Concurrent programming

BENZA Amandine – FORNALI Damien

Chat Server in Elixir

Architecture

In order to achieve the chat server, we used the following approach.

When the server main processor [**MP**] launches, it creates a new processor that will update the users idle counter [**IDLE_UPDATER**].

After that, it generates a listener in order to listen to new user connections [**LISTENER**] and start a unique processor for user acceptation [**ACCEPTOR**].

ACCEPTOR will checks the user inquired pseudo and if it is wrong, it will ask to the user to retry. If the inquired pseudo is valid, a new processor responsible of the newly connected user will be generated [**USER_PROC**].

MP will then waits for signals and processing according what it receives. For example if **MP** receives a « **broadcast** » signal, it means it has to send to all users a message.

Please, see next page for an architecture overview schema.

Communication

And this is how we perform communication between processors. Only MP stores the users in his memory. Other USER_PROCs don't know the existence of each other. To be able to communicate between them, USER_PROCs have to send a signal to MP in order to perform a certain operation, for instance to communicate messages.

Data approach

A user is represented by the following structure.

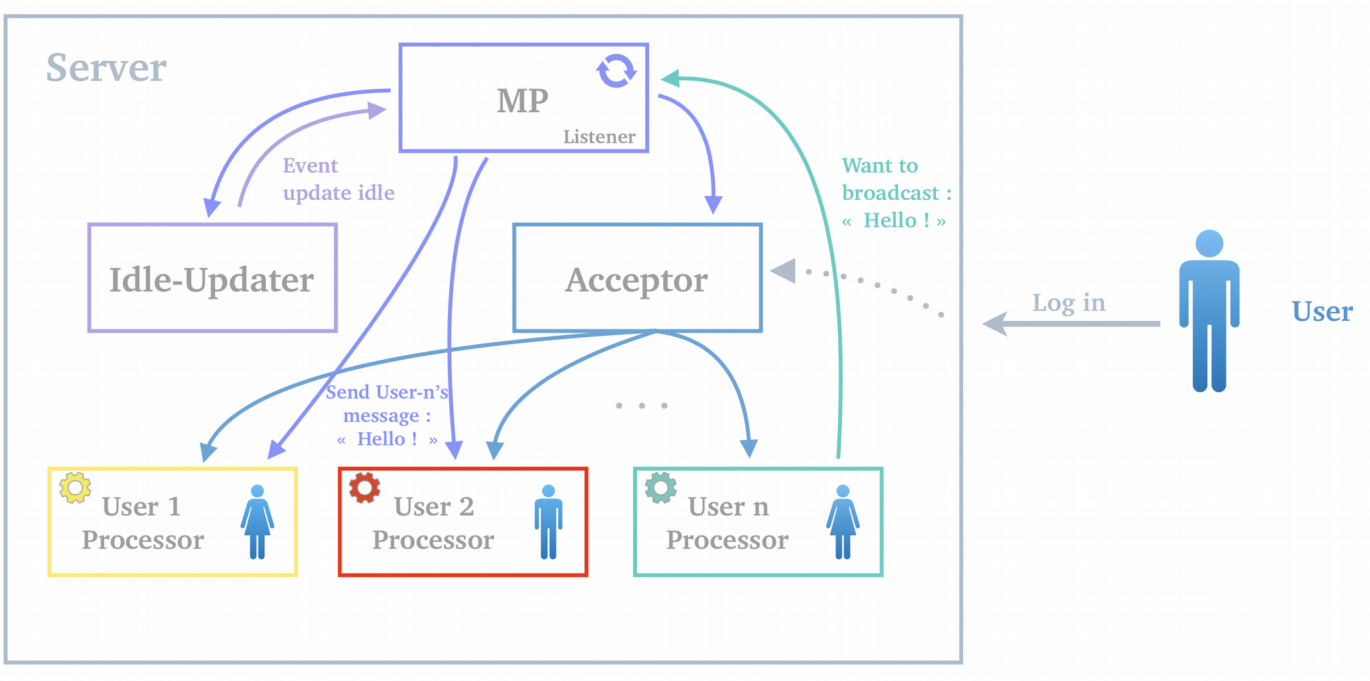
```
defmodule User do
    defstruct socket: nil, pseudo: nil, pid: -1, idleTime: 0
end
```

It allows us to keep trace of user information, such as pseudo and associated processor identifier. We also use a Timeout module in order to keep trace to the maximum authorized idle time.

