Seminar Report: Chatty

Lluís Marquès, Marc Catrisse, Èric Casanovas

March 6, 2019

Upload your report in PDF format.
Use this LaTeX template to format the report.
A compressed file (.tar.gz) containing all your source code files must be submitted together with this report[1].

1 Introduction

Introduce in a couple of sentences the seminar and the main topic related to distributed systems it covers.
En aquesta pràctica hem d'implementar un sistema distribuit que permeti xatejar amb 2 o més clients. L'idea del seminari és la d'experimentar amb els problemes típics dels sistemes distribuits i veure diferents estrategies per evitar-los. A la primera part vam implementar un sistema distribuit compost per un únic servidor, seguit d'una segona implementació més robusta que incloeix un segon servidor.

2 Experiments

2.1 Part 1

El servidor del nostre sistema distribuit el servidor és l'encarregat d'enregistrar els usuaris que estàn connectats i reenviar els missatges dels clients a la resta. En el cas del client aquest té dues tasques a realitzar:

- La primera, en segon pla, que s'encarrega de gestionar els missatges enviats pel servidor. Aquests poden ser events (un usuari es connecta al chat) o missatges d'altres clients.
- La segona consisteix en llegir el teclat i enviar missatges al servidor.

Primer vam fer una versió local on teniem 2 clients i un servidor, que s'executaven al mateix ordinador. Posteriorment vam executar el mateix codi en 2 ordinadors diferents, un amb el servidor i el client 1 i l'altre amb el client 2. Per fer-ho vam haver d'indicar els ports a utilitzar(del 1025-2000) per erlang en el moment d'executar el runtime.

```
erl -kernel inet_dist_listen_min 1025 -kernel inet_dist_listen_max 2000
```

[1] Describe in the report any design decision required to understand your code (if any)

Screenshot d'un client connectat al servidor en local

Screenshot d'un segon client connectat al servidor en local

Screenshot d'un client connectat al servidor en xarxa

Screenshot d'un segon client connectat al servidor en xarxa

2.2 Part 2

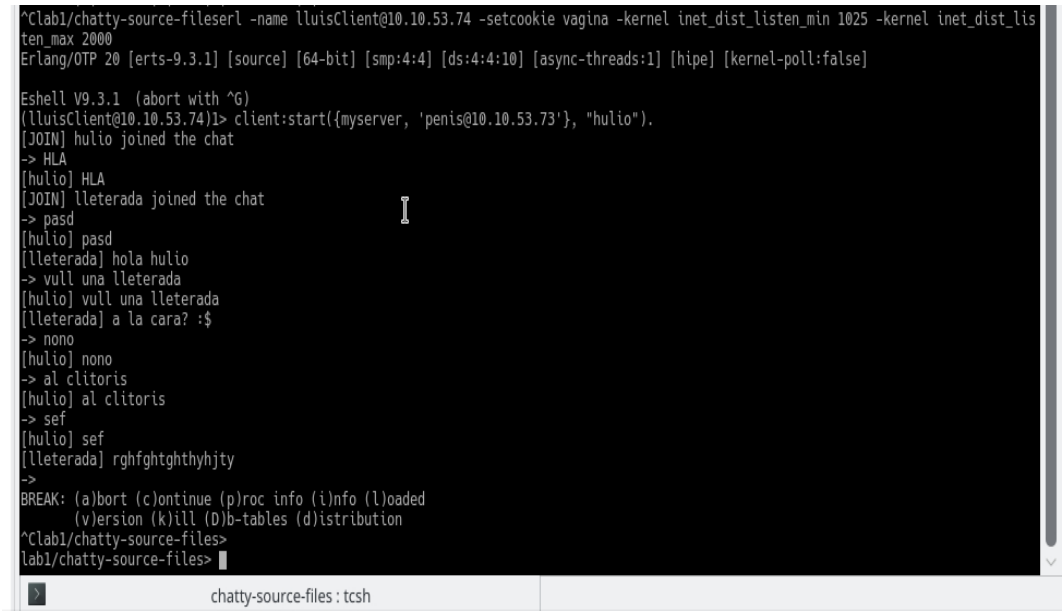
En aquest cas disposem de 2 servidors. Ambdós estableixen una connexió que els permet comunicar-se amb els clients de l'altre, d'aquesta manera, si cau un d'ells, els clients d'un dels extrems es manté i no es penja el sistema distribuit sencer.
Els clients operen de la mateixa manera que a la primera part.
Els servidors però, tenen dues maneres de fer start(). Un d'ells farà:

```
server2:start().
```

