Seminar Report: Namy

Pedro Garcia Mota, Pau Espin Pedrol

May 15, 2013

1 Introduction

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

The aim of this seminar is to implement a distributed name server in erlang similar to DNS. Instead of addresses we will store process identifiers of hosts.

2 Work done

Present the new code that you have added or how you have implemented a required functionality by using small Erlang code snippets (you do not need to copy&paste all the code).

2.1 entry.erl

```
-module(entry).
-export([lookup/2, add/3, remove/2]).

lookup(Req, Entries) ->
    case lists:keyfind(Req, 1, Entries) of
    {_, Entry} ->
      Entry;
    false ->
      unknown
  end.


add(Name, Entry, Entries) ->
  lists:keystore(Name, 1, Entries, {Name, Entry}).

remove(Name, Entries) ->
  lists:keydelete(Name, 1, Entries).
```

2.2 cache.erl

```
-module(cache).
-export([lookup/2, new/0, add/4, remove/2]).
```

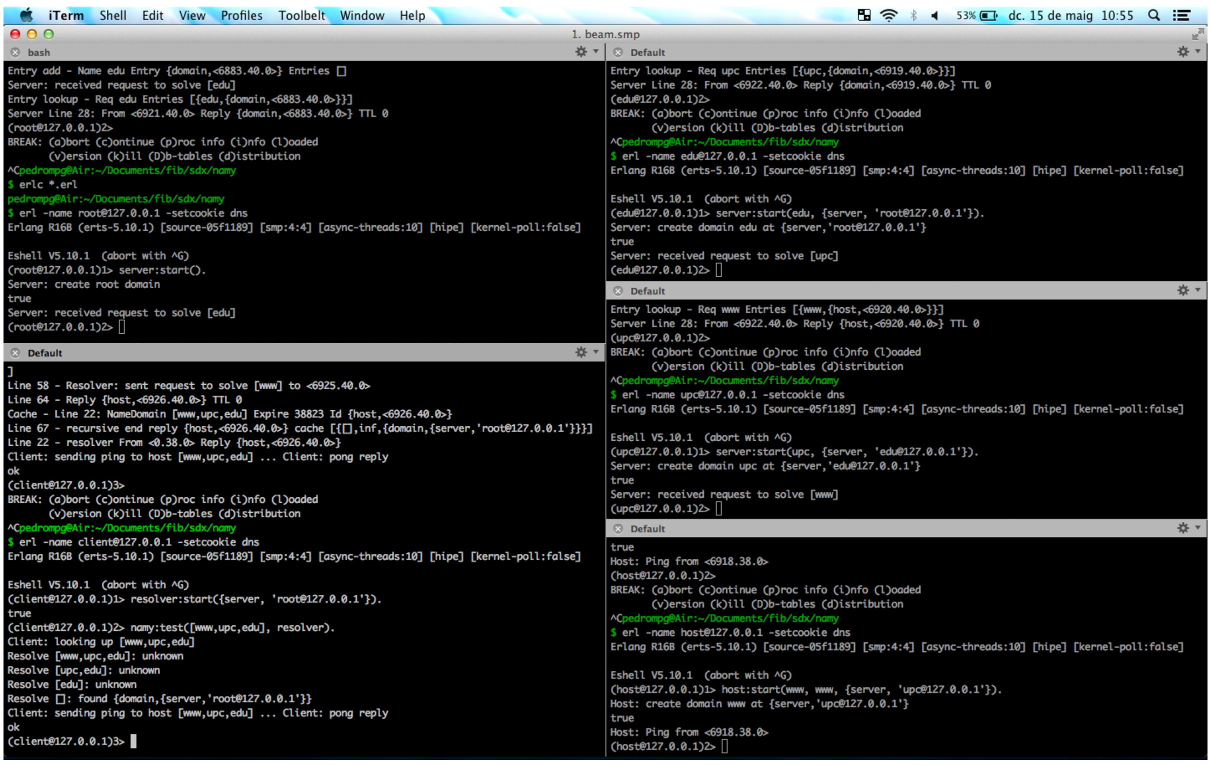
```
lookup(Name, Cache) ->
    case lists:keyfind(Name, 1, Cache) of
    {_, Expire, Id} ->
        case time:valid(Expire, time:now()) of
        true ->
          {ok, Id};
        false ->
          invalid
      end;
    false ->
      unknown
  end.

new() ->
  [].

add(NameDomain, Expire, Reply, Cache) ->
  T = {NameDomain, Expire, Reply},
  lists:keystore(NameDomain, 1, Cache, T).

remove(Name, Cache) ->
  lists:keydelete(Name, 1, Cache).
```

