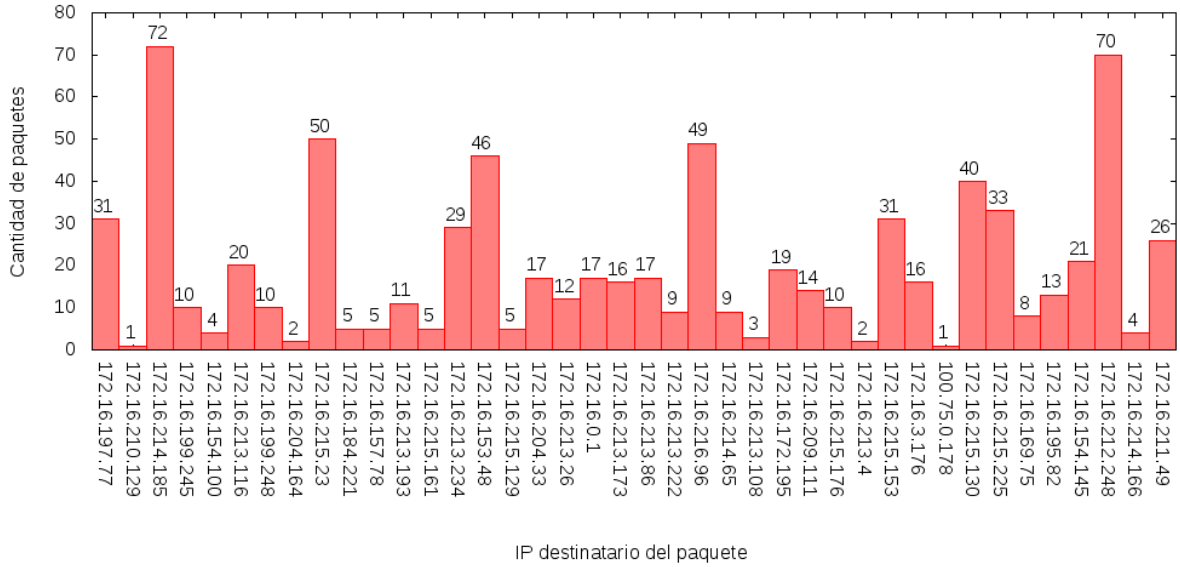


Seguimos viendo en esta red que el protocolo IPv4 es el más utilizado para el envío de paquetes. Es particularmente notoria la escasa cantidad de paquetes de tipo IPv6, mucho

menor que en la red hogareña del experimento anterior. Sí se incrementa la cantidad de paquetes de tipo ARP en la misma cantidad de tiempo, respecto a la red hogareña. Esto puede deberse a que la red pública es considerablemente mayor.



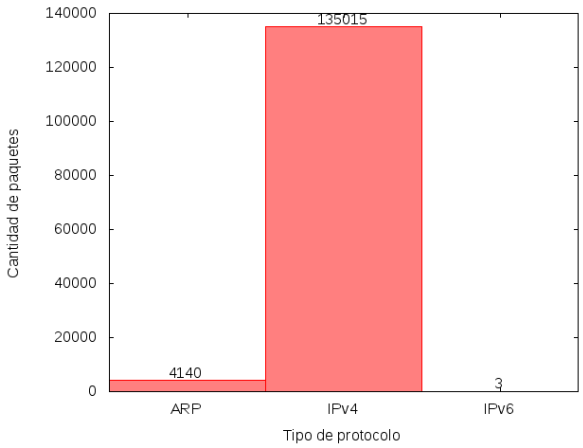
En este caso nos encontramos con un comportamiento inesperado. Más del 97 % de los paquetes capturados fueron emitidos por una misma dirección IP, que podemos suponer que es el router. Por otro lado, sólo pudieron identificarse 3 direcciones IP distintas en la red, a diferencia de lo que había sucedido en la red hogareña, en la que se pudieron identificar varios nodos más. Claramente, la IP **172.16.0.1** es un símbolo distinguido de la fuente.