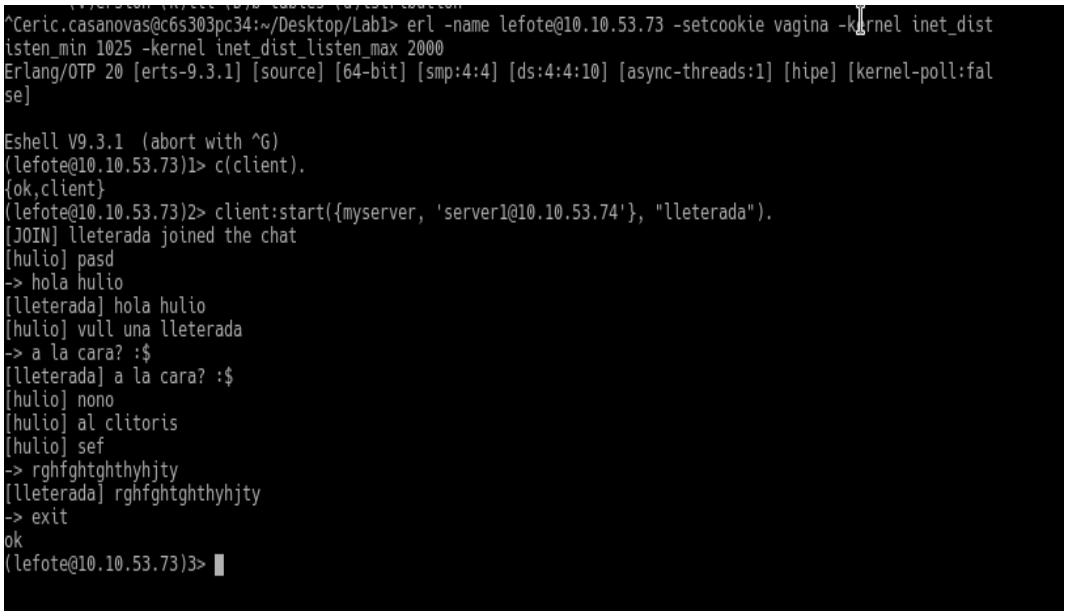
L'altre però, haurà de fer, en el nostre cas:

```
server:start({myserver, 'nom_server2@10.10.53.73}).
```

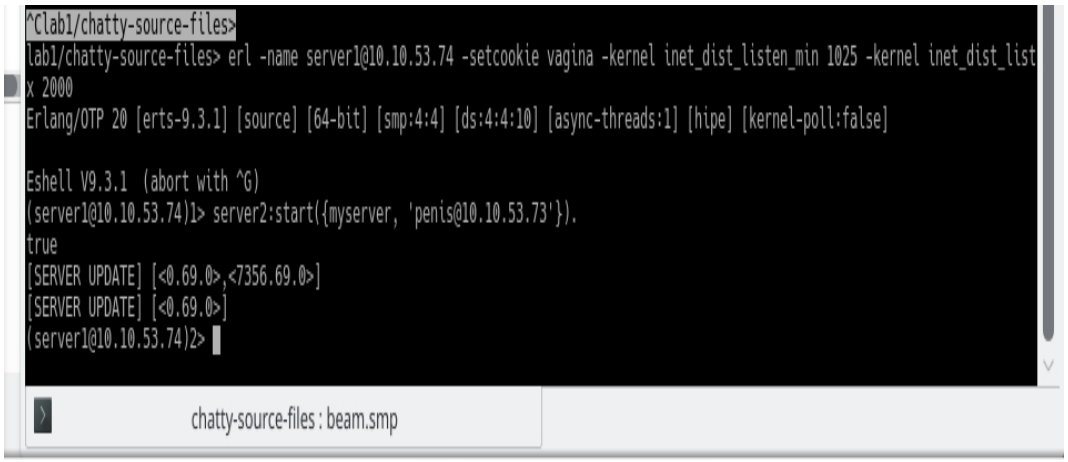
D'aquesta manera vinculem els dos servidors i poden passar missatges d'un a l'altre.
Al treballar en màquines diferents, obviament, hem hagut de tornar a activar els ports [1025, 2000] amb l'opció vista a la part 1.
A les captures següents podem observar un extracte de com funciona aquest sistema distribuit.



