

# Seminar Report: Muty

Marc Guinovart, Carles Salvador, Pol Valls

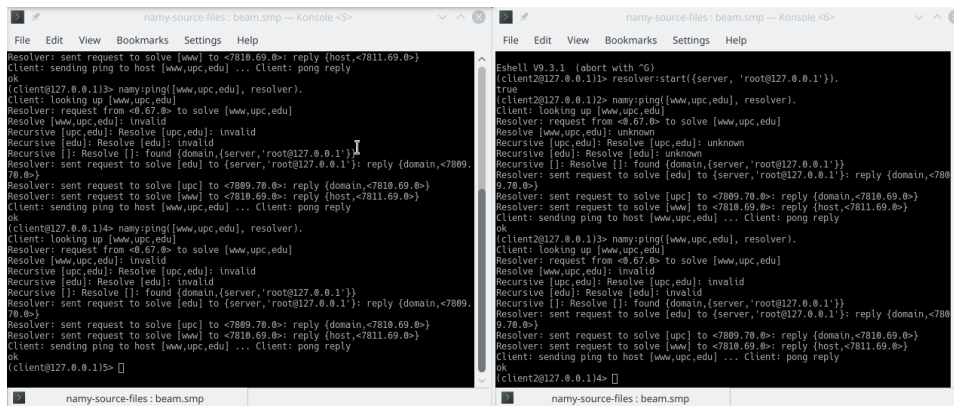
May 15, 2019

## 1 Introduction

*Introduce in a couple of sentences the seminar and the main topic related to distributed systems it covers.*

## 2 Testing

i) Make experiments with several clients asking for name resolution concurrently.

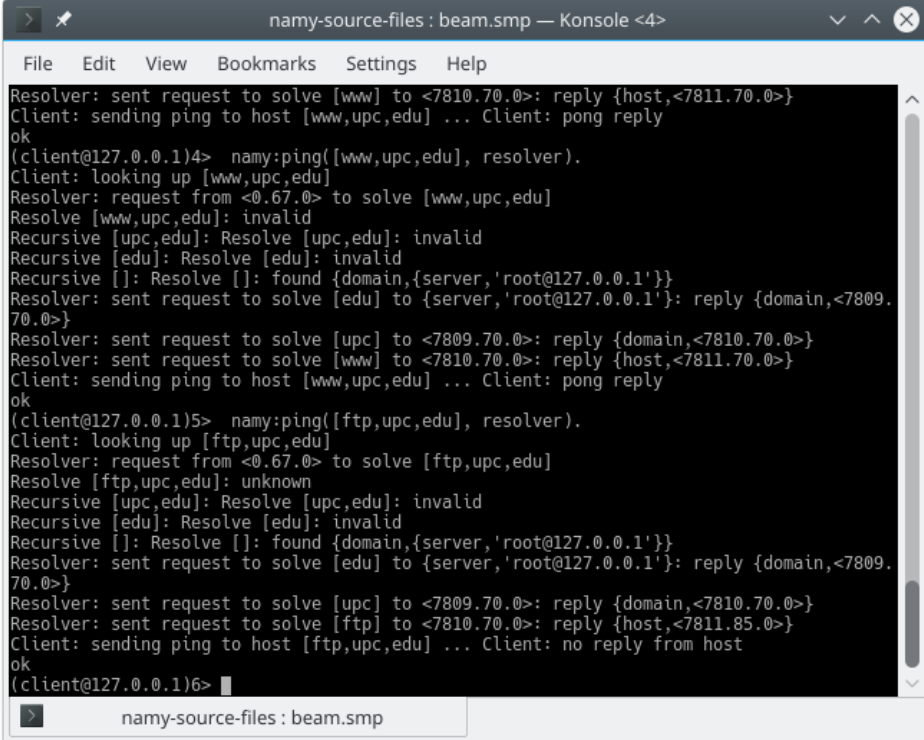


The image shows two terminal windows side-by-side, both titled 'namy-source-files : beam.smp — Konsole <5>'. The left window shows a client at IP 127.0.0.13 sending a ping to a host [www.upc.edu] and then making a series of DNS queries for 'www.upc.edu' and 'upc.edu'. The right window shows a client at IP 127.0.0.12 performing similar actions. Both clients receive responses from a server at IP 7811.69.0, indicating successful name resolution for the 'www' subdomain.

Figure 1: Dos clients fent petició pel mateix host

Podem veure com els dos clients reben exactament la mateixa resolució amb els mateixos passos.