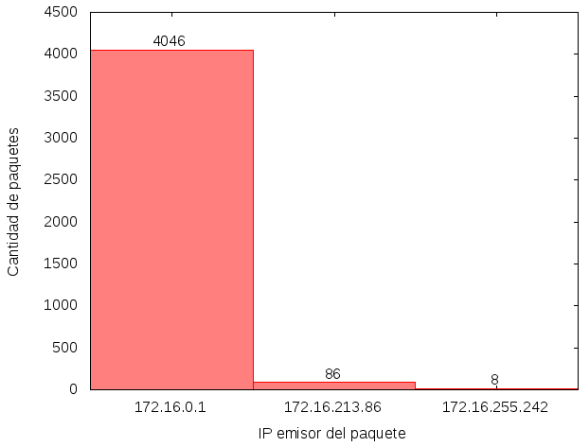


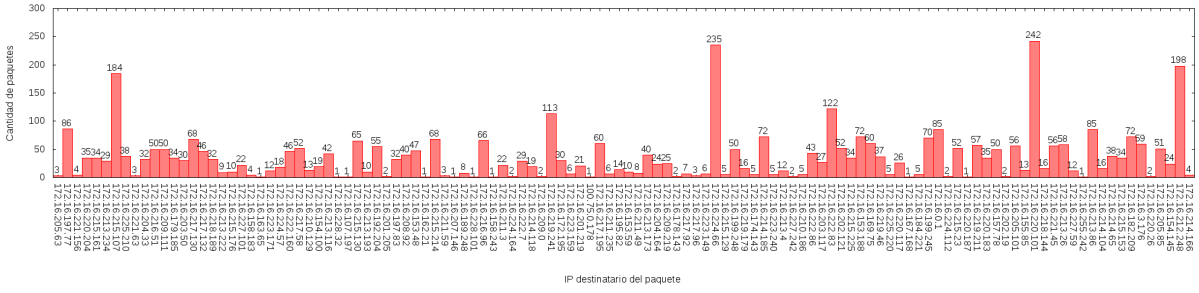
Este gráfico muestra algo más parecido a lo esperado en cuanto a la cantidad de nodos identificados. Lo que es incierto es la razón por la cual los paquetes son dirigidos a tantos nodos distintos pero emitidos por un grupo reducido.

Entropía de los protocolos	0.227743354201
Entropía de las IPs origen de ARP	0.180229910313
Entropía de las IPs destino de ARP	4.77276412705

3.2.2. Medición 60 minutos





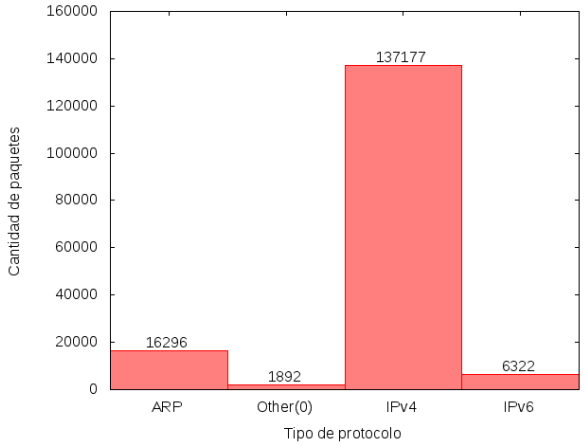


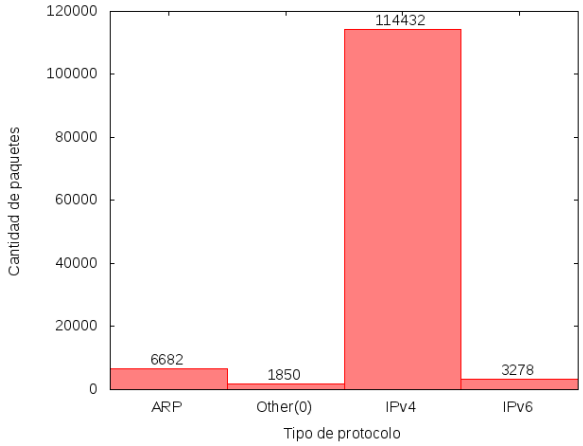
Para poder observar este gráfico mejor, lo agregamos en el Apéndice, en la sección 5.1.1.

