DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

TP 1: Wiretapping

Teoría de las Comunicaciones
Segundo Cuatrimestre de 2016

Integrante	LU	Correo electrónico
Axel Straminsky	769/11	axelstraminsky@gmail.com
Jorge Quintana	344/11	jorge.quintana.81@gmail.com
Florencia Zanollo	934/11	florenciazanollo@gmail.com
Luis Toffoletti	827/11	luis.toffoletti@gmail.com

Índice

1. Introducción

El objetivo del presente trabajo es descubrir la topología de distintas redes utilizando captura de paquetes ARP. Para la clasificación de los datos así obtenidos se modelaran por cada red dos fuentes de memoria nula, que identificaremos como S y $S1$. Para las fuentes S se distinguirán los paquetes broadcast vs. los paquetes unicast y para las fuentes $S1$ los símbolos se distinguirán basados en las direcciones IP de los orígenes de los paquetes "Who-Has" ARP.

ARP es un protocolo de capa 2.5 que se encarga de traducir direcciones IP (Nivel de red) a direcciones físicas de los dispositivos o "MAC addresses" (Nivel de Enlace). Este protocolo distingue dos tipos de mensajes, "Who has" e "Is At".
"Who has" son típicamente mensajes de "pedido" (request) enviados a toda la red (Broadcast) preguntando a los dispositivos, identificados por MAC address, quién poseé cierta dirección IP.
"Is At" son mensajes de "respuesta" (reply) enviados a un sólo nodo (unicast) que es el nodo que efectuó el pedido, indicando que el dispositivo con la IP buscada se encuentra en la dirección física que envia la respuesta.

Un dispositivo comunicándose en una red a nivel capa de enlace, necesita conocer la dirección MAC del dispositivo con el que desea comunicarse, pero el protocolo IP utiliza direccionamiento por dirección IP. Para traducir de un tipo de direccionamiento al otro de manera eficiente, los dispositivos mantienen una tabla ARP que "cachea" la información. Estas tablas ARP son actualizadas cada cierto tiempo, lo que genera los mensajes de protocolo ARP que capturaremos.

La distinción entre tipos de paquetes que soporta el protocolo ARP nos conduce de forma natural a la primera distinción entre símbolos que utilizaremos para modelar la fuente S, que distinguirá entre paquetes de tipo Broadcast y paquetes de tipo Unicast.

Para el modelado de la fuente $S1$ el criterio de distinción entre los símbolos de la fuente se justifica matemáticamente de acuerdo a la cantidad de información que cada símbolo trae aparejado y su comparación con la entropía total del sistema.
La cantidad de información que aporta un evento E que ocurre con probabilidad $P(E)$ se define como

$$I(E) = log\frac{1}{P(E)}$$

Para calcular la cantidad promedio de información de una fuente de memoria nula S, tenemos que cuando el simbolo s_i ocurre, obtenemos una cantidad de información

$$I(s_i) = log\frac{1}{P(s_i)}$$

y la probabilidad de que esto ocurra es directamente $P(s_i)$ con lo cual la cantidad promedio de información por cada símbolo de la fuente S será

$$H(S) = \sum_S P(s_i)I(s_i)$$

A esta cantidad se la conoce como la entropía de la fuente S: $H(S)$.
La primera conclusión que se puede sacar de la definición es que los eventos que más información aportan son aquellos con menor probabilidad de ocurrir.

Es claro que en nuestro modelo no estamos trabajando con fuentes de memoria nula ideales, sino que estamos utilizando redes reales para modelar las mismas, con lo cual las probabilidades serán en realidad estadísticos obtenidos en base a la experimentación (captura de paquetes). Los estadísticos utilizados serán el ratio entre la cantidad de ocurrencias de un evento y el total de los eventos capturados, por esta razón y con la intención que el estadístico sea representativo de la probabilidad de ocurrencia de los eventos se efectuarán capturas durante intervalos de tiempo mayores a diez minutos.

2. Desarrollo

Para el presente trabajo práctico se desarrolló un programa en Python utilizando la libreria Scapy que captura los paquetes que escucha la interfaz definida y filtra los mismos quedandose solamente con aquellos que son paquetes del protocolo ARP, el mismo programa acepta como parámetros la ruta de un archivo en formato ".pcap" que puede ser generado por medio de capturas anteriores o utilizando software alternativo como Wireshark.

Tenemos dos variantes del programa, uno para cada fuente explicadas más adelante. Con los datos obtenidos por el programa se calculan la entropía y la cantidad de información de cada símbolo en cada fuente. En realidad el programa que modela la fuente "s1" es genérico y permite modelar diversas fuentes de información en base a los parámetros seleccionados en tiempo de ejecución.

Adicionalmente se utilizaron programas auxiliares para generar los gráficos, se crearon archivos dot por medio de scripts en python y luego se graficaron mediante GraphViz.

Se capturaron redes de distintos tamaños utilizando distintas tecnologías a nivel enlance: Switched Ethernet y Wireless LAN

2.1. Fuente S

Esta fuente binaria está compuesta por los símbolos $s_{Broadcast}, s_{Unicast}$ pertenecientes al protocolo ARP. Como sus nombres lo indican, el símbolo $s_{Broadcast}$ es un paquete que está destinado a toda la red (mensaje ARP "Who-has"), mientras que el símbolo $s_{Unicast}$ (mensaje ARP Ïs-at") corresponde al paquete ARP que se envía como respuesta al mensaje "Who-has".

