

Seminar Report: Namy

May 15, 2019

Carles Tornel
Jesus Alfaro
Ricard Abril

1 Introduction

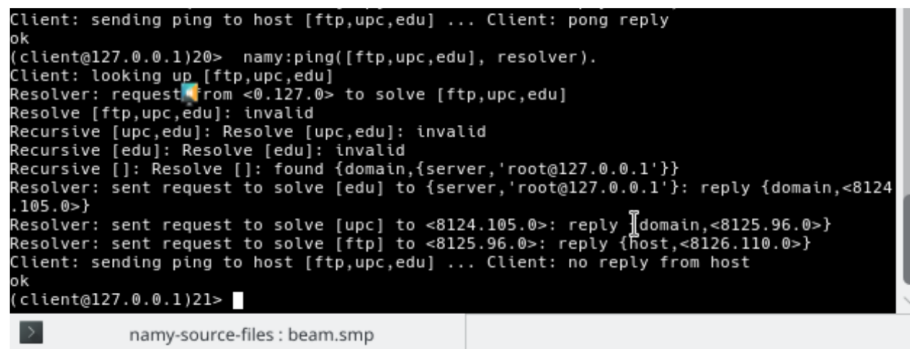
En aquesta pràctica haurem d'implementar un servidor de noms distribuïts on s'emmagatzemaran els identificadors dels processos hosts. S'utilitzarà el sistema de cache, però, com el TTL per defecte és 0, no hi haurà intervenció de la cache a la primera versió, a la segona versió demana modificar aquest valor per tal de fer servir la cache.

2 Experiments

2.1 Testing

2.1.1 Make experiments with several clients asking for name resolution concurrently

En aquest primer experiment hem iniciat els diferents nodes tot seguint les indicacions de la documentació, després hem iniciat la resolució de www, upc i edu en diversos clients de forma concurrent tal com podem veure a la imatge següent:



```
Client: sending ping to host [ftp,upc,edu] ... Client: pong reply
ok
(client@127.0.0.1)20> namy:ping([ftp,upc,edu], resolver).
Client: looking up [ftp,upc,edu]
Resolver: request from <0.127.0> to solve [ftp,upc,edu]
Resolve [ftp,upc,edu]: invalid
Recursive [upc,edu]: Resolve [upc,edu]: invalid
Recursive [edu]: Resolve [edu]: invalid
Recursive []: Resolve []: found {domain,{server,'root@127.0.0.1'}}
Resolver: sent request to solve [edu] to {server,'root@127.0.0.1'}: reply {domain,<8124
.105.0>}
Resolver: sent request to solve [upc] to <8124.105.0>: reply {domain,<8125.96.0>}
Resolver: sent request to solve [ftp] to <8125.96.0>: reply {host,<8126.110.0>}
Client: sending ping to host [ftp,upc,edu] ... Client: no reply from host
ok
(client@127.0.0.1)21>
```

The screenshot shows a terminal window with a dark background. The text is white and shows the output of a program named 'namy'. The user enters a command to ping three hosts (ftp, upc, edu) using a resolver. The output shows the resolver's internal state, including requests to solve, recursive resolution attempts, and the final replies with IP addresses. The terminal window has a title bar that says 'namy-source-files: beam.smp'.

2.1.2 Shut down a host (by sending a stop message) and try to solve its name again.

Simplement, la petició no podrà ser atesa, de forma que saltara el timeout i obtindrem un missatge d'error.

Amb aquesta modificació, quan eliminem un host, aquest és eliminat en el seu pare, de forma que quan intentem resoldre'l, en canvi d'obtenir un error a causa del timeout rebem la resposta del pare dient que no pot trobar el host en qüestió, tal com podem veure en la següent imatge:

2.2 Using the cache

Si assignem un TTL més alt, les resolucions tardaran més a ser esborrades de la cache, de forma que si movem el host i intentem fer una petició ha aquest abans d'haver resolt de nou, no obtindrem cap resposta, ja que aquell host ja no es troba en l'adreça a la qual estem fent ping.