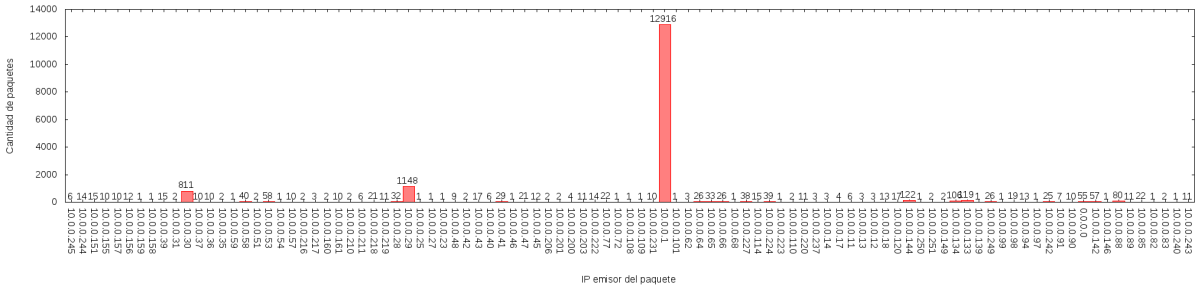
Entropía de los protocolos	0.193502700102
Entropía de las IPs origen de ARP	0.1659063338
Entropía de las IPs destino de ARP	6.07060528117