2.2. Fuente S1

Para S1 teníamos varias opciones dentro de los paquetes ARP. Podíamos ver los paquetes Who-Has o Is-At, así como también podíamos centrarnos en source o destino. Es decir, cuatro combinaciones. Para decidirnos por una de ellas lo que hicimos fue experimentar con todas y analizar los resultados. Nos terminamos quedando con las IP *source* de los mensajes *Who-has*, ya que ésta fuente fue la que mejor modelaba en general las redes con las que experimentamos. De todas maneras queremos notar que ninguna de las fuentes es perfecta, y que en algunas redes funcionan mejor otras fuentes (por ejemplo quedandonos con las IPs destino en vez de source), pero por lo dicho anteriormente terminamos eligiendo los paquetes source. En los casos en que funcionaron mejor fuentes alternativas Incluimos un análisis de la red con la fuente elegida, y otro con la fuente alternativa aventurando alguna conclusión.

Como se mencionó anteriormente, al ser el software de naturaleza genérica, para modelar la fuente escogida como s1, las opciones que hay que pasarle al programa son *Who-has* y *Source*.

2.3. Implementación

Para llevar a cabo los experimentos implementamos una herramienta en Python utilizando la librería *scapy*[1], y capturamos los paquetes con *Wireshark*[2].

Nuestro programa comienza preguntando si se desea capturar paquetes Who-has o Is-at, y a su vez si se quiere filtrar por origen o destino. Luego, crea 2 diccionarios: *nodos*, que es un diccionario de la forma $host : cantdeapariciones$, y *connections*, que es de la forma $src : [dst]$ ó $dst : [src]$, según qué modo se elija. Éste último diccionario guarda, para cada key, una lista de todos los nodos con los que se conecta. Esto nos sirve luego para poder visualizar la red como un grafo.

Luego calculamos la entropía y la información que aporta cada nodo. Esto nos sirve para poder dividir los nodos en 2 categorías: **distinguidos** y **no distinguidos**. Los nodos distinguidos los definimos como aquellos cuya información está por debajo de la entropía. Lo que esperamos conseguir con esto es que entre los nodos distinguidos esté el default gateway. Probamos también con otros criterios de corte para considerar un nodo distinguido o no: por ejemplo, en vez de la entropía, tomar el logaritmo de la entropía, la raíz cuadrada de la entropía, dividir la entropía por una constante, etc. Por motivos de espacio no colocamos los resultados para cada uno de estos casos, pero no observamos ninguna mejoría notable con respecto a simplemente tomar la entropía. En algunos casos funcionaba mejor tomar simplemente la

entropía, en otros el logaritmo de la entropía, pero en general funcionó mejor lo primero, y por lo tanto nos terminamos quedando con eso.

Calculado esto, procedemos a realizar distintos gráficos: el primero consiste en un gráfico de la información de cada nodo junto con la entropía máxima y real de la red. Nos quedamos con los 8 nodos con menor información, lo cuál resultó ser una buena heurística para observar tanto los nodos distinguidos como algunos nodos no distinguidos. Para éste gráfico utilizamos la librería *matplotlib*. Otro gráfico que realizamos es un grafo con todos los nodos de la red y sus conexiones. Este lo realizamos con la librería *networkx*.

Las redes que utilizamos para nuestros análisis son: una red corporativa grande, dos redes corporativas pequeñas, y una red hogareña.

A lo largo de las sucesivas experimentaciones notamos que las redes grandes y complejas generaban graficos e información muy difíciles de analizar sin acudir a algún tipo de resumen por lo cual, se agregó una opción para resumir nodos no distinguidos en subredes. Desde el punto de vista de código, se acudió a una simplificación primaria que fue suponer todas las subredes como redes /24 (a pesar de que somos conscientes que existen redes más chicas en la realidad), luego se utilizó la libreria *netaddr* para definir dichas subredes /24 y testear pertenencia de un host a dichas redes y la funcion *cidr_merge* para agrupar subredes /24 contiguas en redes más grandes. La selección entre sumarizar los nodos no distinguidos o no hacerlos, queda a criterio del usuario a quién se le pide que seleccione una opción en tiempo de ejecución.

Luego de implementada esta *sumarización* de redes, notamos que los gráficos aún no eran lo suficientemente representativos de la realidad por lo que acudimos a modificar la generación de los mismos con el objetivo de que visualmente aporten más información. Lo que se implementó fue, distinguir mediante colores aquellos nodos distinguidos (verde) de aquellos que no lo son (azul), y a dibujar cada nodo en escala dependiendo de la cantidad de mensajes que haya generado según el modelo. Cuando los gráficos se generan a partir de los nodos sin sumarizar, el tamaño de cada nodo distinguido es inversamente proporcional a la cantidad de información que aporta; en el caso de los nodos no distinguidos, depende de la cantidad de ejes que posee el nodo (esta cantidad es la suma entre los mensajes que lo tienen como destino y aquellos que los tiene como origen). En el caso de las redes sumarizadas, los nodos distinguidos se comportan de la misma manera, y lo no distinguidos dependen de la cantidad de equipos que estan dentro de la red, esto se hizo para que el tamaño de la *nube* que representa la red sea representativo de la cantidad de equipos activos que la componen.

