

Seminar Report: Chatty

Daniel García, Aitor Ardila, Oscar Hernández
February 4, 2016

1 - Introduction

We have created a general chat service. It is based on a server that maintains a list of its clients, when it receives a message from one of them, the message is broadcasted to all the members in the list. It is basically based on the message passing between nodes that Erlang implements.

2 - Work done

We have done 2 implementations: The first one is only using one central server which manage all the connections. The problem is that if this server goes down, all clients will lose the connection. To solve this problem, we have implemented a second version, which can use n servers and every client is connected to one of these servers but can also communicate with clients from other servers, because all servers can communicate with each other.

As we only had to fill some gaps in the code, there were not many options to implement the code of this seminar. The source files are attached in this document.

3 - Experiments

First of all, we implemented the first version which only uses one main server to establish all the connections. After compiling both the server code and the client code, we started 1 server and started 4 clients. After that, we checked that chat was working posting messages and see everybody received the same message.

```
SDX: tcsh - Konsole <2>
File Edit View Bookmarks Settings Help
(bob@127.0.0.1)2> client:start(myserver,'Aitor').
[JOIN] Aitor joined the chat
[JOIN] Dani joined the chat
-> It works!!!!
[Aitor] It works!!!!
[SERVER UPDATE] [<7282.41.0>,<0.41.0>]
[JOIN] oscar joined the chat
-> esto funciona!
[Aitor] esto funciona!
[JOIN] Oscar left the chat
[JOIN] Pere joined the chat
-> hola
[Aitor] hola
[JOIN] Dani joined the chat
-> ol
[Aitor] ol
[Dani] pedro
[JOIN] Dani left the chat
[JOIN] otro joined the chat
[otro]
[Pere]
-> aola
[Aitor] aola
[Pere] hello
[JOIN] Pere left the chat
[otro] ola
[JOIN] otro left the chat
-> halt().
[Aitor] halt().
[JOIN] Dani left the chat
-> exit
```

As you can see, we have used different instances of Erlang on other O.S.
Two clients on Linux Open Suse and the other ones on Windows 7.

```
(dani@10.192.201.100)9> client:start(myserver,"Dani").
[JOIN] Dani joined the chat
-> Hola
[Dani] Hola
[Dani2] eyyy
-> █
```

```
(dani2@10.192.201.100)1> client:start({myserver,'dani@10.192.201.100'},"Dani2").
[JOIN] Dani2 joined the chat
[JOIN] Dani joined the chat
[Dani] Hola
-> eyyy
[Dani2] eyyy
-> █
```

To start the server we run the following command:

```
1133835/SDX> erl -name bob@127.0.0.1 -setcookie secret
Erlang/OTP 17 [erts-6.1] [source] [64-bit] [smp:4:4] [async-threads:10] [hipe] [kernel-poll:false]
```

Secondly, we have implemented the second version of code, including a list of servers between communications. Thus, the solution now provides a level of security against server crashes.

```

File Edit View Bookmarks Settings Help

we trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

#1) Respect the privacy of others.
#2) Think before you type.
#3) With great power comes great responsibility.

root's password:
^CSorry, try again.
root's password:
^CSorry, try again.
root's password:
(pam_mount.c:173): conv->conv(...): Authentication failure
sudo: PAM authentication error: Authentication information cannot be recovered
1125925@SDX:~$ sudo ifconfig

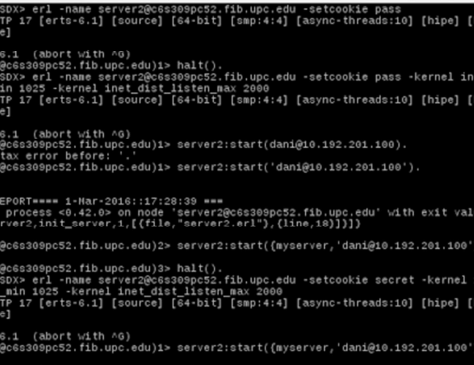
we trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

#1) Respect the privacy of others.
#2) Think before you type.
#3) With great power comes great responsibility.

root's password:
(pam_mount.c:173): conv->conv(...): Authentication failure
sudo: PAM authentication error: Authentication information cannot be recovered
1125925@SDX:~$
1125925@SDX:~$ traceroute 8.8.8.8
traceroute: Command not found.
1125925@SDX:~$ ls
client.beam  seminar-report-template.pdf  server2.beam  server.beam
client.erl   seminar-report-template.tex  server2.erl   server.erl
1125925@SDX:~$

SDX: tsh

