

Seminar Report: Chatty

Carles,Ricard

July 20, 2019

**Carles Tornel
Jesus Alfaro
Ricard Abril**

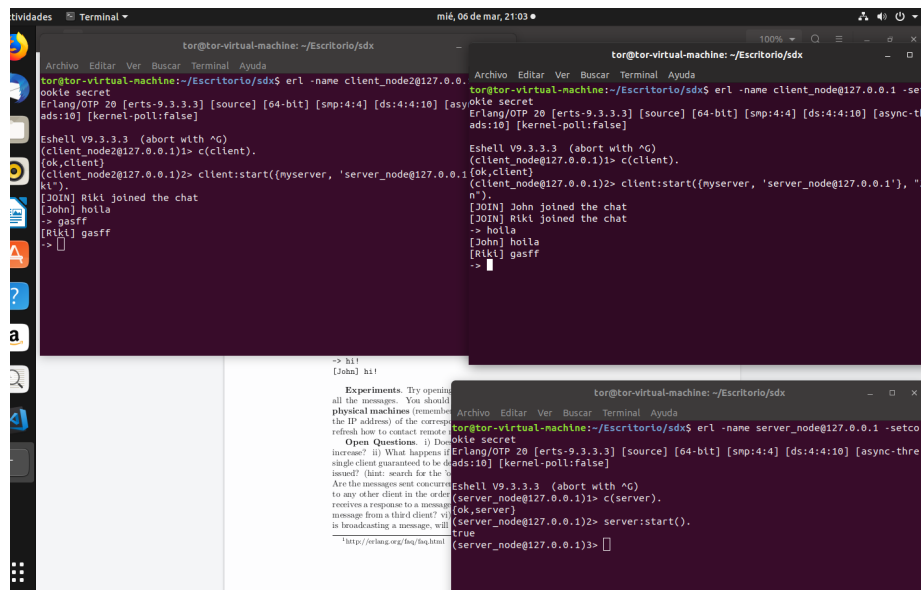
1 Introduction

Aquesta pràctica tenia com a objectiu familiaritzar-se amb Erlang, programant la comunicació entre diferents nodes amb un servidor, o més, com intermediari. Vam haver de programar dos fitxers de servidors i un de client, que feiem servir per enviar missatges a altres clients.

2 Experiments

1. Try opening 2 or 3 clients and test that all of them receive all the messages. You should also try to communicate clients on different physical machines (remember to change the local IP by the domain name (or the IP address) of the corresponding machines). Read the 'Erlang primer' to refresh how to contact remote nodes. Per obrir un numero N de clients i un servidor, nomes haurem de inicialitzar el servidor i diversos clients amb les comandes indicades a la documentació de la practica.

En la figura següent, es pot observar el correcte funcionament del sistema.



The image shows three terminal windows from a 'tor@tor-virtual-machine' environment, demonstrating an Erlang/OTP chat system. The top-left window shows a client node starting and sending a 'secret' message. The top-right window shows another client node starting and sending a 'secret' message. The bottom window shows the server node starting and receiving messages from both clients. The server logs show '[JOIN] Riki joined the chat' and '[JOHN] holla', followed by '[Riki] gasff' and '[JOHN] holla'. The server also shows '[Riki] gasff' and '[JOHN] holla'.

```
tor@tor-virtual-machine: ~/Escritorio/sdx
tor@tor-virtual-machine:~/Escritorio/sdx$ erl -name client_node2@127.0.0.1 -setcookie secret
Erlang/OTP 20 [erts-9.3.3.3] [source] [64-bit] [smp:4:4] [ds:4:4:10] [async-thr
ads:10] [kernel-poll:false]

Eshell V9.3.3.3 (abort with ^C)
(client_node2@127.0.0.1)1> c(client).
(ok,client)
(client_node2@127.0.0.1)2> c!nt:start([myserver, 'server_node@127.0.0.1'
kt").
[JOIN] Riki joined the chat
-> gasff
[Riki] gasff
->
```

```
tor@tor-virtual-machine:~/Escritorio/sdx
tor@tor-virtual-machine:~/Escritorio/sdx$ erl -name client_node@127.0.0.1 -setcookie secret
Erlang/OTP 20 [erts-9.3.3.3] [source] [64-bit] [smp:4:4] [ds:4:4:10] [async-th
ads:10] [kernel-poll:false]

Eshell V9.3.3.3 (abort with ^C)
(client_node@127.0.0.1)1> c(client).
(ok,client)
(client_node@127.0.0.1)2> c!nt:start([myserver, 'server_node@127.0.0.1'
n").
[JOIN] John joined the chat
[JOHN] holla
[JOHN] holla
[Riki] gasff
->
```

```
tor@tor-virtual-machine:~/Escritorio/sdx
tor@tor-virtual-machine:~/Escritorio/sdx$ erl -name server_node@127.0.0.1 -setcookie secret
Erlang/OTP 20 [erts-9.3.3.3] [source] [64-bit] [smp:4:4] [ds:4:4:10] [async-thr
ads:10] [kernel-poll:false]

Eshell V9.3.3.3 (abort with ^C)
(server_node@127.0.0.1)1> c(server).
(ok,server)
(server_node@127.0.0.1)2> server:start().
true
(server_node@127.0.0.1)3>
```