2.4. Utilización de la herramienta

Para correr el programa, se debe ejecutar el siguiente comando: python s1.py *paquete*, donde *paquete* es una captura en formato .pcap. De no especificarse éste parámetro, se realiza una captura en vivo de la red.

En ambos casos el usuario debe responder a los prompts de la aplicación para indicar cuál es el modelo de fuente que desea utilizar además de las opciones de visualización.

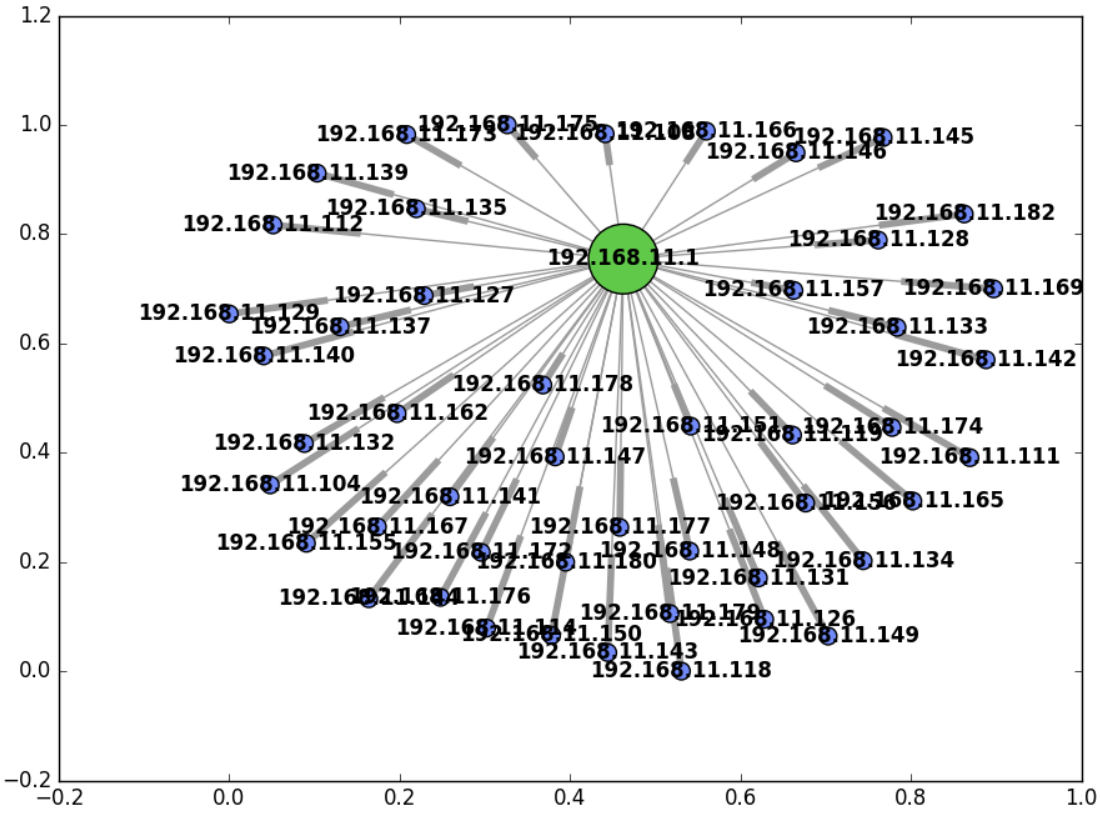
3. Experimentación

3.1. Red corporativa de empresa de Telecomunicaciones

3.2. Red corporativa de empresa de Telecomunicaciones

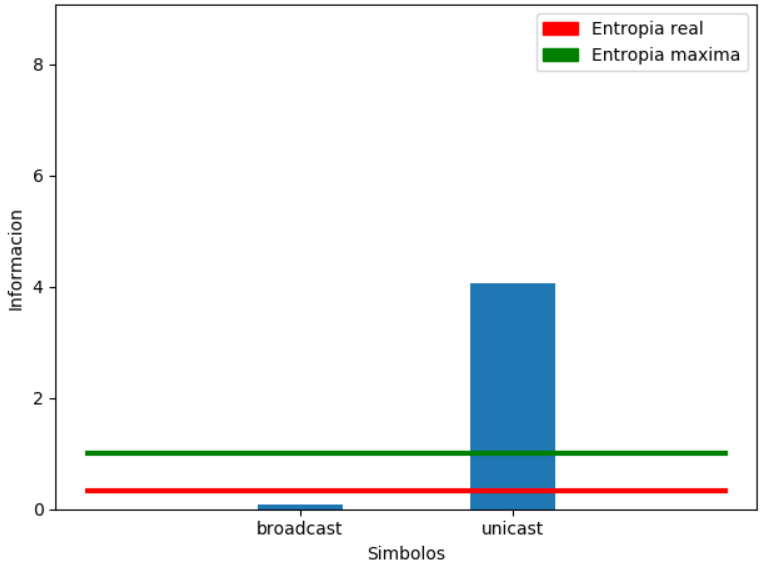
