Seminar Report: Chatty

Pedro Garcia Mota, Pau Espin Pedrol

March 6, 2013

1 Introduction

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

Implementation of a distributed system that will allow the user to chat among buddies. The main topic is communication between processes in different machines.

2 Work done

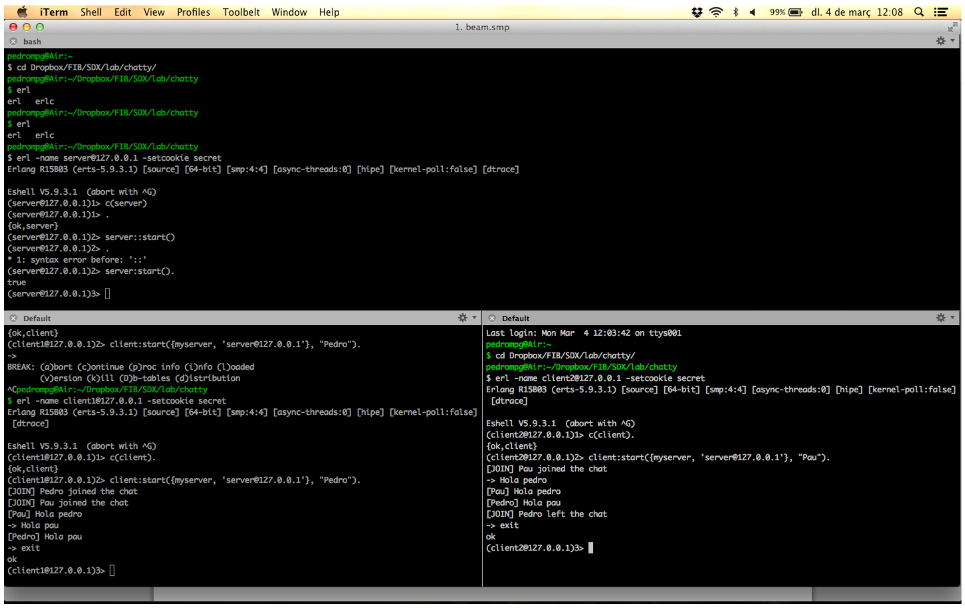
Present the new code that you have added or how you have implemented a required functionality by using small Erlang code snippets (you do not need to copy&paste all the code).

All the changes are commented in the code files.

3 Experiments

Describe the experiments you did and provide evidence of the results you got (e.g., use screenshots). In addition, you may provide figures or tables with experimental results of the system evaluation. For each seminar, we will provide you with some guidance on which kind of evaluation you should do.

On the first image we can see the server running on the top terminal window and two clients running on the bottom tabs.



```
pedrompg@Air:~
$ cd Dropbox/FIB/SDX/lab/chatty/
pedrompg@Air:~/Dropbox/FIB/SDX/lab/chatty
$ erl
erl   erlc
pedrompg@Air:~/Dropbox/FIB/SDX/lab/chatty
$ erl
erl   erlc
pedrompg@Air:~/Dropbox/FIB/SDX/lab/chatty
$ erl -name server@127.0.0.1 -setcookie secret
Erlang R15B03 (erts-5.9.3.1) [source] [64-bit] [smp:4:4] [async-threads:0] [hipe] [kernel-poll:false] [dtrace]

Eshell V5.9.3.1  (abort with ^G)
(server@127.0.0.1)1> c(server)
(server@127.0.0.1)1> .
{ok,server}
(server@127.0.0.1)2> server::start()
(server@127.0.0.1)2> .
* 1: syntax error before: '::'
(server@127.0.0.1)2> server:start().
true
(server@127.0.0.1)3> 
```