Per fer-ho utilitzant diverses maquines físiques, es faria de la mateixa manera, però substituint la @IP de loopback, per el Hostname o @IP del node en qüestió.

2. Once you have a set of servers up and running, try connecting some clients to each of the server instances and begin to chat. Does it work? You can also try to crash some of the servers (e.g. by sending a disconnect message from the Erlang prompt) and see what happens. Al connectar diversos servidors i diversos usuaris a aquets, el chat segueix funcionant perfectament, de forma que encara estan en servidors diferents, tots els clients de tots els servidors podran veure els missatges de la resta. Tal i com es pot veure en la figura següent:

```
tor@tor-virtual-machine: ~/Escritorio/sdx
tor@tor-virtual-machine:~/Escritorio/sdx$ erl -name client_node@127.0.0.1
Eshell V9.3.3.3 (abort with ^G)
(client_node@127.0.0.1)> c(client).
* 2: syntax error before: c
(client_node@127.0.0.1)> c(client).
(ok,client)
(client_node@127.0.0.1)> client:start(myserver, 'server_node@127.0.0.1')
ard).
[JOIN] Ricard joined the chat
-> hola
[Ricard] hola
[John] k tl ?
[JOHN] Carles joined the chat
[Carles] hola
[John] com estas carles ?
[JOHN] jhon joined the chat
[jhon] hola
[jhon] com esteu ?
-> []
(client_node@127.0.0.1)> client:start(myserver, 'server_node@127.0.0.1')
server_node@127.0.0.1: "Carles".
* 1: syntax error before: '.'
(client_node@127.0.0.1)> client:start(myserver, 'server_node@127.0.0.1')
hon").
[JOIN] jhon joined the chat
-> hola
[jhon] hola
-> com esteu ?
[jhon] com esteu ?
-> []

tor@tor-virtual-machine:~/Escritorio/sdx
tor@tor-virtual-machine:~/Escritorio/sdx$ erl -name server_node2@127.0.0.1 -setc
cookie secret
Erlang/OTP 20 [erts-9.3.3.3] [source] [64-bit] [smp:4:4] [ds:4:4:10] [async-thr
ads:10] [kernel-poll:false]
Eshell V9.3.3.3 (abort with ^G)
(server_node2@127.0.0.1)> c(server2).
(ok,server2)
(server_node2@127.0.0.1)> server2:start(myserver, 'server_node@127.0.0.1').
true
(SERVER UPDATE) [+0.73.0s, <10655.73.0s>]

tor@tor-virtual-machine:~/Escritorio/sdx
tor@tor-virtual-machine:~/Escritorio/sdx$ erl -name client_node3@127.0.0.1 -setc
cookie secret
Erlang/OTP 20 [erts-9.3.3.3] [source] [64-bit] [smp:4:4] [ds:4:4:10] [async-thr
ads:10] [kernel-poll:false]
Eshell V9.3.3.3 (abort with ^G)
(client_node3@127.0.0.1)> c(client).
(ok,client)
(client_node3@127.0.0.1)> client:start(myserver, 'server_node@127.0.0.1'), "Ca
rles").
[JOIN] Carles joined the chat
-> hola
[Carles] hola
[John] com estas carles ?
[JOIN] jhon joined the chat
```

El tenir més d'un servidor, també aporta robustes al sistema, ja que si un dels servidors cau, només es queden sense connexió els clients que estan connectats en aquest servidor en específic. Tal i com es pot apreciar en les figures següents:

The image displays two terminal windows from a Tor virtual machine, illustrating the robustness of a multi-server system. The left window shows the client-side interaction where users 'Rit', 'carles', and 'John' join a chat and exchange messages. The right window shows the server-side logs, including the start of the server, client connections, and the receipt of messages. The interface includes a sidebar with application icons and a top bar with system information.

```

tor@tor-virtual-machine: ~/Escritorio/sdx
tor@tor-virtual-machine:~/Escritorio/sdx$ erl -name client_node@127.0.0.1 -setco
okie secret
Erlang/OTP 20 [erts-9.3.3.3] [source] [64-bit] [smp:4:4] [ds:4:4:10] [async-thre
ads:10] [kernel-poll:false]

Eshell V9.3.3.3 (abort with ^G)
(client_node@127.0.0.1)> c(client).
(ok,client)
(client_node@127.0.0.1)> client:start(myserver, 'server_node@127.0.0.1'), "Rit
").
[JOIN] Rit joined the chat
[JOIN] carles joined the chat
[carles] hola
-> com estas
[Rit] com estas
[John] be i tu
->

tor@tor-virtual-machine:~/Escritorio/sdx
tor@tor-virtual-machine:~/Escritorio/sdx$ erl -name client_node2@127.0.0.1 -setc
okie secret
Erlang/OTP 20 [erts-9.3.3.3] [source] [64-bit] [smp:4:4] [ds:4:4:10] [async-thre
ads:10] [kernel-poll:false]

Eshell V9.3.3.3 (abort with ^G)
(client_node2@127.0.0.1)> c(client).
(ok,client)
(client_node2@127.0.0.1)> client:start(myserver, 'server_node@127.0.0.1'), "Jo
hn").
[JOIN] John joined the chat
[JOIN] Rit joined the chat
[JOIN] carles joined the chat
[carles] hola
tor@rit] com estas
okie-> be i tu
Erl@John] be i tu
ads:->

Eshe
(ser
(ok,
(ser
True
[SERVER UPDATE] [<0.73.0>,<0.76.0>]

tor@tor-virtual-machine:~/Escritorio/sdx
tor@tor-virtual-machine:~/Escritorio/sdx$ erl -name client_node3@127.0.0.1 -setc
okie secret
Erlang/OTP 20 [erts-9.3.3.3] [source] [64-bit] [smp:4:4] [ds:4:4:10] [async-thre
ads:10] [kernel-poll:false]

Eshell V9.3.3.3 (abort with ^G)
(client_node3@127.0.0.1)> c(client).
(ok,client)
(client_node3@127.0.0.1)> client:start(myserver, 'server_node2@127.0.0.1'), "c
arles").
[JOIN] carles joined the chat
[carles] hola
[rit] com estas
[John] be i tu
-> hola
[carles] hola
[rit] com estas
[John] be i tu
-> hola
->

tor@tor-virtual-machine:~/Escritorio/sdx
tor@tor-virtual-machine:~/Escritorio/sdx$ erl -name server_node2@127.0.0.1 -s
etco
okie secret
Erlang/OTP 20 [erts-9.3.3.3] [source] [64-bit] [smp:4:4] [ds:4:4:10] [async-thre
ads:10] [kernel-poll:false]

Eshell V9.3.3.3 (abort with ^G)
(node2@127.0.0.1)> c(server2).
2)
(node2@127.0.0.1)> server2:start(myserver, 'server_node@127.0.0.1').
[DATE] [<0.73.0>,<0.76.0>]
(node2@127.0.0.1)>

tor@tor-virtual-machine:~/Escritorio/sdx
tor@tor-virtual-machine:~/Escritorio/sdx$ erl -name client_node3@127.0.0.1 -setc
okie secret
Erlang/OTP 20 [erts-9.3.3.3] [source] [64-bit] [smp:4:4] [ds:4:4:10] [async-thre
ads:10] [kernel-poll:false]

Eshell V9.3.3.3 (abort with ^G)
(client_node3@127.0.0.1)> c(client).
(ok,client)
(client_node3@127.0.0.1)> client:start(myserver, 'server_node2@127.0.0.1'), "c
arles").
* 1: illegal character
(client_node3@127.0.0.1)> client:start(myserver, 'server_node2@127.0.0.1'), "c
arles").
[JOIN] carles joined the chat
-> hola
[carles] hola
[rit] com estas
[John] be i tu
-> hola
->

tor@tor-virtual-machine:~/Escritorio/sdx
tor@tor-virtual-machine:~/Escritorio/sdx$ erl -name server_node2@127.0.0.1 -s
etco
okie secret
Erlang/OTP 20 [erts-9.3.3.3] [source] [64-bit] [smp:4:4] [ds:4:4:10] [async-thre
ads:10] [kernel-poll:false]

Eshell V9.3.3.3 (abort with ^G)
(node2@127.0.0.1)> c(server2).
server2
(node2@127.0.0.1)> server2:start(myserver, 'server_node@127.0.0.1').
[UPDATE] [<0.73.0>,<0.76.0>]
(node2@127.0.0.1)> q().
ok

