Tarea 5 - Monitoreo LAN

Para la tarea se efectuaron capturas distintas para las partes a y b, y para la parte c. De este modo, en el *.zip* se entregan el informe, y los dumps de las capturas, junto con el output del terminal en un archivo *.txt* . Por último, cabe señalar que en la sala se utilizó una red en la que habían switches, un servidor, y la opción para nosotros, los clientes, de conectarse por cable ethernet, o por WIFI. En nuestro caso nos conectamos por WIFI.

Parte A

1. 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_4) AppleWebKit/603.1.30 (KHTML, like Gecko) Version/10.1 Safari/603.1.30'.
   Comentario: Si bien el que se utilizó en realidad fue Safari, como resultado de una batalla empresarial que ocurrió con el surgimiento de los browsers, la identificación es un poco confusa.[1]

2. 'Apache/2.4.18 (Ubuntu)'. Es decir, responde un sistema operativo Ubuntu y un web server Apache en su versión 2.4.18.

Las preguntas 3, 4, 5 y 6 se condensan en la siguiente tabla:

Sitio Acceso	Formato Transferencia	Código HTTP Respuesta	Bytes Retorno Browser (Sin headers)	Cantidad de GETS
http://192.168.1.9/	text/html (2/3) y GIF89a (1/3)	200 (2/3) y 404 (1/3)	8.306	3, para el html y el nyan cat.
http://192.168.1.9/build/slides/5-disk.html	text/html (1/32) y PNG (31/32)	200 (32/32)	2.693.169	32, serie de imágenes PNG y html.
http://192.168.1.9/build/slides/5-disk.html	text/html	200	7833	1, pues las imágenes ya están en caché.
http://192.168.1.9/big.txt	text/plain	200	2.385.723 → 6.488.666	1, para el archivo de texto plano.
http://192.168.1.9/up.html	text/html	200	266 → 425	1, para el html.

[1] http://webaim.org/blog/user-agent-string-history/

7. En el caso de la URL http://192.168.1.9/up.html, se hace además una request de tipo POST, que se envía al responder el formulario. El request de tipo POST tiene como objetivo el envío de datos al servidor para ser procesados, generalmente para generar un cambio en una base de datos, a diferencia de los request tipo GET, cuyo objetivo es solicitar recursos al servidor (html, css, javascript, imágenes, etc).

Parte B

Para las preguntas 1,2,3 y 5 se dispondrá la siguiente tabla:

URL	**Cantidad de segmentos (TCP/HTTP/ ALL)**	**Rangos de segmentos**	**Perdidos/Dañados/Duplicados**	**Throughput (Bytes/s)**	**Pseudo - Throughput (Bytes/s)**
http://192.168.1.9/	18/4/22	1098 - 1299	0/0/0	2125 (10829B en 5.0954s)	561013 (10565B en 0.018832s)
http://192.168.1.9/build/slides/5-disk.html	867/12/879	4454 - 7897	0/0/0	150717 (804kB en 5.3345s)	2726357 (804kB en 0.294899s)
http://192.168.1.9/build/slides/5-disk.html	17/2/19	8413 - 8623	0/0/0	1918.5 (9821B en 5.1191s)	944834 (9557B en 0.010115s)
http://192.168.1.9/big.txt	2188/2/2190	9633 - 11978	0/0/0	452136 (2533kB en 5.6023s)	4364269 (2533kB en 0.580395s)
http://192.168.1.9/up.html	9/2/11	12465 - 12666	0/0/0	334 (1707B en 5.1093s)	143682 (1443B en 0.010043s)

Para encontrar los paquetes se revisó el TCP stream de cada request, y se filtró según ese strem, para luego obtener todos los datos pedidos.

Para el *throughput* en la misma lógica, se utilizó la opción “Conversations” de la sección “Statistics” de Wireshark, filtrando por stream correspondiente a cada request. Así, se obtuvo los bytes enviados de destino a fuente y de fuente a destino, junto con el tiempo empleado, de modo de obtener así el *throughput* final, como el cuociente entre la suma de A a B y de B a A, y el tiempo empleado.