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



```
namy-source-files : beam.smp — Konsole <4>
File Edit View Bookmarks Settings Help
Resolver: sent request to solve [www] to <7810.70.0>: reply {host,<7811.70.0>}
Client: sending ping to host [www,upc,edu] ... Client: pong reply
ok
(client@127.0.0.1)4> namy: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]: 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,<7809.70.0>}
Resolver: sent request to solve [upc] to <7809.70.0>: reply {domain,<7810.70.0>}
Resolver: sent request to solve [www] to <7810.70.0>: reply {host,<7811.70.0>}
Client: sending ping to host [www,upc,edu] ... Client: pong reply
ok
(client@127.0.0.1)5> namy:ping([ftp,upc,edu], resolver).
Client: looking up [ftp,upc,edu]
Resolver: request from <0.67.0> to solve [ftp,upc,edu]
Resolve [ftp,upc,edu]: unknown
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,<7809.70.0>}
Resolver: sent request to solve [upc] to <7809.70.0>: reply {domain,<7810.70.0>}
Resolver: sent request to solve [ftp] to <7810.70.0>: reply {host,<7811.85.0>}
Client: sending ping to host [ftp,upc,edu] ... Client: no reply from host
ok
(client@127.0.0.1)6> 
```

Figure 2: Client fent petició d'un host caigut

### Q) What is the observed behavior?

Quan intentem fer *ping* a un host que està caigut veiem que obtenim la resolució del host antiga però alhora d'intentar comunicar-nos amb ell no obtenim resposta.

iii) Modify the server and the host code so that when they are shut down, they unregister their entry in their corresponding parent domain and repeat experiment ii.

### Q) What is different now?

Com veiem a la figura 3a, la petició arriba fins el servidor *upc* però ell no té cap entrada amb el nom de ftp (ja que previàment l'hem eliminat de la seva taula quan hem parat ftp). El mateix passa si parem el servidor upc (figura 3b), que el seu servidor pare (edu) no sabrà la seva direcció. En canvi, si ho fem amb el servidor root, la seva direcció no serà borrada de l'entrada del

```

[client@127.0.0.1]4> nmap:ping([ftp,upc,edu],resolver).
Client: looking up [ftp,upc,edu]
Resolver: request from <0.67.0> 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,<7889.69.0>}
Resolver: sent request to solve [upc] to <7889.69.0>: reply {domain,<7810.69.0>}
Resolver: sent request to solve [ftp] to <7810.69.0>: unknown
Client: unknown host
ok

[client@127.0.0.1]7> nmap: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
Recursive [upc,edu]: Resolve [upc,edu]: unknown
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,<7889.69.0>}
Resolver: sent request to solve [upc] to <7889.69.0>: unknown
Client: unknown host
ok

```

- (a) Client fent petició d'ftp quan està caigut      (b) Client fent petició d'upc quan està caigut

Figure 3: Peticions amb host/servidors caiguts

```

[client@127.0.0.1]9> nmap: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
Recursive [upc,edu]: Resolve [upc,edu]: unknown
Recursive [edu]: Resolve [edu]: unknown
Recursive []: Resolve []: found {domain,{server,'root@127.0.0.1'}}
Resolver: sent request to solve [edu] to {server,'root@127.0.0.1'}: timeout
Client: unknown host
ok

```

Figure 4: Client fent petició d'root quan està caigut

resolver, però alhora de comunicar-nos no ens serà possible.

### 3 Using the cache

- i) Set a long TTL (i.e. minutes) and 'move' a host (i.e. query for a host name, shut it down, start it up registered under the same name, and finally query it again).

```

File Edit View Bookmarks Settings Help
(v1ersion (k)ill (D)b-tables (d)istribution
Erlang/OTP 20 [erts-9.3.1] [source] [64-bit] [smp:4:4] [ds:4:4:10] [async-threads:1] [h
ue] [kernel-poll:false]
Eshell V9.3.1 (abort with ^G)
[client@127.0.0.1]1> host:start(www, www, {server, 'upc@127.0.0.1'}).
true
Host: create domain www at {server, 'upc@127.0.0.1'}
[client@127.0.0.1]2> host:start(www, www, {server, 'upc@127.0.0.1'}).
true
Host: create domain www at {server, 'upc@127.0.0.1'}
** exception error: bad argument
    in function register/2
    in call from host:start/3 (host.erl, line 5)
[client@127.0.0.1]3> host:start(ftp, ftp, {server, 'upc@127.0.0.1'}).
true
Host: create domain ftp at {server, 'upc@127.0.0.1'}
Host: Ping from <7886.67.0>
Host: Ping from <7886.67.0>
[client@127.0.0.1]4> host:stop(ftp).
Host: Closing down
true
[client@127.0.0.1]5> host:start(ftp, ftp, {server, 'upc@127.0.0.1'}).
true
Host: create domain ftp at {server, 'upc@127.0.0.1'}
Host: Ping from <7886.67.0>
[client@127.0.0.1]6>

[client@127.0.0.1]4> nmap: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]: found {host,<7811.91.0>}
Client: sending ping to host [www,upc,edu] ... Client: pong reply
ok
[client@127.0.0.1]5> nmap:ping([ftp,upc,edu],resolver).
Client: looking up [ftp,upc,edu]
Resolver: request from <0.67.0> to solve [ftp,upc,edu]
Recursive [upc,edu]: Resolve [upc,edu]: found {domain,<7810.69.0>}
Resolver: sent request to solve [ftp] to <7810.69.0>: reply {host,<7811.91.0>}
Client: sending ping to host [ftp,upc,edu] ... Client: pong reply
ok
[client@127.0.0.1]6>

```

Figure 5: Restart host amb la cache habilitada

#### Q) What is the observed behavior?

Com no netejem de la cache l'entrada anterior, quan un host es mou de direcció nosaltres mantenim la seva direcció antiga a la cache fins que el seu TTL s'esgota. Això significa que si durant aquest transcurs ens arriba una petició i fa *match* amb el nom del host retornarem la direcció antiga.