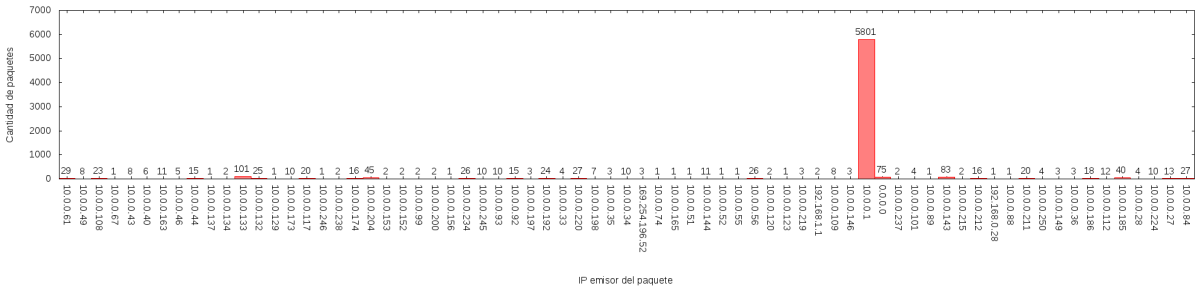
3.3. Experimento 3: Red laboral

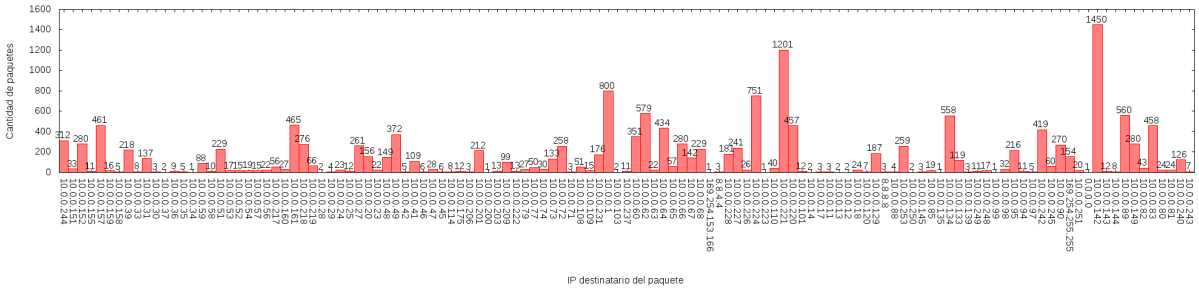
3.3.1. Primera medición 60 minutos

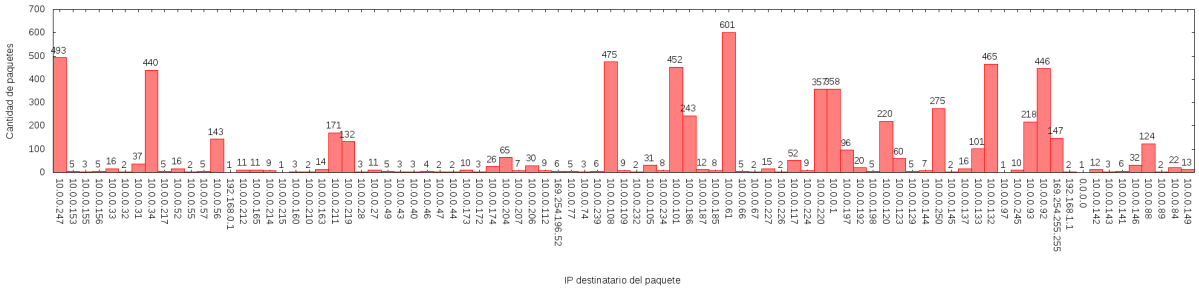






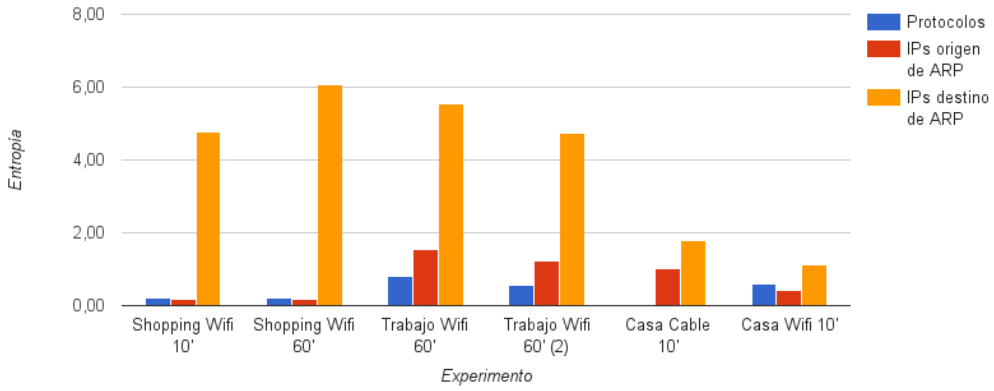






Entropía	Medición 1	Medición 2
Entropía de los protocolos	0.792831613911	0.578907674515
Entropía de las IPs origen de ARP	1.53247970384	1.23402710535
Entropía de las IPs destino de ARP	5.53038672313	4.7457277713