3.3. Red corporativa mediana

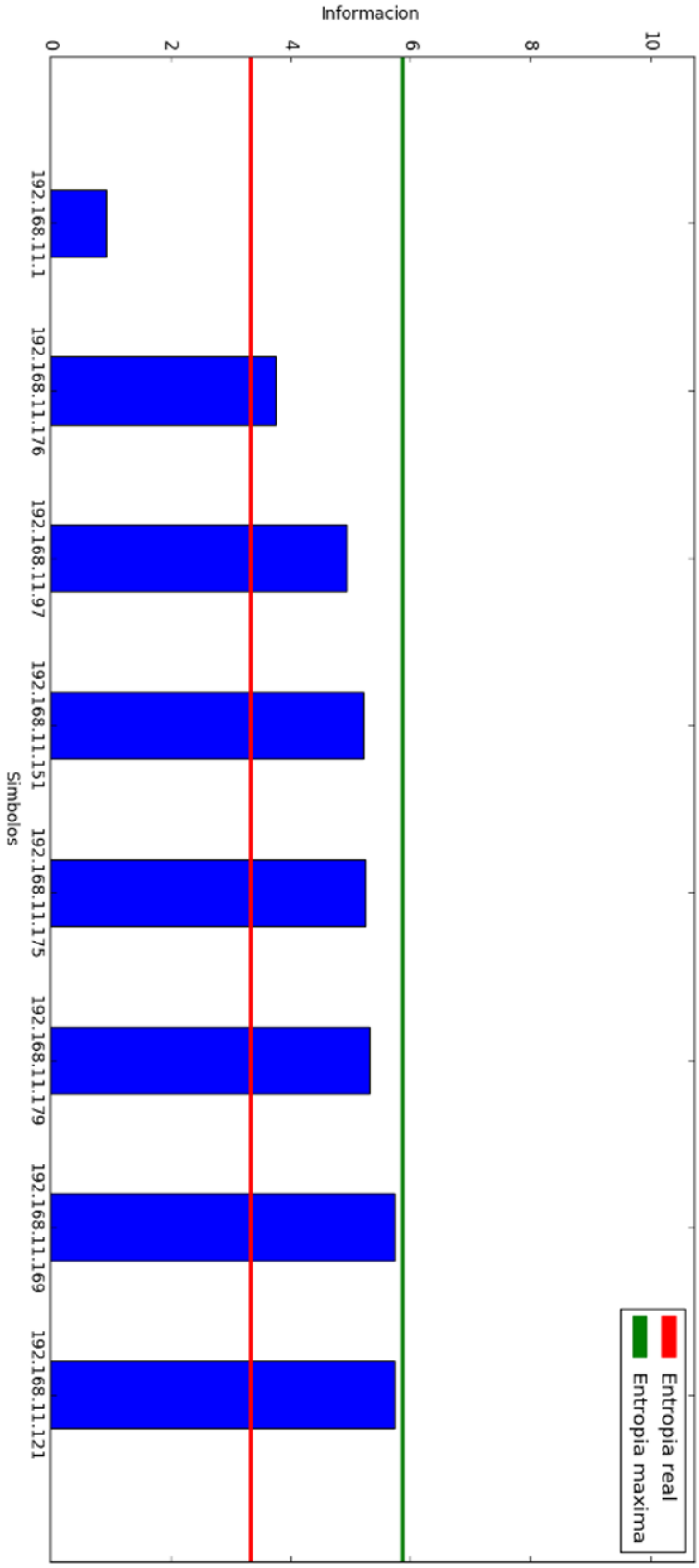
Esta es una red de la cuál conocemos la topología: consiste de un único router al cuál se conectan unas 30 pc's y una cantidad similar de celulares.



Se puede ver que el default gateway (192.168.11.1), aparece como el nodo central al cual todo el resto de los nodos se conectan.

A continuación mostramos la entropía de las fuentes S y $S1$ y la información de cada símbolo:





Tanto para S como para $S1$ podemos observar que la entropía es menor que la entropía máxima. Sabemos que una fuente de memoria nula tiene entropía máxima cuando todos sus símbolos son equiprobables.