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

La cache sempre ha mantingut la informació del host amb la seva direcció i TTL pertinent. Els servidors que han de ser notificats sobre el canvi són aquells que tenen informació sobre aquest host.

**ii) 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).**

En el cas amb la cache activada:

La primera query farà D missatges ja que aquesta no es trobava guardada en cap cache. A partir d'aleshores els següents  $\frac{T}{F}$  missatges donaran encert en la cache i per tant tant sols necessitarem  $\frac{T}{F}$  missatges. La formula resultant seria, sent M missatges totals:

$$M = (D + \frac{T}{F}) \times \frac{R}{T} \quad (1)$$

En el cas de la cache desactivada:

Al ser el TTL = 0, sempre fallarem en la cache i per tant en cada query necessitarem D missatges. Sabent això aleshores és senzill derivar l'equació:

$$M = \frac{R}{F} \times D \quad (2)$$

**iii) Our cache also suffers from old entries that are never removed. Invalid entries are removed and updated but if we never search for the entry we will not remove it. Make experiments to show that the purging procedure works.**

```
(client#127.0.0.1)> resolver:start([server, 'root#127.0.0.1']).
true
(client#127.0.0.1)> resolver ! status.
Resolver: cache content: [[[]], [domain, [server, 'root#127.0.0.1']], inf]]
status
(client#127.0.0.1)> name:ping([www.upc.edu], resolver).
Client: looking up [www.upc.edu]
Resolver: request from <8428.83.0> to solve [www.upc.edu]
Resolve [www.upc.edu]: unknown
Recursive [upc.edu]: resolve [upc.edu]: unknown
Recursive [edu]: resolve [edu]: unknown
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, <8428.83.0>]
Resolver: sent request to solve [upc] to <8428.83.0>: reply [domain, <8428.83.0>]
Resolver: sent request to solve [www] to <8428.83.0>: reply [host, <8430.83.0>]
Client: sending ping to host [www.upc.edu] ... Client: pong reply
ok
(client#127.0.0.1)> resolver ! status.
Resolver: cache content: [[[]], [domain, [server, 'root#127.0.0.1']], inf], [[edu], [domain, <8428.83.0>], 38], [[upc.edu], [domain, <8428.83.0>], 38], [[www.upc.edu], [host, <8430.83.0>], 38]]
status
(client#127.0.0.1)> resolver ! purge.
Cache: purge
Cache: returning [[[]], [domain, [server, 'root#127.0.0.1']], inf]]
purge
Resolver: reception of cache [[[]], [domain, [server, 'root#127.0.0.1']], inf]]
(client#127.0.0.1)> resolver ! status.
Resolver: cache content: [[[]], [domain, [server, 'root#127.0.0.1']], inf]]
status
(client#127.0.0.1)>
```

Figure 6: Funcionament de la funcionalitat *purge*

## 4 Recursive resolution

i) 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.



Figure 7: Resolució recursiva en múltiples clients

En aquesta nova versió veiem com els resultats són invàlids en el client dret de la figura, però quan amb un altre client, veiem que automàticament es fa el pong cachejat mentre que amb la resolució recursiva no ho aconseguim. De fet per això el dns es tracta d'una primera query iterativa i a continuació tot un procés recursiu, doncs volem aprofitar el màxim totes les búsquedes de tots els usuaris que usin el sistema, tot obtenint uns els beneficis (resultats cachejats) a la primera demanada de la dada (doncs un altre client ja

ho ha demanat) i això en mitjana acaba provocant resultats òptims.

**Q) Which version can exploit caching better? Justify why**

La recursiva doncs tenim cachejats els resultats no només en el propi resolver del client, sinó que ho fem de manera comuna per totes les peticions que ens entrin, així poden oferir les penalitzacions de temps de resolució que han patit aquells usuaris que han arribat a fer la resolució fins al final de la cadena, com a beneficis per altres usuaris que reproduïxin les mateixes cerques, i realment, si el sistema escala considerablement i el nombre d'usuaris es fa immens, l'optimització que produeix en aquest cas la recursivitat és brutal.

## 5 Personal opinion

*Provide your personal opinion of the seminar, indicating whether it should be included in next year's course or not.*

Creiem que ha estat una bona pràctica per acabar d'entendre el funcionament del sistema de dns, que a la pràctica acaba següent un sistema distribuït. Si que s'hauria d'incloure en un pròxim any doncs encara que el dns ja sigui una eina que està més que acabada i que no implementarem mai en un cas real, si que va bé una pràctica així per veure que estem envoltats de sistemes distribuïts, i en aquest cas veiem des de primera mà perquè la resolució de dns és majoritàriament recursiva.