2.2.2 Derive a theoretical quantification of the amount of messages needed for name resolution without and with the cache and check experimentally that your formulation is correct (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).

Per realitzar aquest experiment em assignat un TTL de 600 (en el cas que utilitzem la cache) i un TTL de 0 en el cas que no volem utilitzar la cache.

El número de pings obtinguts amb cache és de 32305, en canvi quan no utilitzem la cache el número de pings és de 15437.

Sense utilitzar la cache podem veure que obtenim menys pings, però generarà un tràfic molt major a la xarxa, en el cas que utilitzem cache, només ens caldran 32310 missatges, per a fer 32304 pings, però en el cas de no utilitzar la cache ens caldran 123496 missatges per poder fer 15437 pings. En la imatge següent observem les diferències en l'execució d'ambdós "models".

```

[ok, host],
[ok, resolver],
[ok, entry],
[ok, name],
[ok, cache]
(cclient@127.0.0.1)> resolver:start((server, 'root@127.0.0.1')).
true
(cclient@127.0.0.1)> nmap:ping([www.upc.edu], resolver).
Client: looking up [www.upc.edu]
Resolver: request from <8.67.8> to solve [www.upc.edu]
Resolve [www.upc.edu]: unknown
Resolve [upc.edu]: unknown
Resolve [edu]: found domain (server, 'root@127.0.0.1')
Resolver: sent request to solve [www.upc.edu] to (server, 'root@127.0.0.1'): reply [www.upc.edu] {host,<18673.187.8>->- ALL [[edu], {domain,<18671.187.8>->->,0}, {upc.edu], {domain,<18672.187.8>->->,0}, {www.upc.edu], {host,<18673.187.8>->->,0}}
Client: sending ping to host [www.upc.edu] ... Client: pong reply
ok
(cclient@127.0.0.1)> resolver:changeTTL((server, 'root@127.0.0.1'), 600).
(ttl,600)
(cclient@127.0.0.1)> nmap:ping([www.upc.edu], resolver).
Client: looking up [www.upc.edu]
Resolver: request from <8.67.8> to solve [www.upc.edu]
Resolve [www.upc.edu]: invalid
Resolve [upc.edu]: invalid
Resolve [edu]: invalid
Resolve [edu]: found domain (server, 'root@127.0.0.1')
Resolver: sent request to solve [www.upc.edu] to (server, 'root@127.0.0.1'): reply [www.upc.edu] {host,<18673.187.8>->->- ALL [[edu], {domain,<18671.187.8>->->,600}, {upc.edu], {domain,<18672.187.8>->->,0}, {www.upc.edu], {host,<18673.187.8>->->,0}}
Client: sending ping to host [www.upc.edu] ... Client: pong reply
ok
(cclient@127.0.0.1)>

```

```

recursive: beam.smp
recursive: beam.smp -- Konsole
File Edit View Bookmarks Settings Help
Server: received request to solve [www.upc]
Server: resolve upc: known domain
Server: resolve [www.upc]: unknown in cache
Server: sent request to solve [www] to <18668.187.8>
Server: resolve [www.upc]: done: replies [[upc], {domain,<18668.187.8>->->,0}, {www.upc], {host,<18678.187.8>->->,0}}
Server: received request to solve [www.upc]
Server: resolve upc: known domain
Server: resolve [www.upc]: invalid in cache
Server: sent request to solve [www] to <18668.187.8>
Server: resolve [www.upc]: done: replies [[upc], {domain,<18668.187.8>->->,0}, {www.upc], {host,<18678.187.8>->->,0}}
(cclient@127.0.0.1)>