```



```
SDX : beam.smp - Konsole
File Edit View Bookmarks Settings Help

1125935/SDX> erl -name server2@cgs309pc52.fib.upc.edu -setcookie pass
1125935/OTF 17 [erts-6.1] [source] [64-bit] [asp4:4] [async-threads:10] [hipe] [kernel-p
oll:false]

Shell v5.1 (abort with ^g)
(server2@cgs309pc52.fib.upc.edu)> halt().
1125935/SDX> erl -name server2@cgs309pc52.fib.upc.edu -setcookie pass -kernel inet_dis
t_listen_min 1025 -kernel inet_dist_listen_max 2000
Erlang/OTP 17 [erts-6.1] [source] [64-bit] [asp4:4] [async-threads:10] [hipe] [kernel-p
oll:false]

Shell v5.1 (abort with ^g)
(server2@cgs309pc52.fib.upc.edu)> server2:start(dan@10.192.201.100).
1> syntax error before: .
(server2@cgs309pc52.fib.upc.edu)> server2:start('dan@10.192.201.100').
true

**ERROR REPORT**==== 1-Mar-2016:17:28:39 ===
Error in process <0.42.0> on node 'server2@cgs309pc52.fib.upc.edu' with exit value: {bad
arg, [{server2_int_server,1,[{file,"server2.erl"},{line,18}]}]}

(server2@cgs309pc52.fib.upc.edu)> server2:start({myserver,'dan@10.192.201.100'}).
true
(server2@cgs309pc52.fib.upc.edu)> halt().
1125935/SDX> erl -name server2@cgs309pc52.fib.upc.edu -setcookie secret -kernel inet_dis
t_listen_min 1025 -kernel inet_dist_listen_max 2000
Erlang/OTP 17 [erts-6.1] [source] [64-bit] [asp4:4] [async-threads:10] [hipe] [kernel-p
oll:false]

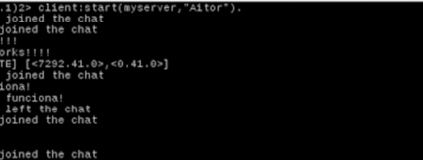
Shell v5.1 (abort with ^g)
(server2@cgs309pc52.fib.upc.edu)> server2:start({myserver,'dan@10.192.201.100'}).
true
[SERVER UPDATE] <0.41.0>,<6843.45.0>
(server2@cgs309pc52.fib.upc.edu)> []
```

```
SDX: beam.smp - Konsole <2>
File Edit View Bookmarks Settings Help

[0ro] ok
[30ro] Otro left the chat
[AIor] halt().
-> exit
ok
(boby@127.0.0.1)2> halt().
1125935/SDX> cat /etc/in
ini.d/ / imputrc inserv.conf install.inf
1125935/SDX> erl -name boby@127.0.0.1 -setcookie secret
Erlang/OTP 17 [erts-6.1] [source] [64-bit] [smp:4:4] [hipe] [kernel-p
lib:false]

Eshell v6.1 (abort with ^G)
(boby@127.0.0.1)1> client:start((myserver,'server2@c6309pc52.fib.upc.edu'),"Clientes").
-> exit
ok
(boby@127.0.0.1)2> client:start((myserver,'server2@c6309pc52.fib.upc.edu'),"Clientes").

[JOIN] Client3 joined the chat
[Client2]
[Client2]
[Client1]
-> hola
[Client3] hola
-> ei
[Client3] ei
-> te paso la repuesta
[Client2] te paso la repuesta
-> del examen
[Client3] del examen
-> shhh
[Client3] shhh
[Client2] shhh
[Client2] ajajaja
■
```



```

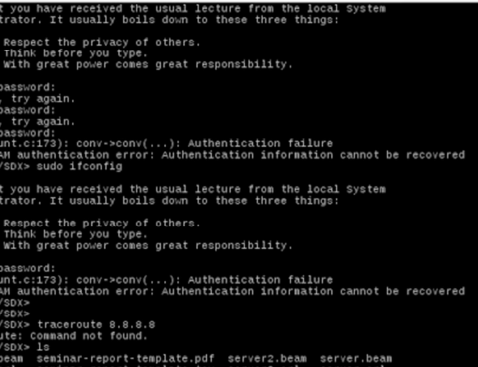
SDX: tcsh - Konsole <2>

File Edit View Bookmarks Settings Help

[Bob@127.0.0.1:33] client: start(myserver, 'Aitor').
[JOIN] Aitor joined the chat
[JOIN] Dani joined the chat
[Aitor] It works!!!!
[SERVER UPDATE] [7292.41.0>,0.41.0>]
[JOIN] Oscar joined the chat
-> esto funciona!
[Aitor] esto funciona!
[JOIN] Oscar left the chat
[JOIN] Pere joined the chat
-> hola
[Aitor] hola
[JOIN] Dani joined the chat
-> ol
[Aitor] ol
[Dani] pedro
[JOIN] Dani left the chat
[JOIN] Otro joined the chat
[Otro] puta
[Pere] ol
-> aola
[Aitor] aola
[Pere] hello
[JOIN] Pere left the chat
[Otro] oia
[JOIN] Otro left the chat
-> halt().
[Aitor] halt().
[JOIN] Dani left the chat
-> exit
ok
(bob@127.0.0.1:33)> halt().
123532/sdx>

