Hoja de ejercicios 1
Soluciones

Problemas

Problema 1

1. (HTTP 1.1) Como utilizamos HTTP 1.1, consideramos que el servidor soporta conexiones persistentes. Además, se indica que el servidor soporta hasta tres conexiones simultáneas con un mismo servidor, por lo que los recursos R1 y R3 pueden cargarse simultáneamente, aprovechando la conexión abierta para descargar R0. Los tiempos de descarga de cada recurso son:

   - R0: $t_{R0} = RTT0 + RTT1$
   - R1: La conexión con www.eps.uam.es ya está establecida, por lo que el tiempo es $t_{R1} = RTT2$
   - R2: $t_{R2} = RTT0 + RTT2$
   - R3: La conexión con www.eps.uam.es ya está establecida, por lo que el tiempo es $t_{R3} = RTT2$
   - R4: $t_{R4} = RTT0 + RTT2$

   Por lo que:

$$\begin{aligned} t_{total} &= t_{R0} + \max\{t_{R1}, t_{R2}, t_{R3}, t_{R4}\} \\ &= t_{R0} + \max\{RTT2, RTT0 + RTT2, RTT2, RTT0 + RTT2\} \end{aligned} \tag{1}$$

   Como $RTT0 > RTT2$, queda:

$$t_{total} = RTT0 + RTT1 + RTT0 + RTT2 = 2RTT0 + RTT1 + RTT2$$

2. (HTTP 1.0) En este caso, no existen conexiones persistentes, por lo que cada recurso implica el establecimiento y cierre de una conexión. Sin embargo, todavía las conexiones sí pueden ser simultáneas, por lo que:

3. R0: $t_{R0} = RTT0 + RTT1$
4. R1: Ahora $t_{R1} = RTT0 + RTT2$
5. R2: $t_{R2} = RTT0 + RTT2$
6. R3: $t_{R3} = RTT0 + RTT2$
7. R4: $t_{R4} = RTT0 + RTT2$

$$t_{total} = t_{R0} + \max\{t_{R1}, t_{R2}, t_{R3}, t_{R4}\} = 2RTT0 + RTT1 + RRT2$$

Como vemos, en este caso las conexiones persistentes no hacen que el tiempo de carga sea menor, porque aún tiene que esperar por los recursos externos al dominio base.

8. (Caché local) En este caso, entra en juego la caché local del navegador. Como se nos impone ninguna restricción, consideraremos que no tiene limitaciones de espacio, y que ha podido almacenar todos los recursos solicitados. También se nos dice que los servidores no incluyen cabeceras específicas sobre la duración de los recursos en caché, por lo que una vez encontrados en caché local, el navegador deberá comprobar si siguen siendo válidos con cada servidor, con una cabecera similar a:

   ```
   If-Modified-Since:  Wed, 21 Oct 2018 07:28:00 GMT
   ```

   Como han pasado cinco minutos, consideramos que los recursos no han sido modificados. Cada comprobación implica, por tanto, $RTT0 + RTT3$, por lo que el tiempo total es:

   $$t_{total} = RTT0 + RTT3$$

   (Si R0 no ha sido modificado, no han podido serlo R1 ni R3).

Problema 2

El navegador realizará una petición similar a la siguiente:

```
POST /action_page.php HTTP/1.1
Host: www.disney.com
[...]

firstname=John&lastname=Carrey
```

En una petición real, típicamente se incluirían otras cabeceras, pero no son relevantes para el ejercicio.

Problema 3

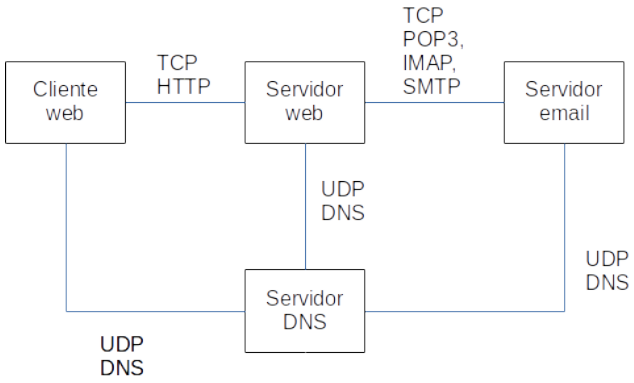
El problema únicamente pretende llamar la atención sobre que tras una cabecera `Set-Cookie` las peticiones posteriores al servidor que la envía incluirán la cabecera `Cookie`.

1. Petición al mismo servidor que estableció la cookie:

```
GET /promotions.html HTTP/1.1
Host: www.example.org
Cookie: yummy_cookie=choco; tasty_cookie=strawberry
```

2. La cookie ha caducado, no se incluye la cabecera `Cookie`.
3. Petición a otro servidor, no se incluye la cabecera `Cookie`.