```

```

recursive: beam.smp -- Konsole
File Edit View Bookmarks Settings Help
Server: received request to solve [www.upc.edu]
Server: resolve edu: known domain
Server: resolve [www.upc.edu]: unknown in cache
Server: resolve [upc.edu]: unknown in cache
Server: sent request to solve [www.upc] to <18666.187.8>
Server: resolve [www.upc.edu]: done: replies [[edu], {domain,<18666.187.8>->->,0}, {upc.edu], {host,<18669.187.8>->->,0}}
Server: received request to solve [www.upc.edu]
Server: resolve [www.upc.edu]: invalid in cache
Server: resolve [upc.edu]: invalid in cache
Server: sent request to solve [www.upc] to <18666.187.8>
Server: resolve [www.upc.edu]: done: replies [[edu], {domain,<18666.187.8>->->,600}, {upc.edu], {host,<18669.187.8>->->,0}}
(cclient@127.0.0.1)>

```

```

recursive: beam.smp -- Konsole
File Edit View Bookmarks Settings Help
[ok, cache],
[ok, entry],
[ok, host],
[ok, name]
(host@127.0.0.1)> host:start([www, www, (server, 'upc@127.0.0.1')]).
true
Host: create domain www at (server, 'upc@127.0.0.1')
Host: Ping from <18678.67.8>
Host: Ping from <18678.67.8>
(host@127.0.0.1)>

```

2.3 Recursive resolution

2.3.1 Set up the naming system by using the new versions of the resolver and the server that implement recursive resolution. Perform experiments to compare this version with the former one using iterative resolution. Focus especially on how caching performs on each one when using several clients.

En aquest experiment, hem assignat un TTL de 0 per la implementació recursiva i per l'implementació iterativa, després hem fet el mateix amb un TTL de 3.

Per provar els diferents models, hem posat el sistema en funcionament durant mig minut aproximadament.

Pel model recursiu amb TTL 0, hem obtingut 8862 pings.

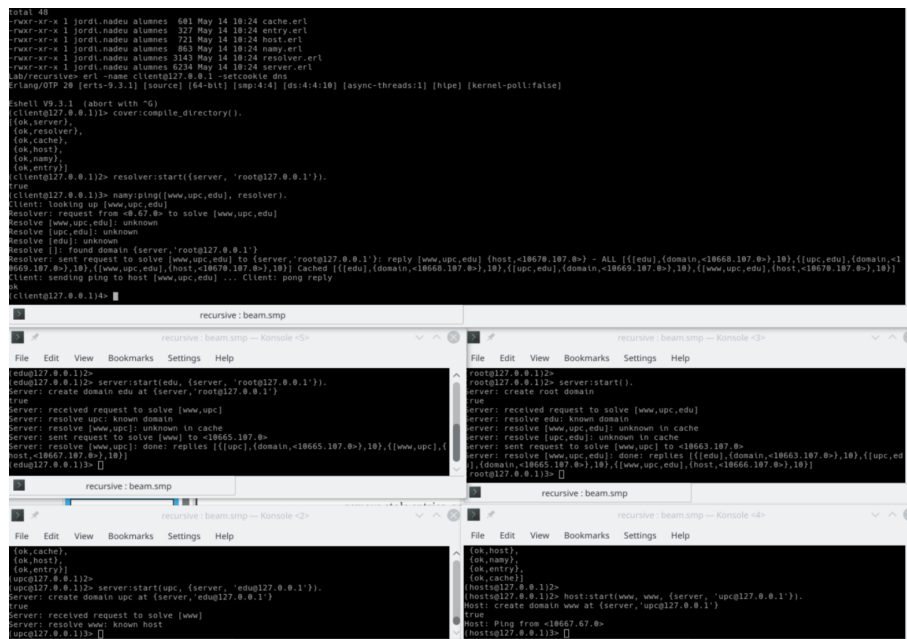
Pel model iteratiu amb TTL 0, hem obtingut 12891 pings.

En canvi per el model amb TTL 3, hem obtingut:

Pel model recursiu 49589.

Pel model iteratiu 50120.