```

SDX: tcsh



```

File Edit View Bookmarks Settings Help

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

#1) Respect the privacy of others.
#2) Think before you type.
#3) With great power comes great responsibility.

root's password:
^CSorry, try again.
root's password:
^CSorry, try again.
root's password:
(pam_mount.c:173): conv=>conv(...): Authentication failure
sudo: PAM authentication error: Authentication information cannot be recovered
1125932/SDX> sudo ifconfig

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

#1) Respect the privacy of others.
#2) Think before you type.
#3) With great power comes great responsibility.

root's password:
(pam_mount.c:173): conv=>conv(...): Authentication failure
sudo: PAM authentication error: Authentication information cannot be recovered
1125932/SDX>
1125932/SDX>
1125932/SDX> traceroute 8.8.8.8
traceroute: Command not found.
1125932/SDX> ls
client.beam  seminar-report-template.pdf  server2.beam  server.beam
client.erl   seminar-report-template.tex  server2.erl   server.erl
1125932/SDX> pwd
/

```

SDX: tch\$

```
SDX : beam samp - Konsole
File Edit View Bookmarks Settings Help

true
(server2@c6s309pc52.fib.upc.edu)> halt().
1125935/sdpx.er1 -name server2@c6s309pc52.fib.upc.edu -setcookie secret -kernel inet_dis
listen_min 1025 -kernel inet_dist_listen_max 2000
 Erlang/OTP 17 [erts-9.1] [source] [64-bit] [smp:4:4] [async-threads:10] [hipe] [kernel-p
oll:false]

Shell v6.1 (abort with ^C)
(server2@c6s309pc52.fib.upc.edu)> server2:start([myserver,'dan13@10.192.201.100']).
true
(server2@c6s309pc52.fib.upc.edu)> server2:start([myserver,'dan13@10.192.201.100']).
** exception error: bad argument
    in function register/2
        called as register(myserver,<0.48.0>)
    in call from server2:start/1 (server2.erl, line 12)
(server2@c6s309pc52.fib.upc.edu)> exit
(server2@c6s309pc52.fib.upc.edu)> halt().
    2: syntax error before halt
(server2@c6s309pc52.fib.upc.edu)> halt().
1125835/sdpx.er1 -name server2@c6s309pc52.fib.upc.edu -setcookie secret -kernel inet_dis
listen_min 1025 -kernel inet_dist_listen_max 2000
 Erlang/OTP 17 [erts-9.1] [source] [64-bit] [smp:4:4] [async-threads:10] [hipe] [kernel-p
oll:false]

Shell v6.1 (abort with ^C)
(server2@c6s309pc52.fib.upc.edu)> server2:start([myserver,'dan13@10.192.201.100']).
true
(server2@c6s309pc52.fib.upc.edu)> server2:start([myserver,'dan13@10.192.201.100']).
** exception error: bad argument
    in function register/2
        called as register(myserver,<0.48.0>)
    in call from server2:start/1 (server2.erl, line 12)
[SERVER UPDATE] [<0.48.0>,<843.40.0>]
(server2@c6s309pc52.fib.upc.edu)>
```

```
SDX : beam.smp -- Konsola <2>

File Edit View Bookmarks Settings Help

[any@ntp 17 [erts-6.1] [source] [64-bit] [smp:4:4] [async-threads:10] [hipe] [kernel-0]
[all:raise]]

[shell v0.1 (abort with ^G)]
[booby@127.0.0.1:1]> client:start([myserver, 'server2@c6309pc52.fib.upc.edu'], 'Clientes')
-> exit
ok
[booby@127.0.0.1:2]> client:start([myserver, 'server2@c6309pc52.fib.upc.edu'], 'Clientes')

[JOIN] Clientes joined the chat
[Client2]
[Client2]
[Client1]
-> hola
[Client2] hola
-> ei
[Client2] ei
-> te paso la respuesta
[Client2] te paso la respuesta
-> del examen
[Client2] del examen
-> shhh
[Client2] shhh
[Client2] ajaja
[Client2] escribe exit, ya veras que divertido
[JOIN] Gani joined the chat
[Client2] hola
[Dani] hola
[Dani] me piro
[Client2] el server se va a dsconectar
-> vale mierda porque tambien estoy conectado
[Client2] vale mierda porque tambien estoy conectado
-> ola
-> hola
```

```
SDX : beam.smp - Konsole <3>
File Edit View Bookmarks Settings Help

[0ani] pedro
[JOIN] Dani left the chat
[JOIN] Otro joined the chat
[otro] puta
[per] ol
-> aola
[Ator] aola
[per] hello
[JOIN] Pere left the chat