Problema 4



Problema 5

En este ejercicio basta con aplicar las expresiones que proporcionan los tiempos de distribución de ficheros en entornos cliente/servidor y P2P, respectivamente:

$$t_{cs} = max\left\{\frac{NF}{u_s}, \frac{F}{min(d_i)}\right\}$$

$$t_{p2p} = max\left\{\frac{F}{u_s}, \frac{F}{min(d_i)}, \frac{NF}{u_s + \sum_{i=1}^{n} u_i}\right\}$$

Para el caso CS:

$$
\begin{aligned}
t_{cs} &= max\Big\{\frac{100 \cdot 10 \cdot 10^3 Mb}{10 Mbps}, \frac{10 \cdot 10^3 Mb}{2 Mbps}\Big\} \\
&= max\Big\{10^5 s, 5000 s\Big\} = 10^5 s
\end{aligned}
\tag{2}
$$

En el segundo caso, tenemos que:

$$
\begin{aligned}
t_{p2p} &= max\Big\{\frac{10 \cdot 10^3 Mb}{10 Mbps}, \frac{10 \cdot 10^3 Mb}{2 Mbps}, \frac{100 \cdot 10 Mb \cdot 10^3 Mb}{10 Mb + 50 \cdot 5 Mbps + 50 \cdot 0.5 Mbps}\Big\} \\
&= max\Big\{1000 s, 5000 s, 3508 s\Big\} = 5000 s
\end{aligned}
\tag{3}
$$

Finalmente, se observa que, en este caso, el esquema P2P es 20 veces más rápido que el cliente/servidor.

Problema 6

Los *fingers* proporcionan una especie de "atajos", que hacen el enrutamiento en una red con topología en anillo más eficiente. Para calcular la tabla de *fingers* de un nodo n basta con aplicar la expresión:

$$finger_n[i] = sucesor(n.id + 2^i \mod 2^{160})$$

para $i = 1, .., log(M)$

siendo M el número total de nodos de la red. Por ejemplo, para el índice i=0 tenemos:

$$finger_{12}[0] = sucesor(12 + 2^0) \mod 2^{160} = sucesor(13) = 16$$

O para el índice $i = 4$ se obtiene:

$$finger_{12}[4] = sucesor(12{+}2^4) \mod 2^{160} = sucesor(12{+}16) = sucesor(28) = 30$$

Iterando para los diferentes índices, obtenemos la tabla:

Índice	**Nodo**
N12 + 2^0	16
N12 + 2^1	16
N12 + 2^2	16
N12 + 2^3	22
N12 + 2^4	30
N12 + 2^5	45

Problema 7

Para calcular los identificadores de nodos y contenidos basta con calcular el hash SHA1 de cada cadena asociada a ellos, direcciones IP y valor, respectivamente.

Los resultados y distribución de contenidos queda finalmente:

Dirección IP	**Valor SHA1**	**ID**
138.100.10.100	2ced31c12cfdf5ba1597f22bd740ca44beccf8c1	c1
150.244.55.43	a686977ee7f28360a7a112ac5a29394d16e7309f	9f
138.100.50.5	698441857971460088da3982e9402a2cd0f8e394	94
175.55.28.21	0f157b42033724817c5597876216e614cd1da2e1	e1
Contenido	**Valor SHA1**	**ID**
Los pachachos	aff146329f51be780addb13d8d6ddbc029243708	08
Redes2	99f61025ca9c3b3fc7aef84d4bc59eacf0b953bb	bb
Los monologo	3109ef715e4ae469cccbae7f0afa4034395bf779	79

ID Nodo	**Nodo anterior**	**Nodo posterior**	**Contenido**
94	e1	9f	K08, K79
9f	94	c1	-
c1	9f	e1	Kbb
e1	c1	94	-

Problema 8

1. **Falso**. HTTP es un protocolo de petición/respuesta, por lo que el cliente envía cuatro peticiones y recibe cuatro respuestas. La única diferencia es que si se utilizase conexiones no persistentes, todas las peticiones y respuestas se harían por cuatro conexiones independientes. Por el contrario, si se utilizasen conexiones persistentes (por defecto en HTTP/1.1), se utilizaría una única conexión que "encapsularía" todas las peticiones y respuestas.

2. **Cierto**. Lo importante aquí es saber que las conexiones, persistentes o no, se establecen entre hosts, y no entre páginas. Por tanto, aunque sean páginas diferentes, residen en el mismo servidor y pueden recibirse con una única conexión.