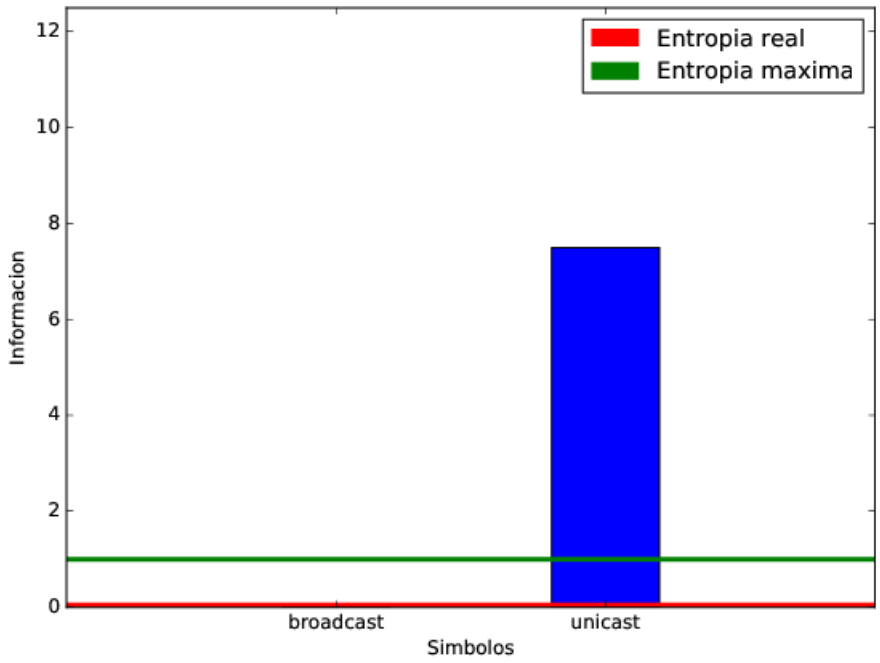
Que la entropía real sea menor nos indica que hay ciertos símbolos más probables que otros.

En el caso de la fuente S, en la captura observamos que el router está constantemente emitiendo paquetes *Who-has*. Por lo tanto, observar el símbolo *Broadcast* es altamente probable, lo cual implica que aporta muy poca información (0.0894 bits), mientras que el símbolo *Unicast* aporta mucha más información (4.0557 bits), haciendo que *Broadcast* sea un nodo distinguido, ya que la entropía de la fuente es 0.3279. La entropía máxima es 10.6995. Creemos que debido a que las tablas de ARP se deben refrescar cada cierto tiempo, la red experimenta un flujo mucho mayor de paquetes *Broadcast* que *Unicast*, dando lugar a una disparidad en la probabilidad de cada símbolo.

En el caso de la fuente $S1$, el único nodo distinguido que nos quedó fue el Default Gateway (pudimos comprobar empíricamente que este nodo lo era ejecutando el comando netstat -nr — grep default), indicando que la manera de seleccionar los símbolos que elegimos para esta fuente funciona muy bien con esta red. Al ser una red con una topología relativamente simple, no podemos asegurar que estos resultados se puedan extrapolar a redes más complejas. Algunos de los otros nodos, como 192.168.11.97 y 192.168.11.151, corresponden a otras notebooks conectadas a la red, lo cual nos resultó un comportamiento extraño. Suponemos que en esas notebooks se estaba ejecutando algún programa que trabaja a bajo nivel y está constantemente obteniendo información de la red local.

3.4. Red corporativa de empresa de Telecomunicaciones - Ethernet switcheada

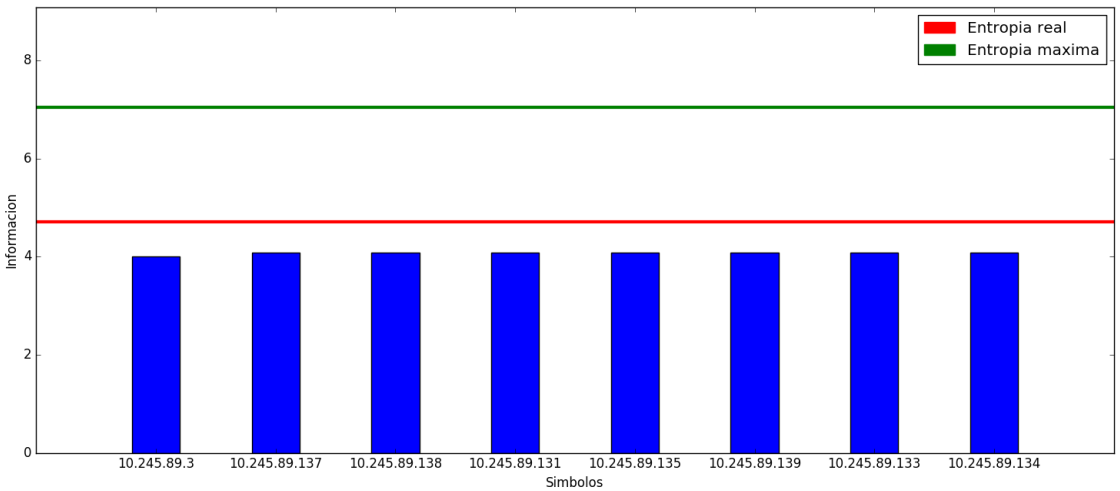
Datos obtenidos con la fuente S:



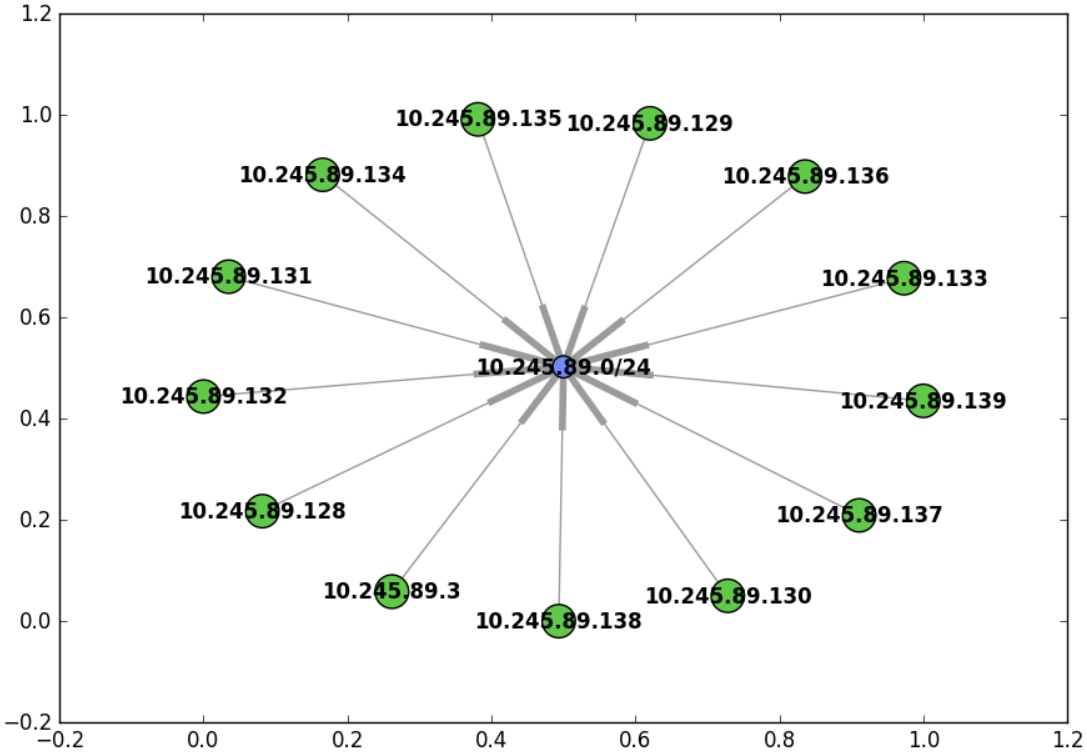
Como se puede observar hay muchísimos más broadcast que unicast, es por ello que el último dá más información. Las probabilidades son las siguientes: broadcast 0.99, unicast 0.01. La entropía es 0.049, casi nula.

Los broadcast son visibles para todos los dispositivos, mientras que los unicast no. Al ser una red switcheada, nosotros estamos viendo los broadcast de todos los hosts y sólo nuestros unicast (respuestas enviadas al host con el cuál se hizo la captura), es por esto que la diferencia es tan grande.

Datos obtenidos con la fuente $S1$:



Observando la disposición de la red descubrimos manualmente que posee dos IP físicas de gateways, 10.245.89.3 y 10.245.89.2 y una IP que actúa como balanceador, 10.245.89.1, a la cuál los hosts realizan sus pedidos y ésta los redirecciona a 10.245.89.3 o 10.245.89.2 según la carga de cada uno.

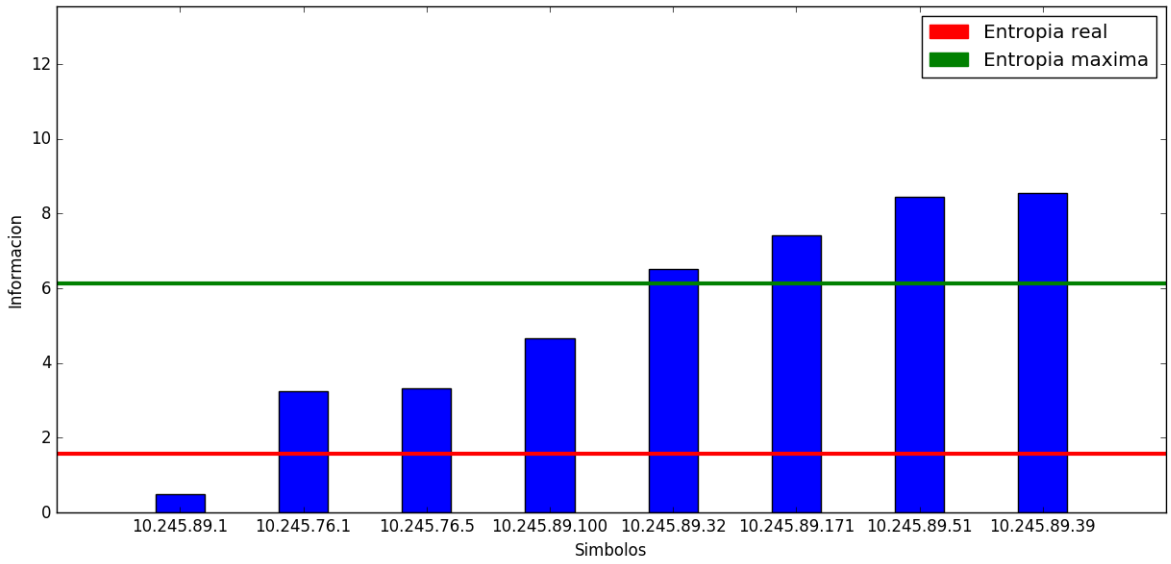


Los nodos distinguidos que vemos en el gráfico son hosts con mucha actividad, los cuáles están consultando a 10.245.89.1. Quien se encuentra incluído en la red ya que recibe pedidos pero las respuestas a estos son emitidas por 10.245.89.3 o 10.245.89.2 entonces no llega a ser distinguido si nuestros símbolos son los IP de origen.