3.4. Entropía



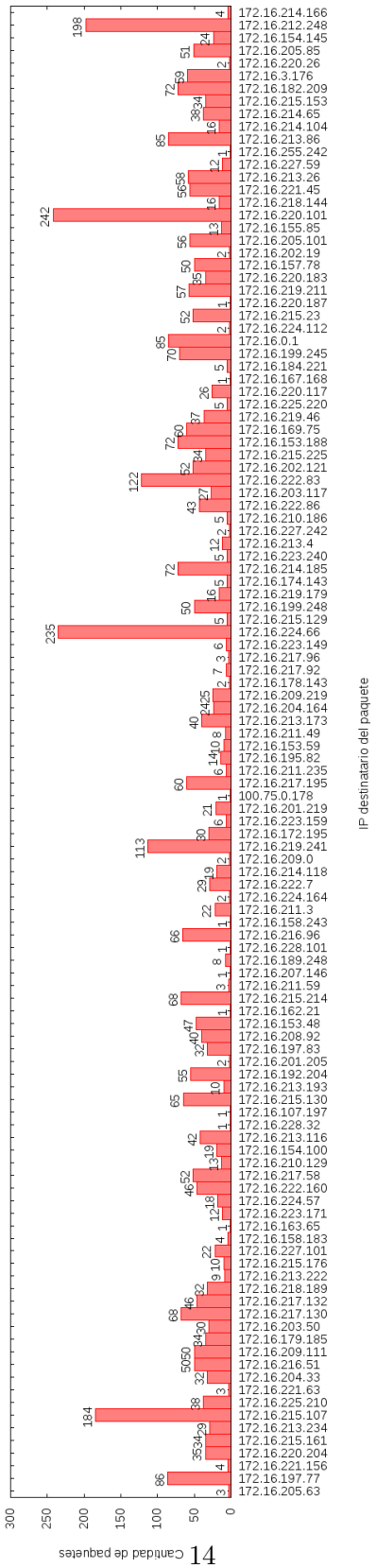
4. Conclusiones

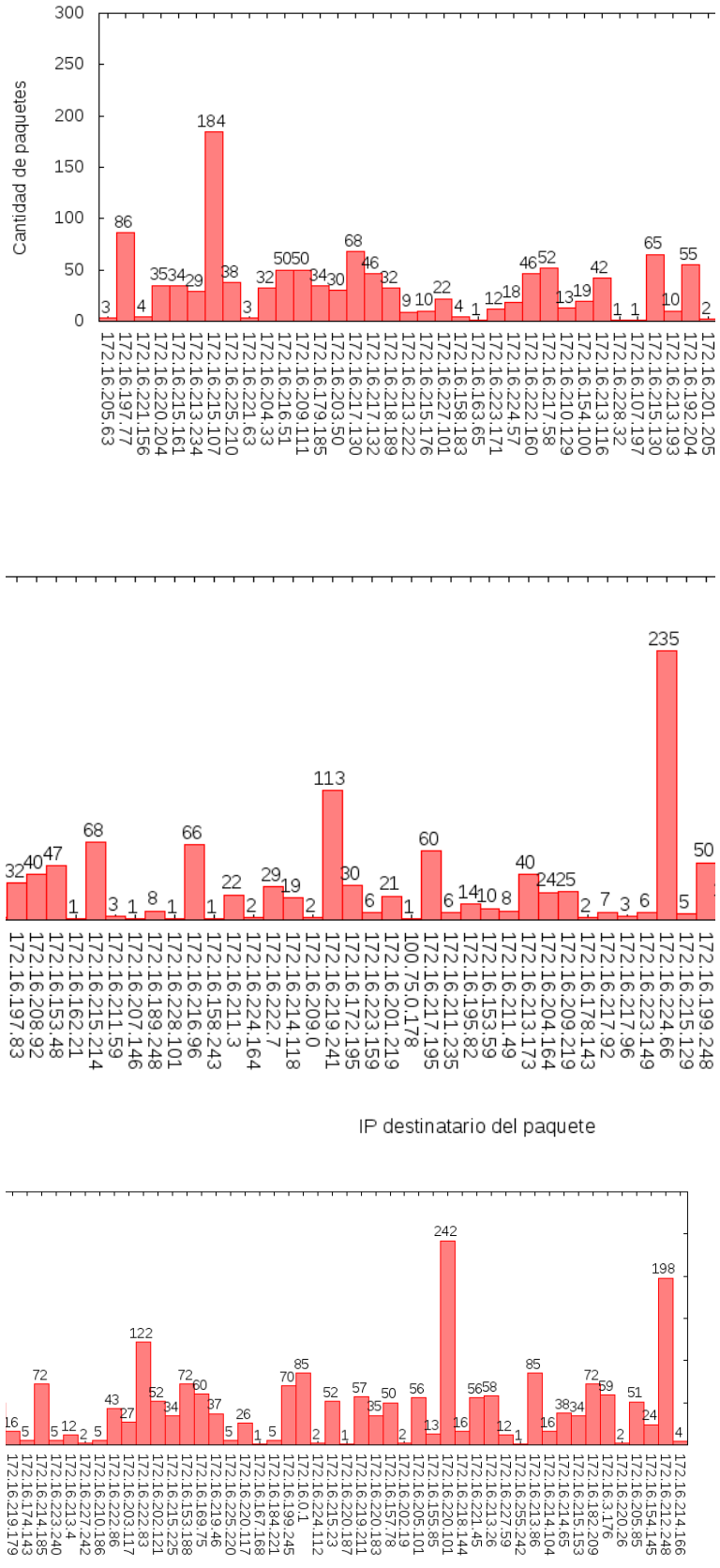
En base a los experimentos realizados, podemos concluír que ...

