Open Questions: Namy

a) What happens if a client repeats the same query on experiment MS1?
La primera vez que se hace la consulta se marcan las resoluciones como unknown, y se consulta a los servidores.

Al repetir una query el cliente tendrá que volver a hacer todas las consultas, **volverá a hacer una resolución iterativa** consultando los "servidores" desde la raíz hasta el host, indicando que son *invalid*, eso es porque las entradas en caché tienen un TTL de cero segundos [TTL = 0], por lo tanto las resoluciones de nombres caducan al instante y la entrada queda como inválida.

Esto resulta en un nuevo conjunto de consultas a los servidores para cada petición repetida.

b) What is the observed behavior on experiment MS2? Justify why.

MS2 implica resolver un nombre de host, cerrarlo enviando un mensaje de parada, y luego resolver su nombre de nuevo. Se observa que el resolver todavía intenta retornar el PID del proceso host terminado, pero su pong resulta en "no reply from host".
Es decir:

- Se apaga el host (se le envía stop), pero antes de modificar el código de servidor/host no se envía el mensaje ***deregister*** al servidor padre.
- Así que el servidor sigue manteniendo en su lista la entrada del host, entonces al volver a consultar, el resolver devuelve ese OldPid, que se trata del mismo proceso ya disfuncional.
- El cliente envía ping a OldPid, pero como ese proceso ya no existe, no recibe pong (timeout).

Por tanto la falta de limpieza de la entrada en el servidor deja un identificador obsoleto que el resolver sigue devolviendo.

c) What is now the observed behavior on experiment MS2?

Comportamiento visto: Si se consulta el host después de apagarlo, el resultado es "unknown host" porque el host se da de baja del servidor al apagarse.

Ahora si se desregistra el host/server en el servidor padre. El mismo host de desregistra el mismo cuando termina. Por tanto ahora sí se elimina esa entrada de la lista del padre.

Cuando el cliente envíe un ping, esta vez no se va a encontrar el subdominio indicado (porque se ha desregistrado) y por lo tanto no se tratará de hacer el ping porque directamente la resolución de nombres va a detectar que el host/servidor no existe. Es decir esta vez no tendremos en el servidor padre un Pid inválido guardado.

d) What is the observed behavior on experiment MS3? Justify why.

El experimento MS3 establece un TTL largo y “mueve” un host: consulta su nombre, apagarlo, reiniciarlo con el mismo nombre y vuelve a consultar.

Comportamiento visto: El cliente recibe el *oldPid* de la caché y recibe “no reply from host” cuando se hace ping, debido a que el host se reinicia con un nuevo *Pid*.

Ahora al añadir un TTL la entrada se guarda durante ese tiempo en la caché. Por tanto si se vuelve a repetir una consulta no será necesario hacer todas las consultas y que en la caché tenemos el Pid del host en concreto al que se hace el ping. Este nos devolverá el pong al hacerle ping. Cuando el TTL se acabe sí que se tendrán que repetir las consultas.

Al hacer el MS3 , surge un problema. El problema es que guardamos en la caché el resultado de una primera consulta. Guardando su Pid. Pero si “movemos” el host ahora su Pid es distinto, haciendo así que un ping al Pid de la caché no sea exitoso. Esto es porque en ningún momento informamos al cliente sobre el “movimiento”.

e) Which nodes have been informed about the host movement?

El movimiento del host sólo se ha notificado al servidor en el qué está registrado el host.

Los mensajes de register y deregister se envian solo al servidor padre (el responsable del dominio). Ninguno de los otros servers ni el root lo reciben.

El *resolver* sólo se entera indirectamente al expirar la entrada cacheada. Es decir, el *resolver* podrá resolver el nombre del host en su nueva ubicación una vez que la entrada caché del host se expire.

f) When will the client find the host correctly in the new location?

El cliente encontrará correctamente al host en su nueva ubicación una vez que haya expirado el TTL, desde la query inicial, cuando la entrada cacheada en el resolver expira y el resolver obtiene el nuevo PID.
Mientras no haya expirado seguirá devolviendo la información antigua. Cuando expire volverá a hacer la resolución completa, obteniendo así la información actualizada.

g) Derive a theoretical quantification of the amount of messages needed for name resolution without and with the cache (assume a resolver that repeats the same query about a host at depth D in the namespace every F seconds for a total duration of R seconds, being the TTL equal to T seconds).

Razonamiento:

- Resolver cache resultados con TTL T.
- Cada resolución actualiza cache con Expire = Now + T
- Las peticiones usan la caché si son válidas; sino, se produce una nueva resolución.
- Las resoluciones ocurren cuando query time t=kF satisface t >= Expire
- Primera resolución t = 0, Expire = T
- **Período entre resoluciones:** Siguiente resolución t=kF >= T => t = **P = [T/F] *F**
- **Número de resoluciones: K = R/P + 1**
- **Mensajes por resolución: 2*D**

Número de mensajes con caching:

*2*D * (1 + R div (F * (1 + T div F)))* = $2 * D * (1 + \frac{R}{F * (1 + \frac{T}{F})})$

Número de mensajes sin caching:
*2*D * (1 + R div F)*

Mirar sol-ex-final-B-22.pdf

h) What happens if two clients perform the same query on experiment MS5?

El experimento MS5 repite el MS1 (clientes concurrentes consultando un espacio de nombres) utilizando resolución recursiva con caché activado.

- **Primer cliente:** El resolver resuelve la petición, almacena en caché {host, Pid} con Expire = Now + TTL, y lo devuelve.
- **Segundo cliente:** Consulta el mismo nombre poco después. El resolutor encuentra una entrada válida en la caché (Now < Expire) y devuelve {host, Pid} sin consultar a los servidores.

La consulta del segundo cliente se responde desde la caché si el TTL no ha caducado, lo que evita consultas adicionales al servidor y proporciona una respuesta más rápida.

i) Which resolution can exploit caching better? Justify why.

La resolución puede explotar el almacenamiento en caché en ambas resoluciones, pero la resolución recursiva puede explotar mejor el almacenamiento en caché porque, además, permite reutilizar resultados parciales y completos de consultas anteriores de otros clientes, ya que los servidores también pueden almacenar información en caché.

- **Resolución iterativa:** El resolver consulta cada nivel de servidor (por ejemplo, raíz, “edu”, “upc”) y almacena en caché los resultados intermedios (por ejemplo, [“edu”], [“upc”, “edu”]) y el resultado final ([“www”, “upc”, “edu”]). Las consultas posteriores de nombres relacionados (por ejemplo, [“ftp”, “upc”, “edu”]) reutilizan los servidores intermedios almacenados en caché, lo que reduce los mensajes.

- **Resolución recursiva:** El resolver envía una consulta a la raíz, que resuelve recursivamente el nombre y devuelve la respuesta final. El resolver sólo almacena en caché el resultado final (por ejemplo, [“www”, “upc”, “edu”]). Las consultas de nombres relacionados requieren nuevas resoluciones recursivas.