Dentro de estos nodos distinguidos también se encuentra una de los IP físicas de los gateways, 10.245.89.3. Quién queda ocluído por la actividad de los demás equipos.

Como no obtuvimos buenos resultados a la hora de detectar el default gateway, probamos con otra fuente a la cuál llamaremos $S2$, que tiene como símbolos los IP destino de los paquetes ARP Who-Has.

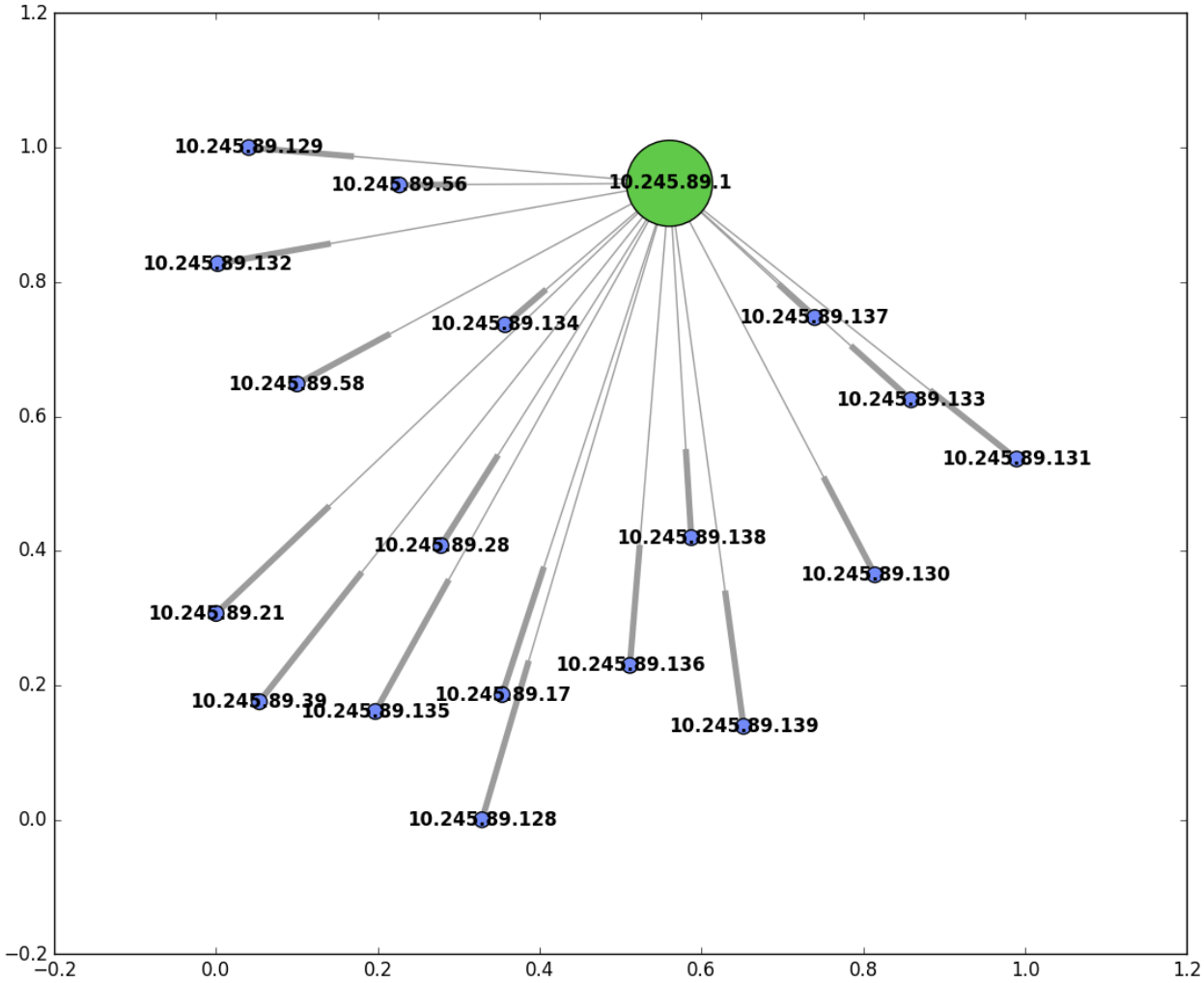
Para esta nueva fuente los datos obtenidos son los siguientes:



En este caso hay un solo nodo distinguido. El cuál, como vimos antes, es el balanceador. Éste es el default gateway para toda la red, quien se encarga de redirigir los pedidos.

Todos los equipos en la Lan preguntan la MAC del default gateway para actualizar sus tablas, lo hacen mediante broadcast a nivel de enlace con lo cual el paquete lo ven todos los nodos en dominio de broadcast, en particular la maquina con la que efectuamos la captura.