```

3 Open questions

1. Does this solution scale when the number of users increase?
No, si el número d'usuaris incrementa. Un únic servidor no podrà mantenir el mateix rendiment i la qualitat del servei baixarà, ja que pot haver-hi saturació i coll d'ampolla.
2. What happens if the server fails?
Aquest sistema només consta d'un únic servidor per gestionar els missatges, per tant, si el servidor falla, tot el sistema fallarà. No hi ha resiliència al sistema.
3. Are the messages from a single client guaranteed to be delivered to any other client in the order they were issued?
Sí perquè hi ha un únic servidor que reenviarà als clients els missatges en l'ordre que els rebi.
4. Are the messages sent concurrently by several clients guaranteed to be delivered to any other client in the order they were issued?
Erlang no garanteix l'ordre dels missatges, ja que això comportaria una sèrie de mecanismes i controls que el llenguatge no adopta.
5. Is it possible that a client receives a response to a message from another client before receiving the original message from a third client?
Com els clients estan connectats al mateix servidor, no seria possible aquest esdeveniment, ja que com que tampoc hi ha cap mena de retenció client-server, els missatges haurien d'arribar en el mateix ordre en què van ser enviats.
6. If a user joins or leaves the chat while the server is broadcasting a message, will he/she receive that message?
No, perquè el servidor per enviar missatges ha de registrar prèviament tots els clients que s'uneixen, i no ho pot fer a l'hora que fa un broadcast.

Making it robust

1. What happens if a server fails?

Els clients que estiguin connectats al servidor caigut es desconnectaran del xat. Per poder reincorporar-se hauran de connectar-se a l'altre servidor existent.

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

Respecte les preguntes 3 i 4 de l'apartat anterior, no es pot assegurar en Erlang el correcte ordre dels missatges entre dos nodes si existeix més d'un procés. Respecte la pregunta 5, es pot donar el cas que descriu l'enunciat si la latència del missatge de resposta és inferior a la latència del missatge inicial. Per exemple: Un client A i un client B estan connectats a un servidor S1 i un client C (connectat a un servidor S2) envia un missatge inicial que serà respost per B. El client A (més proper a B que a C) veuria abans la resposta (feta per B) que el missatge inicial (fet per C) si la latència de C és superior que a la del seu company B.

3. What might happen with the list of servers if there are concurrent requests from servers to join or leave the system?

La llista dels servidors s'aniria modificant al moment en que un servidor forma part del grup de servidors actius o al moment en que un servidor es desconnecta del grup. Això no suposaria cap inconvenient ja que la llista està capacitada per soportar variacions. Un únic problema que podria aparèixer seria la possible saturació de la xarxa de servidors i clients si les peticions fossen moltes i molt concurrents, però això ja depèn de les característiques d'aquesta. En cas d'aquest problema, s'ignorarien les peticions de unir-se i/o marxar i la llista continuaria com anteriorment.

4. What are the advantages and disadvantages of this implementation regarding the previous one? (compare their scalability, fault tolerance, and message latency).

Aquest sistema presenta més escalabilitat que l'anterior ja que a l'existir diversos servidors, la càrrega de treball entre els diferents nodes es pot repartir i així s'evita que un servidor realitzi tota la càrrega del servei. La tolerància a fallades incrementa considerablement, ja que no es depèn d'un sol servidor encarregat de la comunicació entre tots els clients, sinó que són diversos els encarregats a transmetre els missatges entre els nodes i si algun d'aquests falla, pot existir la possibilitat d'anar per un camí alternatiu o en el cas pitjor deixar desconnectats als clients d'aquell únic servidor. La latència del missatge és variable i depèn de la proximitat física entre el servidor i un client. Dos clients connectats a un mateix servidor tindran una latència més baixa que si un client i un altre client a un altre servidor s'intercanvien missatges. Tot i així, l'existència de més d'un servidor per on poder transportar els missatges provoca un alliberament de tràfic de la xarxa que pot reduir la latència efectiva del sistema.

4 Personal opinion

L'implementació d'un xat com a primera pràctica de l'assignatura fa que es pugui apreciar, de manera interactiva i amb exemples, com funcionen a petita escala les xarxes distribuïdes i els seus principals aspectes i avantatges de la seva utilització, així com el paper de client-servidor i tots els passos que segueixen per poder-se comunicar.