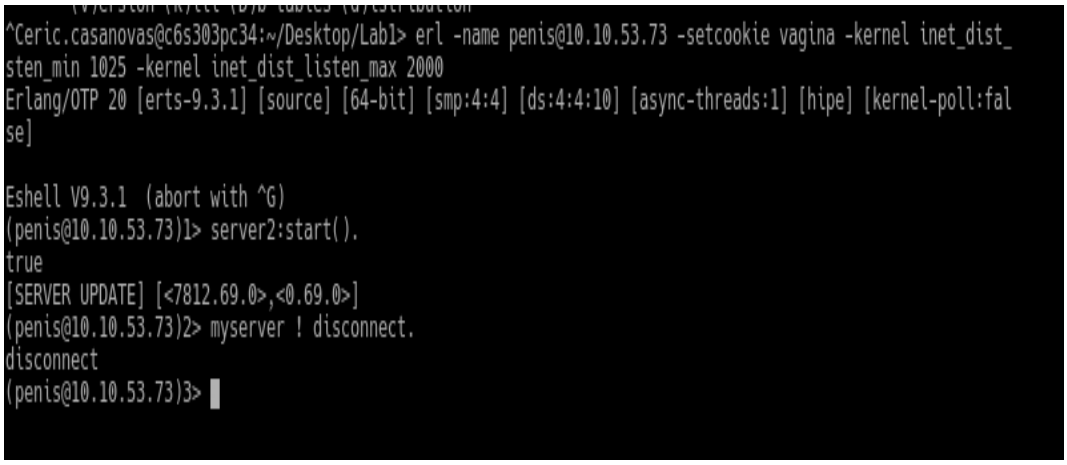
Client 1: connectat al servidor2, situat a un pc remot



Client 2: connectat al servidor 1, situat a un pc remot



Server 1: Realitza un start vinculat al servidor 2



Server 2: Rep la connexió del Server 1. Posteriorment realitza un "disconnect"

3 Open questions

3.1 Part 1

(i) Does this solution scale when the number of users increase?
Sí, podem afegir els usuaris que volguem, però limitat per la capacitat del servidor.

(ii) What happens if the server fails?
No és notifica la fallada de servidor. El sistema es torna inútil i els clients no poden enviar ni rebre missatges.

(iii) Are the messages from a single client guaranteed to be delivered to any other client in the order they were issued?
Sí, es una de les característiques d'Erlang.

(iv) Are the messages sent concurrently by several clients guaranteed to be delivered to any other client in the order they were issued?
El servidor tracta els missatges per FIFO, el primer que rep, és el primer que enviarà per broadcast.No podem assegurar l'ordre d'arribada dels missatges dels clients, ja que provenen de diferents nodes.

(v) Is it possible that a client receives a response to a message from another client before receiving the original message from a third client?
No, es garanteix l'ordre servidor - client. Donat que el servidor primer enviarà el missatge i després la resposta, es garanteix l'ordre dels missatges.

(vi) If a user joins or leaves the chat while the server is broadcasting a message, will he/she receive that message?
No, el pool de Clients del servidor on es fa el broadcast no inclou el client nou que acaba de fer join.

3.2 Part 2

(i) What happens if a server fails?
Quan un dels servidors falla, els seus clients perden la connexió, però els

clients de l'altre servidor segueixen funcionant correctament. Si els clients del servidor caigut volen seguir formant part de la conversa hauràn de connectar-se al servidor actiu.

(ii) Do your answers to previous questions iii, iv, and v still hold in this implementation?
Respecte la iii no, degut a que podria haver una fallada d'un servidor i fer que no arribés el missatge al client remot.
Pel que fa la iv, tampoc tenim un ordre garantit degut a que en cas de tenir varis servidors, és possible que un servidor, per proximitat/latencia/altres factors, capturi abans els missatges que provenen d'una banda i no es correspongui en l'ordre original dels missatges.
Per últim, en cas de join, sí, és possible. Ja que suposant que hi ha un mínim de delay en la comunicació, el join del client es capturaria abans d'arribar el broadcast del missatge d'un node remot, fent que el servidor on es connecta el nou client faci broadcast amb la llista de clients actualitzada. En cas de leave, el client que marxa no rebria el missatge.

(iii) What might happen with the list of servers if there are concurrent requests from servers to join or leave the system?
Cada un dels servidors rebrà un missatge amb que algú ha entrat o sortit del sistema.

(iv) What are the advantages and disadvantages of this implementation regarding the previous one? (compare their scalability, fault tolerance, and message latency)
La segona implementació té més tolerancia a les fallades que la primera, degut a que si un servidor cau no cau tot el sistema. Això fa que sigui més escalable i tolerant a les fallades. Pel que fa a la latència, si tinguessim molts nodes i servers això provocaria que s'enviessin molt missatges d'actualització i podria provocar un augment de la latència a la comunicació.

4 Personal opinion

Ens ha semblat un seminari força interessant on aprenem a fer clients i servidors amb erlang sobre les màquines del laboratori. Pensem que es un laboratori amb una dificultat adecuada i no creiem que s'hagi d'afegir res o treure res en pròxims cursos.