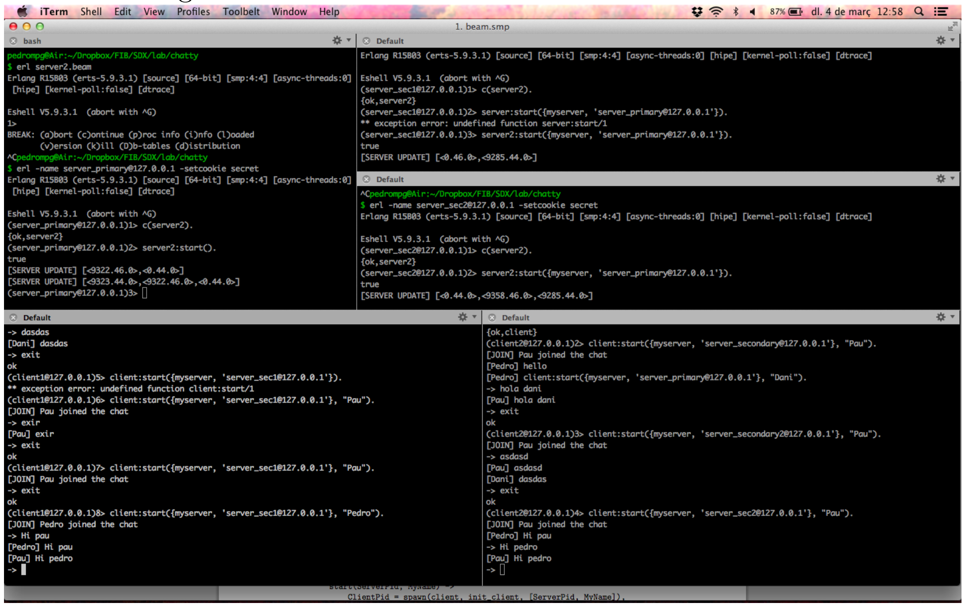
```
{ok,client}
(client1@127.0.0.1)2> client:start({myserver, 'server@127.0.0.1'}, "Pedro").
->
BREAK: (a)bort (c)ontinue (p)roc info (i)nfo (l)oaded
       (v)ersion (k)ill (D)b-tables (d)istribution
^Cpedrompg@Air:~/Dropbox/FIB/SDX/lab/chatty
$ erl -name client1@127.0.0.1 -setcookie secret
Erlang R15B03 (erts-5.9.3.1) [source] [64-bit] [smp:4:4] [async-threads:0] [hipe] [kernel-poll:false]
 [dtrace]

Eshell V5.9.3.1  (abort with ^G)
(client1@127.0.0.1)1> c(client).
{ok,client}
(client1@127.0.0.1)2> client:start({myserver, 'server@127.0.0.1'}, "Pedro").
[JOIN] Pedro joined the chat
[JOIN] Pau joined the chat
[Pau] Hola pedro
-> Hola pau
[Pedro] Hola pau
-> exit
ok
(client1@127.0.0.1)3> 
```

```
Last login: Mon Mar  4 12:03:42 on ttys001
pedrompg@Air:~
$ cd Dropbox/FIB/SDX/lab/chatty/
pedrompg@Air:~/Dropbox/FIB/SDX/lab/chatty
$ erl -name client2@127.0.0.1 -setcookie secret
Erlang R15B03 (erts-5.9.3.1) [source] [64-bit] [smp:4:4] [async-threads:0] [hipe] [kernel-poll:false]
 [dtrace]

Eshell V5.9.3.1  (abort with ^G)
(client2@127.0.0.1)1> c(client).
{ok,client}
(client2@127.0.0.1)2> client:start({myserver, 'server@127.0.0.1'}, "Pau").
[JOIN] Pau joined the chat
-> Hola pedro
[Pau] Hola pedro
[Pedro] Hola pau
[JOIN] Pedro left the chat
-> exit
ok
(client2@127.0.0.1)3> 
```

