Teoría de las Comunicaciones

Segundo Cuatrimestre de 2012

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Taller de Capa de Enlace

Reentrega

Taller N°1

Integrante	LU	Correo electrónico
Mancuso, Emiliano	597/07	emiliano.mancuso@gmail.com
Mataloni, Alejandro	706/07	amataloni@gmail.com
Gonzalez, Matias	453/07	curtu_infinito73@hotmail.com

Reservado para la catedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

Índice

Índice **2**

1. Primera consigna **3**

2. Segunda consigna **3**

3. Tercera consigna **4**

1. Primera consigna

Utilizamos *Scapy* para implementar un pequeño script que, dada una dirección IP, realiza un pedido por la MAC Address y muestra en pantalla la respuesta en caso de recibir alguna.

El script es el siguiente:

```
pkt = ARP(pdst=sys.argv[1], op="who-has");
response = sr1(pkt)
response.show()
```

Lo interesante es ver que ocurre en algunos casos:

- dirección inexistente = el script se cuelga esperando la respuesta que nunca llega.
- dirección de la maquina de origen = el paquete ARP no se envía.
- dirección Broadcast = el script se cuelga esperando la respuesta que nunca llega.
- dirección de red = el script se cuelga esperando la respuesta que nunca llega.

Agregando un **timeout** podemos resolver el problema de que se cuelga el script.

2. Segunda consigna

Implementamos un script en *Scapy* para escuchar pasivamente en la red y capturar cada mensaje ARP enviado. En el mismo script cuando capturamos el mensaje, traducimos los datos del *vendors* y los imprimimos por pantalla.

El script es el siguiente:

```
# Print pretty vendor
def arp_monitor_callback(pkt):
  vendor_prefix = pkt[ARP].hwsrc[0:8].upper()
  strr =  pkt[ARP].psrc + ": \t" + vendors_dict[vendor_prefix]
  print strr


# Build the vendor dictionary
ins = open("vendorsUtil.txt", "r")
vendors_dict = {}
for line in ins:
  vendors_dict[line[0:8]] = line[9:-1]

ins.close()

if len(sys.argv) == 1:
  print "Listening for 5 seconds.."
  to = 5  # Default value
else:
  to = int(sys.argv[1])

sniff(prn=arp_monitor_callback, filter="arp", store=0, timeout=to)
```

3. Tercera consigna

Para esta parte lo que hicimos fue capturar los paquetes *ARP*, y crear un grafo dirigido de IPs. Cada nodo representa una dirección IP y existe un eje entre dos nodos x e y si y solo si, se observo un request ARP con **source** $\overline{\text{IP}}$ de x y **target** $\overline{\text{IP}}$ de y. Con este grafo describimos mejor la topología de la red, y ver el trafico interno de la misma. Descartamos el uso de un histograma pues nos interesaba mostrar entre quienes se envían los paquetes y no cada cuanto tiempo.

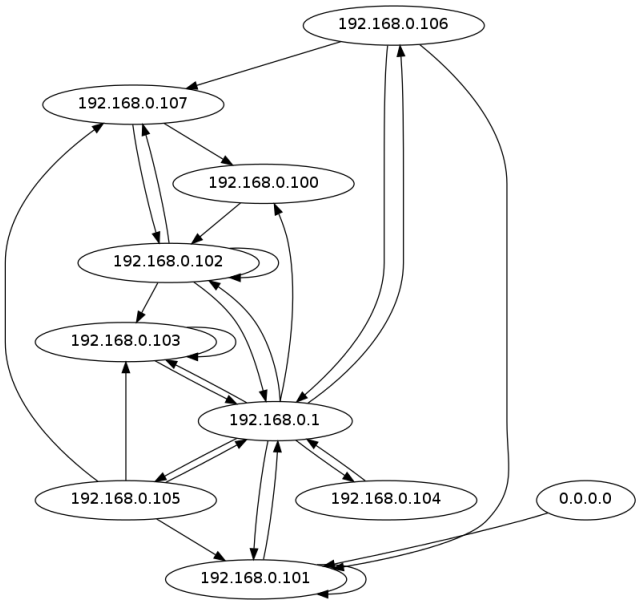

Figura 1: Gráfico dirigido de los distintos request ARP observados.

La muestra fue obtenida de la casa de uno de los integrantes del grupo y la finalizamos al alcanzar los 300 paquetes ARP. Si bien es una red pequeña pensamos que para este taller en particular al trabajar con paquetes ARP no influía tanto la cantidad de dispositivos.

Dispositivos conectados:

- 2 Notebooks
- 2 Celulares
- 2 Desktop
- 1 iPad
- AppleTV

Inmediatamente nos damos cuenta que la **IP 192.168.0.1** es la asignada al router, ya que ésta es la que más se comunica con el resto de los dispositivos. Sin embargo, podemos ver como los dispositivos se comunican entre si y esto se debe a que al momento de las pruebas estábamos utilizando conexiones **ssh** y **Home Sharing** de iTunes para transmitir música. El host **192.168.0.102** es el que interactuaba como *media center*. La mayoría de los dispositivos están siendo sincronizados todo el tiempo ya que están configurados bajo la misma cuenta de Apple.

Otro caso interesante a estudiar es el de los nodos que tienen ejes dirigidos a si mismos. Esto sucede en los paquetes ARP que son del tipo *gratuitous*. Este tipo de paquetes se mandan generalmente cuando un dispositivo se conecta a la red, avisando al resto de su ubicación. Son útiles también para: detectar IPs repetidas.

Otro caso significativo el el nodo con IP 0.0.0.0, la cual no es una dirección válida. No pudimos saber con exactitud por que sucedió pero por lo que investigamos se pude deber a que el dispositivo envió el *request* antes de que se le haya asignado una IP válida.