[otro] ola
[JOIN] Otro left the chat
-> halt().
[Ator] halt().
[JOIN] Dani left the chat
-> exit
ok
(bob@127.0.0.1:3)> halt().
11:35:33/sdx> erl -name bob@127.0.0.1 -setcookie secret
Erlang/OTP 17 [erts-6.1] [source] [64-bit] [smp:4:4] [async-threads:10] [hipe] [kernel-poll:false]

Eshell v6.1 (abort with ^G)
(bob@127.0.0.1)> client:start([myserver, 'server2@c6s39pc52.fab.upc.edu'], 'Dani').
[client2] hola
-> hola
[0ani] hola
-> me piro
[0ani] me piro
-> el server se va a desconectar
[0ani] el server se va a desconectar
[client3] vale mierda porque tambien estoy conectao
-> ola
```

As we can see from the screenshots, all clients can receive the same message when one of them posts something.

4 - Open questions

Server 1

Does this solution scale when the number of users increase?

It won't scale well as all the clients rely on the same server. If the number of clients increase too much, one server wouldn't be able to attend all the client requests and would stop working properly.

What happens if the server fails?

If the server fails, the chat service will stop working. The key of this chat is the list of clients the server has, but given that this particular code is not implementing any kind of state, even if the server started running again, the chat previously established wouldn't work and the clients would need to call again the join function.

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

Yes the order is guaranteed as they are from the same client.

Are the messages sent concurrently by several clients guaranteed to be delivered to any other client in the order they were issued?

In this case not guaranteed because the messages are sent from different processes and (probably) from different places. So the order can't be determined with precision.

Is it possible that a client receives a response to a message from another client before receiving the original message from a third client?

It is not possible as the server can only address one request at a time. If there is a response for a message, it means that the server has previously broadcasted the original message. So a third client would never receive a conversation between two clients in a different order.

If a user joins or leaves the chat while the server is broadcasting a message, will he/she receive that message?

Yes because the server is busy attending the broadcast request, so the "leave chat" request won't be received until the server finishes broadcasting the message.

Server 2

What happens if a server fails?

If the server fails the chat service will only stop working for the clients that joined the chat through the server that has failed. The chat for the other clients would continue normally.

Do your answers to previous questions iii, iv, and v still hold in this implementation?

iii) Continues being guaranteed.

iv) Continues the same.

v) Yes, it isn't guaranteed that a server is broadcasting a message in the same order that clients sent, because they don't have the same clock and the messages don't have any timestamp.

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

If that is the case, the clients will see that new clients have joined (in the case that the server joins), or some clients have gone in case the server leaves.

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

The main advantage is that it scales better, with N servers we would be able to attend infinite requests.

5 - Personal opinion

In our opinion, this seminar is a good approach to the programming with Erlang. It is a good way to see the communication between Erlang shells and you can easily verify if the communication is working or not. For this reason, we would recommend to use this seminar for future courses.