El motivo por el cuál funciona mejor la fuente $S2$ que la $S1$ en este caso, suponemos que se debe tanto al tipo de conexión (ethernet switcheada) como a la disposición (i.e. el hecho de que tenga un balanceador distinto de las IP físicas de los gateways).



Quedaría, a futuro, realizar más experimentos en redes cableadas en orden de afirmar la hipótesis de que la fuente $S2$ funciona mejor para éstas.

3.5. Red doméstica - wireless LAN

Esta captura se realizó a partir de una red doméstica, con dos computadoras conectadas por cable ethernet y varios dispositivos wireless (celulares, impresora y chromecast).

Fuente S: La entropía obtenida (aproximadamente 0.924) se acerca bastante a la máxima, a diferencia de las otras capturas que poseen una entropía muy baja. Además como se observa en el gráfico, la proporción de mensajes unicast fue mayor que la de broadcast.

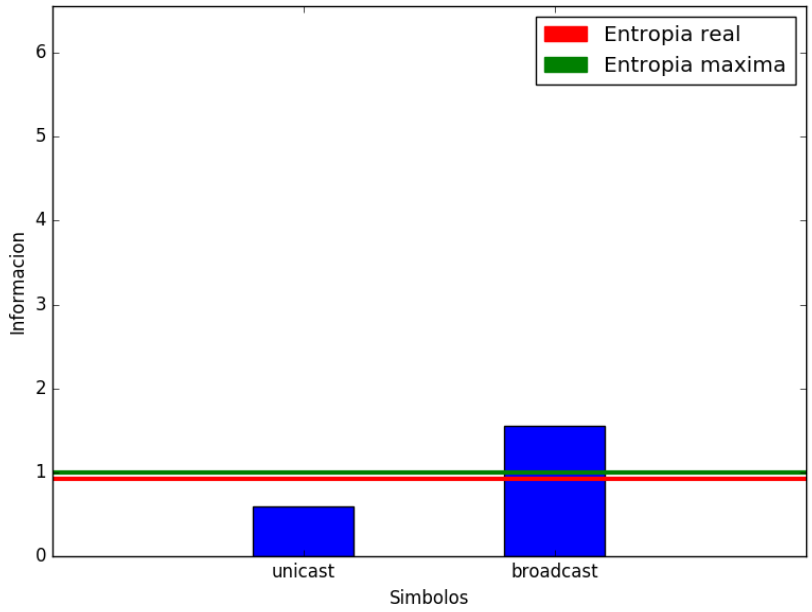


Figura 1: Red doméstica - Fuente S

Todo este comportamiento se debe a que al haber pocos dispositivos dentro de la red y siempre los mismos, las tablas ARP de cada nodo se mantienen más estables (menos probabilidad de anomalías) y los mensajes broadcast se observan con menos frecuencia.

Fuente S1:

1.93284800528

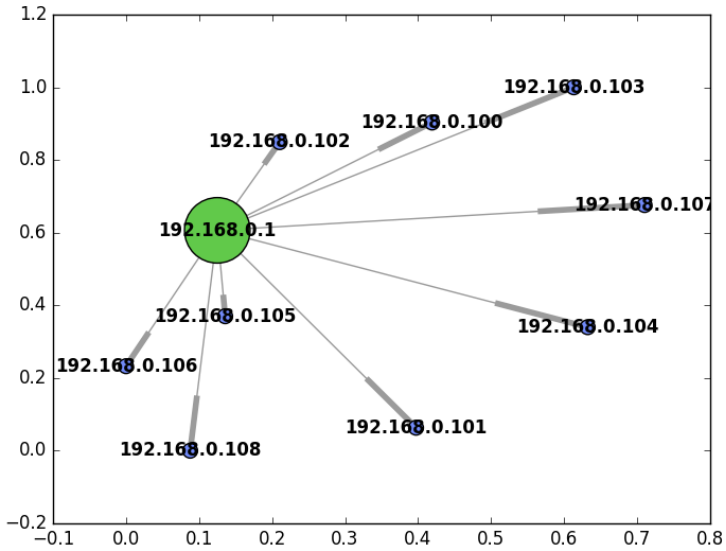


Figura 2:

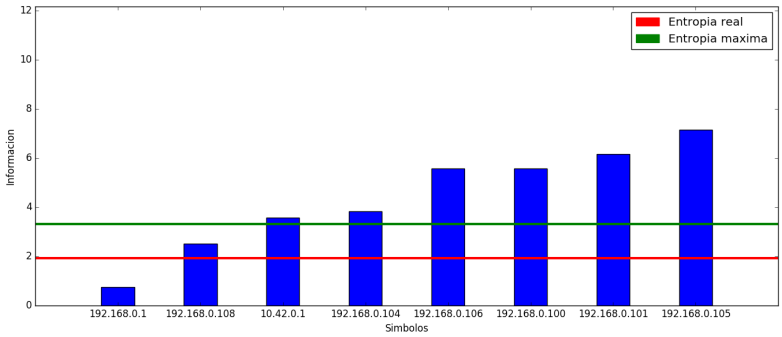


Figura 3:

4. Conclusiones

5. Referencias

1. [1] http://www.secdev.org/projects/scapy/

2 http://www.wireshark.org