Per tant podem concloure que la versió iterativa serà més ràpida. Tal com podem veure en la imatge següent:



```
total 40
-rwxr-xr-x 1 jordi.nadou alumnas 601 May 14 10:24 cache.erl
-rwxr-xr-x 1 jordi.nadou alumnas 227 May 14 10:24 entry.erl
-rwxr-xr-x 1 jordi.nadou alumnas 721 May 14 10:24 host.erl
-rwxr-xr-x 1 jordi.nadou alumnas 463 May 14 10:24 nam.erl
-rwxr-xr-x 1 jordi.nadou alumnas 3143 May 14 10:24 resolver.erl
-rwxr-xr-x 1 jordi.nadou alumnas 6234 May 14 10:24 server.erl
lsbrrecursive.erl -name client@127.0.0.1 -setcookie dos
 Erlang/OTP 20 [erts-9.3.1] [source] [64-bit] [smp:4:4] [ds:4:4:10] [async-threads:1] [hipe] [kernel-poll:false]

Eshell V9.3.1 (abort with ^C)
client@127.0.0.1> cover:compile_directory().
{ok,server},
{ok,resolver},
{ok,cache},
{ok,host},
{ok,nam},
{ok,entry}}
client@127.0.0.12> resolver:start(server, 'root@127.0.0.1').
true
client@127.0.0.13> nam:ping(www.upc.edu, resolver).
client: looking up www.upc.edu
Resolver: request from <0.67.0> to solve www.upc.edu
Resolve www.upc.edu: unknown
Resolve upc.edu: unknown
Resolve edu: unknown
Resolve []: found domain (server, 'root@127.0.0.1')
Resolver: sent request to solve www.upc.edu to (server, 'root@127.0.0.1'): reply www.upc.edu {host,<18678.187.0>,10} - All [{edu},{domain,<18668.187.0>,10},{upc.edu},{domain,<18668.187.0>,10},{www.upc.edu},{host,<18678.187.0>,10}]
Client: sending ping to host www.upc.edu ... Client: pong reply
ok
client@127.0.0.14>
```

3 Open Questions

4 Testing

4.0.1 What is the observed behavior?

El comportament després de tancar el host i tornar a resoldre el seu nom, serà el mateix que quan es fa per primera vegada la resolució, ja que les entrades de cache tenen TTL 0, i per tant no estarà disponible per al client.

4.0.2 What is different now?

Using the cache

4.0.3 What is the observed behavior?

Una vegada el resolver rep una resposta dels servidors, la guarda a una entrada de cache amb el TTL corresponent, en segons, i quan torni a rebre una petició d'un client, en l'interval indicat pels segons, podrà saltar-se la resolució i retornar el resultat de l'entrada de cache corresponent.

4.0.4 What happened with cached information in the resolver? Which nodes need to be notified about the host movement?

La informació de cache estarà disponible per als clients segons el temps TTL ho indiqui. Com el resolver fa les peticions iterativament, la càrrega de treball es major, i per tant, tardarà més temps en satisfer les peticions dels clients. A més, si el TTL és petit pot resultar poc eficient perquè el resolver haurà de tornar a fer tot el treball una vegada més, i pot no haver gaire diferencia amb una cache de 0 TTL.

Els nodes que hauran de ser notificats per un moviment de hosts son els que pertanyen al administrational i managerial layer, i que estiguin a un nivell superior al host desplaçat, els pares del host.

4.1 Recursive Resolution

4.1.1 Which version can exploit caching better? Justify why.

L'escenari on ens trobem determinarà si una solució és millor que una altra.

El temps de càlcul d'inserció i d'extracció es capaç de ralentitzar una petició qualsevol, fins i tot quan les llistes no han presentat més de 3 elements, segons s'ha pogut observar a la realització dels experiments.

És més favorable utilitzar la sol·lució recursiva en un escenari on el Time To Live és considerablement gran i hi han moltes peticions de clients per un mateix host, però en el nostre cas (amb un TTL menor i molts hosts) és millor utilitzar la sol·lució iterativa.

5 Personal opinion

Aquest laboratori ha ajudat a clarificar un dels conceptes més comuns dels sistemes distribuïts en xarxa: La comunicació multicast. Mitjançant la pràctica de l'implementació de nodes com poden ser els Workers, Leaders i Slaves, es té una visió més exemplificadora del problema que s'ha tractat.