3 Experiments



```
Entry add - Name edu Entry {domain,<6883.40.0>} Entries []
Server: received request to solve [edu]
Entry lookup - Req edu Entries [{edu,{domain,<6883.40.0>}}]
Server Line 28: From <6921.40.0> Reply {domain,<6883.40.0>} TTL 0
(root@127.0.0.1)2>
BREAK: (a)bort (c)ontinue (p)roc info (i)nfo (l)oaded
       (v)ersion (k)ill (D)b-tables (d)istribution
^Cpedrompg@Air:~/Documents/fib/sdx/namy
$ erlc *.erl
pedrompg@Air:~/Documents/fib/sdx/namy
$ erl -name root@127.0.0.1 -setcookie dns
Erlang R16B (erts-5.10.1) [source-05f1189] [smp:4:4] [async-threads:10] [hipe] [kernel-poll:false]

Eshell V5.10.1  (abort with ^G)
(root@127.0.0.1)1> server:start().
Server: create root domain
true
Server: received request to solve [edu]
(root@127.0.0.1)2>
```

```
]
Line 58 - Resolver: sent request to solve [www] to <6925.40.0>
Line 64 - Reply {host,<6926.40.0>} TTL 0
Cache - Line 22: NameDomain [www,upc,edu] Expire 38823 Id {host,<6926.40.0>}
Line 67 - recursive end reply {host,<6926.40.0>} cache [{[],inf,{domain,{server,'root@127.0.0.1'}}}]
Line 22 - resolver From <0.38.0> Reply {host,<6926.40.0>}
Client: sending ping to host [www,upc,edu] ... Client: pong reply
ok
(client@127.0.0.1)3>
BREAK: (a)bort (c)ontinue (p)roc info (i)nfo (l)oaded
       (v)ersion (k)ill (D)b-tables (d)istribution
^Cpedrompg@Air:~/Documents/fib/sdx/namy
$ erl -name client@127.0.0.1 -setcookie dns
Erlang R16B (erts-5.10.1) [source-05f1189] [smp:4:4] [async-threads:10] [hipe] [kernel-poll:false]

Eshell V5.10.1  (abort with ^G)
(client@127.0.0.1)1> resolver:start({server, 'root@127.0.0.1'}).
true
(client@127.0.0.1)2> namy:test([www,upc,edu], resolver).
Client: looking up [www,upc,edu]
Resolve [www,upc,edu]: unknown
Resolve [upc,edu]: unknown
Resolve [edu]: unknown
Resolve []: found {domain,{server,'root@127.0.0.1'}}
Client: sending ping to host [www,upc,edu] ... Client: pong reply
ok
(client@127.0.0.1)3>
```

```
Entry lookup - Req upc Entries [{upc,{domain,<6919.40.0>}}]
Server Line 28: From <6922.40.0> Reply {domain,<6919.40.0>} TTL 0
(edu@127.0.0.1)2>
BREAK: (a)bort (c)ontinue (p)roc info (i)nfo (l)oaded
       (v)ersion (k)ill (D)b-tables (d)istribution
^Cpedrompg@Air:~/Documents/fib/sdx/namy
$ erl -name edu@127.0.0.1 -setcookie dns
Erlang R16B (erts-5.10.1) [source-05f1189] [smp:4:4] [async-threads:10] [hipe] [kernel-poll:false]

Eshell V5.10.1  (abort with ^G)
(edu@127.0.0.1)1> server:start(edu, {server, 'root@127.0.0.1'}).
Server: create domain edu at {server,'root@127.0.0.1'}
true
Server: received request to solve [upc]
(edu@127.0.0.1)2>
```

```
Entry lookup - Req www Entries [{www,{host,<6920.40.0>}}]
Server Line 28: From <6922.40.0> Reply {host,<6920.40.0>} TTL 0
(upc@127.0.0.1)2>
BREAK: (a)bort (c)ontinue (p)roc info (i)nfo (l)oaded
       (v)ersion (k)ill (D)b-tables (d)istribution
^Cpedrompg@Air:~/Documents/fib/sdx/namy
$ erl -name upc@127.0.0.1 -setcookie dns
Erlang R16B (erts-5.10.1) [source-05f1189] [smp:4:4] [async-threads:10] [hipe] [kernel-poll:false]

Eshell V5.10.1  (abort with ^G)
(upc@127.0.0.1)1> server:start(upc, {server, 'edu@127.0.0.1'}).
Server: create domain upc at {server,'edu@127.0.0.1'}
true
Server: received request to solve [www]
(upc@127.0.0.1)2>
```

```
true
Host: Ping from <6918.38.0>
(host@127.0.0.1)2>
BREAK: (a)bort (c)ontinue (p)roc info (i)nfo (l)oaded
       (v)ersion (k)ill (D)b-tables (d)istribution
^Cpedrompg@Air:~/Documents/fib/sdx/namy
$ erl -name host@127.0.0.1 -setcookie dns
Erlang R16B (erts-5.10.1) [source-05f1189] [smp:4:4] [async-threads:10] [hipe] [kernel-poll:false]

Eshell V5.10.1  (abort with ^G)
(host@127.0.0.1)1> host:start(www, www, {server, 'upc@127.0.0.1'}).
Host: create domain www at {server,'upc@127.0.0.1'}
true
Host: Ping from <6918.38.0>
(host@127.0.0.1)2>
```