On the second image we can see the master server running on the top left window, two slave servers running on the top right windows and two clients running on the bottom windows.



```
pedrompg@Air:~/Dropbox/FIB/SDX/lab/chatty
$ erl server2.beam
Erlang R15B03 (erts-5.9.3.1) [source] [64-bit] [smp:4:4] [async-threads:0]
 [hipe] [kernel-poll:false] [dtrace]

Eshell V5.9.3.1  (abort with ^G)
1>
BREAK: (a)bort (c)ontinue (p)roc info (i)nfo (l)oaded
       (v)ersion (k)ill (D)b-tables (d)istribution
^Cpedrompg@Air:~/Dropbox/FIB/SDX/lab/chatty
$ erl -name server_primary@127.0.0.1 -setcookie secret
Erlang R15B03 (erts-5.9.3.1) [source] [64-bit] [smp:4:4] [async-threads:0]
 [hipe] [kernel-poll:false] [dtrace]

Eshell V5.9.3.1  (abort with ^G)
(server_primary@127.0.0.1)1> c(server2).
{ok,server2}
(server_primary@127.0.0.1)2> server2:start().
true
[SERVER UPDATE] [<9322.46.0>,<0.44.0>]
[SERVER UPDATE] [<9323.44.0>,<9322.46.0>,<0.44.0>]
(server_primary@127.0.0.1)3> 
```

```
Erlang R15B03 (erts-5.9.3.1) [source] [64-bit] [smp:4:4] [async-threads:0] [hipe] [kernel-poll:false] [dtrace]

Eshell V5.9.3.1  (abort with ^G)
(server_sec1@127.0.0.1)1> c(server2).
{ok,server2}
(server_sec1@127.0.0.1)2> server:start({myserver, 'server_primary@127.0.0.1'}).
** exception error: undefined function server:start/1
(server_sec1@127.0.0.1)3> server2:start({myserver, 'server_primary@127.0.0.1'}).
true
[SERVER UPDATE] [<0.46.0>,<9285.44.0>]
```

```
^Cpedrompg@Air:~/Dropbox/FIB/SDX/lab/chatty
$ erl -name server_sec2@127.0.0.1 -setcookie secret
Erlang R15B03 (erts-5.9.3.1) [source] [64-bit] [smp:4:4] [async-threads:0] [hipe] [kernel-poll:false] [dtrace]

Eshell V5.9.3.1  (abort with ^G)
(server_sec2@127.0.0.1)1> c(server2).
{ok,server2}
(server_sec2@127.0.0.1)2> server2:start({myserver, 'server_primary@127.0.0.1'}).
true
[SERVER UPDATE] [<0.44.0>,<9358.46.0>,<9285.44.0>]
```

```
-> dasdas
[Dani] dasdas
-> exit
ok
(client1@127.0.0.1)5> client:start({myserver, 'server_sec1@127.0.0.1'}).
** exception error: undefined function client:start/1
(client1@127.0.0.1)6> client:start({myserver, 'server_sec1@127.0.0.1'}, "Pau").
[JOIN] Pau joined the chat
-> exir
[Pau] exir
-> exit
ok
(client1@127.0.0.1)7> client:start({myserver, 'server_sec1@127.0.0.1'}, "Pau").
[JOIN] Pau joined the chat
-> exit
ok
(client1@127.0.0.1)8> client:start({myserver, 'server_sec1@127.0.0.1'}, "Pedro").
[JOIN] Pedro joined the chat
-> Hi pau
[Pedro] Hi pau
[Pau] Hi pedro
-> 
```

```
{ok,client}
(client2@127.0.0.1)2> client:start({myserver, 'server_secondary@127.0.0.1'}, "Pau").
[JOIN] Pau joined the chat
[Pedro] hello
[Pedro] client:start({myserver, 'server_primary@127.0.0.1'}, "Dani").
-> hola dani
[Pau] hola dani
-> exit
ok
(client2@127.0.0.1)3> client:start({myserver, 'server_secondary2@127.0.0.1'}, "Pau").
[JOIN] Pau joined the chat
-> asdasd
[Pau] asdasd
[Dani] dasdas
-> exit
ok
(client2@127.0.0.1)4> client:start({myserver, 'server_sec2@127.0.0.1'}, "Pau").
[JOIN] Pau joined the chat
[Pedro] Hi pau
-> Hi pedro
[Pau] Hi pedro
-> 
```

```
    start(ServerPid, MyName) ->
        ClientPid = spawn(client, init_client, [ServerPid, MyName]),
```

The reader will find, with this document, these two captions attached as own png files in full resolution in the deliverable.

4 Open questions

Try to answer all the open questions in the documentation. When possible, do experiments to support your answers.

Open Questions - 1.

i) Does this solution scale when the number of users increase?

No, if would result in massive workload on the single server, and if the server crashes the whole system stops working.

ii) What happens if the server fails?
All the service goes down. All users are unable to comunicate each other.

iii) Are the messages from a single client to any other client guaranteed to be delivered in the order they were issued? (hint: search for the 'order of message reception' in Erlang FAQ)

Yes, but only within one process, which is the case. According to Erlang's FAQ: "If there is a live process and you send it message A and then message B, it's guaranteed that if message B arrived, message A arrived before it."

iv) What about the messages from different clients (hint: think on several clients sending messages concurrently)?
It's not guaranted, as each client spends an extra time sending its information through the server, and there's the posibility that one client, because of network problems or high worload on the node for example, takes more time to send a message to the server than another client which actually issued it after the first one.

v) What happens if a user joins/leaves the chat while the server is broadcasting a message?

It is undefined, it depends on the order of the events. It depends on the client ID being added to the list of clients before or after the server accesses it to send a message to all of the clients in the list.

Open Questions - 2.

i) What happens if a server fails?

The clients connected to that server would no longer be able to chat with other users.

ii) Are the messages from a single client to any other client guaranteed to be delivered in the order they were issued?

The behaviour is the same from the single server implementation.

iii) What happens if there are concurrent requests from servers to join or leave the system?

Nothing, as the FAQ states: "Delivery is guaranteed if nothing breaks"

iv) What are the advantages and disadvantages of this implementation regarding the previous one?

The main advantage is that the system is much more robust, if a server crashes the chat still works for the clients connected in the other servers. It is also much more scalable. On the other hand, the main disasdvantatges are the increased compleixity of the system and the need of further resources for running the system.

5 Personal Opinion

Your personal opinion of the seminar, including whether it should be included in next year's course or not.

The seminar presents a easy introduction into the erlang language and concepts as message passing between different processes in the same machine and on different machines over the same network.