3. **Falso**. Cada conexión HTTP no persistente inicia una nueva conexión a nivel TCP, por lo que no es posible que un único segmento TCP contenga dos cabeceras HTTP (podría darse el caso, aunque es poco probable, si se utilizasen conexiones persistentes).

4. **Falso**. La cabecera **Date** solo contiene la fecha de respuesta actual, no hace referencia al momento de última modificación. Este hecho se señaliza con la cabecera **Last-Modified**.

5. **Falso**. En algunos casos, como cuando el recurso solicitado no existe, el servidor responde con un código de error 404 y un cuerpo vacío.

Problema 9

Intervienen los protocolos HTTP y DNS, ambos en capa de aplicación. Obviamente, también los protocolos TCP e IP, a nivel de transporte y red, respectivamente.

Problema 10

1. La URL completa sería http://gaia.cs.umass.edu/cs453/index.html.
2. HTTP/1.1
3. Persistente, porque la cabecera *Connection* tiene el valor *keep-alive*.
4. Es imposible de determinar a partir de la información proporcionada. Esa información se gestiona no a nivel de aplicación, sino de red (*socket*) por el SO.

Problema 11

1. (**Un solo objeto**) El tiempo total para resolver la dirección IP es simplemente la suma de los tiempos de los n servidores DNS necesarios. Una vez conocida, es necesario RTT_0 para establecer la conexión TCP con el servidor, y otro RTT_0 para solicitar y recibir (aunque el tiempo 'de vuelta' es despreciable) el objeto. Por tanto, el tiempo total de respuesta es:

$$2RTT_0 + \sum_{i=1}^{n} RTT_i$$

2. (**HTTP no persistente, sin pipeline**)

$$\sum_{i=1}^{n} RTT_i + 8(\underbrace{RTT_0}_{TCP} + \underbrace{RTT_0}_{HTTP}) = 16RTT_0 + \sum_{i=1}^{n} RTT_i$$

3. (**HTTP no persistente, 5 conexiones**)

$$\sum_{i=1}^{n} RTT_i + \overbrace{\underbrace{RTT_0}_{TCP} + \underbrace{RTT_0}_{HTTP}}^{\text{5 primeros objetos}} + \overbrace{\underbrace{RTT_0}_{TCP} + \underbrace{RTT_0}_{HTTP}}^{\text{3 últimos}} = 4RTT_0 + \sum_{i=1}^{n} RTT_i$$

4. (**HTTP persistente, 5 conexiones**)

$$\sum_{i=1}^{n} RTT_i + \overbrace{\underbrace{RTT_0}_{TCP} + \underbrace{RTT_0}_{HTTP}}^{\text{5 primeros objetos}} + \overbrace{\underbrace{RTT_0}_{HTTP}}^{\text{3 últimos}} = 3RTT_0 + \sum_{i=1}^{n} RTT_i$$

Problema 12

1. El tiempo de transmisión total en una infraestructura como esta es:

$$t_{TOTAL} = t_{INET} + t_{ACCESO} + t_{LAN}$$

Nos dicen que el tiempo de transmisión en la LAN es despreciable, por lo que el último sumando es cero.

Por otro lado, sabemos que $t_{INET} = 2$ segundos. Para obtener el retardo en el enlace de acceso, comencemos por calcular el tiempo necesario para la transmisión de un petición, Δ_{acc}, que será claramente S/R, donde S es el tamaño medio de la petición, y R el ancho de banda del enlace.

Por tanto:

$$\Delta_{acc} = S/R = \frac{900kbits}{1500kbits/sec} = 0.6\ sec$$

Solo queda sustituir los valores en la expresión que se nos da para calcular t_{ACCESO}:

$$t_{ACCESO} = \frac{\Delta_{acc}}{(1 - \Delta_{acc} \cdot r)} = \frac{0.6\ sec}{1 - 0.6 \cdot 1,5} = 6\ sec$$

Finalmente queda:

$$t_{TOTAL} = t_{INET} + t_{ACCESO} + t_{LAN} = 2\ sec + 6\ sec + 0\ sec = 8\ sec$$

2. Puesto que ahora el 40% de las peticiones son satisfechas directamente por la caché, y no saldrán a Internet, el tráfico en el enlace de acceso se verá reducido en ese mismo porcentaje. Así, de las 1,5 peticiones/sec anteriores, pasaremos a tener $1,5 \cdot 0,6 = 0,9$ peticiones/sec.

Por tanto, el nuevo tiempo de acceso anterior será (0,6 sec)/[1 – (0,6)(0,9)] = 1,3 sec. Y el tiempo de respuesta medio:

$$t_{TOTAL} = 0,6(2\ sec + 1,3sec) + 0,4(0\ sec) = 1,98\ sec$$

Este tiempo se ha reducido, por tanto, de 8 segundos a 1,98.