Como análisis extra, se añadió una nueva columna considerando un *pseudo-throughput* que considera el tiempo en recibir los resultados, pero sin esperar al FIN, de modo que es más dependiente de la cantidad de datos y no de la espera para la finalización. Así se ven velocidades de transmisión entre los 140KB/s y los 4.3MB/s

Se revisó la captura en busca de paquetes perdidos pero no los hubo, utilizando los diferentes filtros de wireshark, lo mismo para dañados y/o duplicados.

Para la pregunta 4 se introduce una nueva tabla para responder a lo pedido.

URL	**# Paquetes (SYN/SYN ACK/ACK)**	**Número de secuencia (client/server/client+1)**
http://192.168.1.9/	1098/1099/1011	0/0/1
http://192.168.1.9/build/slides/5-disk.html	4454/4455/4456	0/0/1
http://192.168.1.9/build/slides/5-disk.html	8413/8414/8415	0/0/1
http://192.168.1.9/big.txt	9633/9634/9635	0/0/1
http://192.168.1.9/up.html	12465/12466/12467	0/0/1

Parte C

Para esta parte se dispone una tabla con lo pedido por dirección IP a la que hicimos ping, se filtra utilizando el protocolo ARP para ver los “who has/tell” y alternativamente el ICMP para ver los ping efectuados, ambos según cada IP. Se utilizan los datos de la captura de la parte c , del archivo *partec.pcapng*.

IP	**MAC**	**Fabricante**	**Packet Loss / Average RTT (ms)**
192.168.1.11	08:00:27:c9:e2:99	PcsCompu (PCS Systemtechnik GmbH)	4.5% / 1005.801
192.168.1.143	-	-	100% / -
192.168.1.162	-	-	100% / -
192.168.1.160	-	-	100% / -
192.168.1.114	c8:2a:14:2f:e9:84	Apple (Apple, Inc.)	0% / 3.280
192.168.1.152	-	-	100% / -
192.168.1.141	ac:bc:32:c5:1f:a5	Apple (Apple, Inc.)	0% / 140.512
192.168.1.140	a8:20:66:48:11:82	Apple (Apple, Inc.)	0% / 2.475
192.168.1.142	-	-	100% / -
192.168.1.158	-	-	100% / -

Para los fabricantes se tomó la primera parte de las mac, que decía el fabricante, pero por completitud se añadió el nombre completo al buscarlo en un sitio web de MAC address lookup[2].

Para los casos donde se incorporan guiones es porque se recibió una respuesta de tipo “Request Timeout”.

Se incluye además el porcentaje de pérdida de paquetes junto con el RTT (round trip time) promedio de cada ping exitoso, basado en los ping efectuados.

[2] https://macvendors.com

Parte D

Revisando la captura se observa que hay muchos otros tipos de tráfico, SSDP, MDNS, NBNS, IGMP, DB-LSP-DISC, DHCP, entre otros. Se utiliza la parte a de la captura.

El primero a destacar es MDNS, que corresponde a *multicast-DNS*. Tiene mucho sentido la aparición de tráfico de este tipo porque en su definición, MDNS se trata de determinar la relación IP-hostname en redes locales pequeñas sin “name server”[3], como es el caso de lo recreado en clases.

Para este tipo de tráfico hay 2456 paquetes, correspondientes al protocolo MDNS.

El segundo a destacar es DHCPv6 (*Dynamic Host Configuration Protocol*) el cual se encarga de la asignación de IPs libres a los clientes, de manera de administrar adecuadamente la distribución de los clientes en la red[4]. En este caso es v6, ya que trabaja sobre las IPv6.

Para este tipo de tráfico correspondiente al protocolo DHCPv6 hay 151 paquetes.

El último a destacar es SSDP (*Simple Service Discovery Protocol*) que se encarga de la búsqueda de dispositivos UPnP en una red, de modo de anunciar los servicios de los mismos[5].

Para el último tipo de tráfico, que tiene que ver con el protocolo SSDP hay 1296 paquetes.

[3] https://en.wikipedia.org/wiki/Multicast_DNS
[4] https://es.wikipedia.org/wiki/DHCPv6
[5] https://es.wikipedia.org/wiki/SSDP