5. Apéndice

5.1. Figuras ampliadas

5.1.1. Experimento 2: Medición 60 minutos





5.2. Enunciado

TP1: Wiretapping

Teoría de las Comunicaciones
Departamento de Computación
FCEN - UBA

30.03.2016

1. Introducción

El objetivo de este trabajo es utilizar técnicas provistas por la teoría de la información para distinguir diversos aspectos de la red de manera analítica. Además, sugerimos el uso de dos herramientas modernas de manipulación y análisis de paquetes frecuentemente usadas en el dominio de las redes de computadoras: Wireshark [?] y Scapy [?].

2. Normativa

- Fecha de entrega: 20-04-2016.
- El informe deberá haber sido enviado por correo para esa fecha con el siguiente formato:
  to: tdc-doc at dc uba ar
  subject: debe tener el prefijo [tdc-wiretapping] y contener en numero de grupo
  body: nombres de los integrantes y las respectivas direcciones de correo electrónico
  attachments: el informe en formato pdf + el código fuente en formato zip. Junto con los fuentes debe haber un archivo de texto con indicaciones para su uso o un makefile en su defecto.
- No esperar confirmación. Todos los mails llegan a la lista a menos que reciban una respuesta indicando explícitamente que el mail fue rechazado. Los avisos por exceso de tamaño no son rechazos.

3. Enunciado

3.1. Introducción

Sean $p_1..p_i$ los paquetes que se transmiten en un enlace. Podemos conocer los protocolos que encapsulan los paquetes con el campo type del frame de capa de enlace (p_i.type en Scapy).

Se define el conjunto de símbolos $S = \{s_1 \cdots s_n\}$ siendo $s_i = p_i.type \ / \ p_i \in P$ entre los instantes de tiempo $[t_i, t_f]$. Podemos pensar la fuente S que tiene como símbolos a los protocolos utilizados en la red en ese intervalo.

Por último, en el marco de una fuente de información, podemos pensar a un **símbolo como distinguido** cuando sobresale del resto en términos de la información que provee.

3.2. Primera consigna: capturando tráfico

1. Implementar una herramienta que escuche pasivamente los paquetes Ethernet de la red.
2. Adaptar la herramienta del punto anterior para calcular la entropıa de la fuente S en la red local.
3. Proponga una fuente de información S1 con el objetivo de distinguir, en lugar de los protocolos como hace S, los nodos (hosts) de la red. La distinción de S1 debe estar basada únicamente en paquetes que utilicen el protocolo ARP y el criterio para la diferenciación lo deberá establecer el grupo.

3.3. Segunda consigna: gráficos y análisis

Utilizando estas herramientas, realizar experimentos para analizar:

- Los protocolos distinguidos.
- La incidencia de paquetes ARP en la red.
- Los nodos distinguidos.

Se deben realizar tantos experimentos como cantidad de miembros tenga el grupo. Además, las capturas deben ser lo más extensas posibles (tf - ti >10 minutos). En la medida de lo posible, intentar capturar en al menos una red que no sea controlada (trabajo, shopping, etc).

Los análisis deben estar basados en conceptos formales de la teoría de la información. O sea, se debe analizar qué símbolos son significativos en cada red, viendo la diferencia entre su información y la entropía de la fuente.

Por último, el informe debe seguir la siguiente estructura: somera introducción, métodos y condiciones de cada experimento, resultados y conclusión. La presentación de los resultados debe efectuarse mediante gráficos y su correspondiente análisis. Sugerimos, entre otros, histogramas (de IPs y protocolos) con cortes en los valores de entropía. Se valorará especialmente en esta consigna la creatividad y el análisis propuesto. Recomendamos entonces pensar cómo resultará más efectivo presentar la información recopilada.

Referencias

[1] RFC 826 (ARP) `http://tools.ietf.org/html/rfc826`

[2] Wireshark (página web oficial) `http://www.wireshark.org`

[3] Scapy (página web oficial) `http://www.secdev.org/projects/scapy/`

[4] OUI (IEEE) `http://standards.ieee.org/develop/regauth/oui/oui.txt`