Working version of the vanilla set up with TTL of 0 seconds.

4 Open questions

- i) In the vanilla set-up the time-to-live (TTL) is zero seconds. What happens if we extend this to two or four seconds? Try to quantify the amount of message traffic reduced.

  If we enable the cache, changing the TTL value for a number greater than 0, all name resolutions are cached, so the amount of message traffic is reduced.

  Let's do, for all tests, Np pings from host A to host B ('www').

  Let's assume we have 1 Root nameserver node, an 'edu' nameserver node, a 'upc' namesever node, which sums a total of up to *NameServers*=3 nameservers involved in the name lookup.

  Name reslutions have to be redone every TTL seconds, thus the amount of name resolutions to be done for *Np*=1000 pings at *PingPeriod*=1 seconds per ping is:

  TotalTime $= \frac{Np}{PingPeriod}$

  NsResolutions $= \frac{TotalTime}{TTL} = \frac{\frac{Np}{PingPeriod}}{TTL}$

  It can be easily probed that as NsResolutions increases, the amount of total messages increases by the amount of messages needed to resolve the name. In this case we have 3 nameservers which interacts with the host doing the ping, so each time we need to do the name resolution we have to request and get the response at each level, at a total cost of Nameservers*2 messages.

  That makes Nmessages = (Np $\cdot 2$) + $(NsResolutions \cdot NsServers \cdot 2) = (Np \cdot 2) + (\frac{\frac{Np}{PingPeriod}}{TTL} \cdot NsServers \cdot 2)$

  Note: Np is multiplied by 2 because we count request and reply.

  So, in our case, depending only on TTL: Nmessages = (2000) + $(\frac{6000}{TTL})$

  Which for TTL=2 makes a total of 5000 messages, and for TTL=4 the amount is of 3500.

- ii) Extend TTL to a minute and analyze the impact (e.g. how many nodes need to know about the change? what happens with cached information?) of the following changes:

  - a) shutdown a host and start it up registered under the same name;

    After shutting down the 'www' host and starting it up again with the same name the client does not reply any petitions. This happen because the cache entry for the 'www' host is not invalidated and the resolver is trying to contact a process that does not exist anymore.

    ```
    (client@127.0.0.1)4> namy:test([www,upc,edu], resolver).
    Client: looking up [www,upc,edu]
    Resolve [www,upc,edu]: found {host,<6926.40.0>}
    Client: sending ping to host [www,upc,edu] ... Client: no reply
    ok
    ```

- b) shutdown a host and start it up registered under a new name.

  After shutting down the 'www' host and starting it up again with the name 'ftp' the resolver uses the cache to ask the [upc,edu] server for the new host, locates it and replies. After the first petition the complete the [ftp,upc,edu] entry is already cached.

  ```
  (client@127.0.0.1)19> namy:test([ftp,upc,edu], resolver).
  Client: looking up [ftp,upc,edu]
  Resolve [ftp,upc,edu]: unknown
  Resolve [upc,edu]: found {domain,<6925.40.0>}
  Client: sending ping to host [ftp,upc,edu] ... Client: pong reply
  ok
  (client@127.0.0.1)20> namy:test([ftp,upc,edu], resolver).
  Client: looking up [ftp,upc,edu]
  Resolve [ftp,upc,edu]: found {host,<6926.67.0>}
  Client: sending ping to host [ftp,upc,edu] ... Client: pong reply
  ok
  ```

- 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. What can we do to solve this issue?

  We could set a timer or a separate process on the resolver to check for invalid entries in the cache and remove them.

5 Personal Opinion

Your personal opinion of the seminar, including whether it should be included in next year's course or no .

The seminar should be included in the next course as it expands the concepts of name servers and gives the opportunity to implement a naming distributed system similar to DNS, having a better idea of how these systems are implemented.