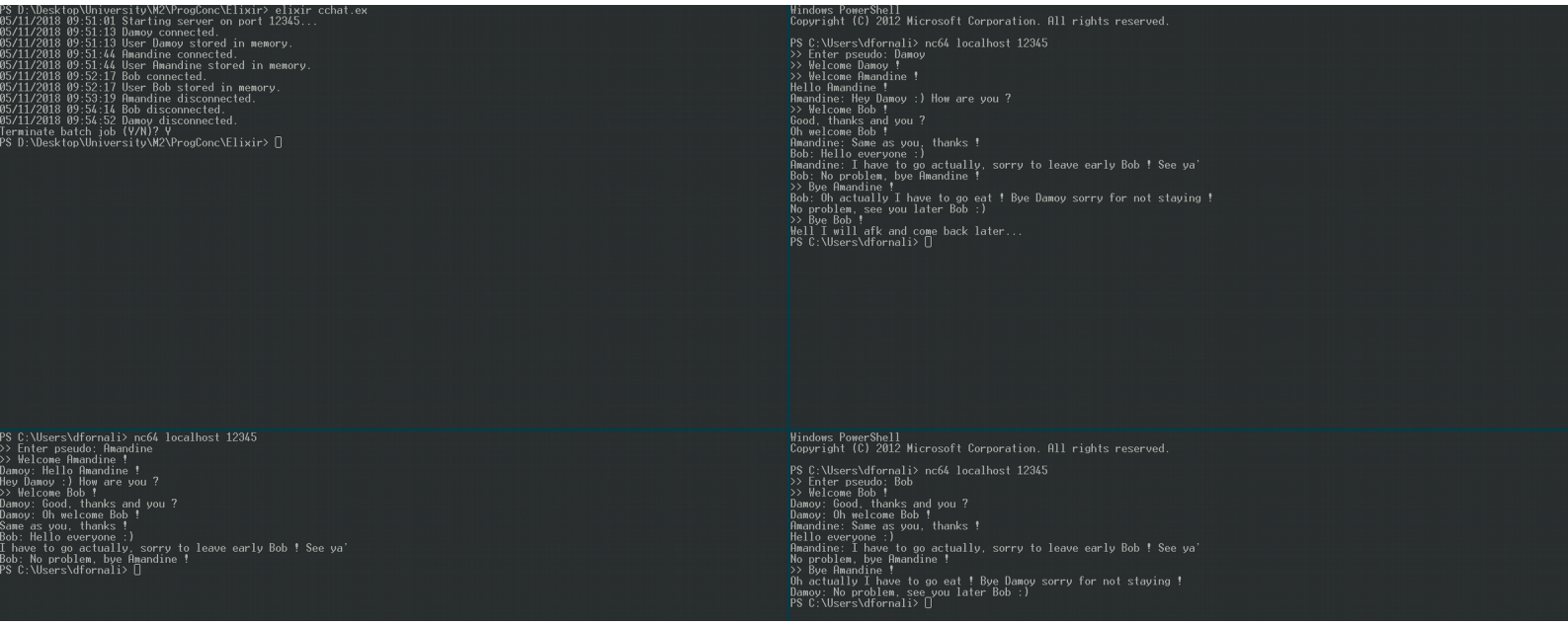
Difficulties

It was quite difficult to create a nice architecture using *Elixir*'s syntax. Making use of Elixir structures has also been difficult when handling idle timeout force disconnections. We tried multiple approaches (enums, agents, events communication...) but encountered difficulties for update user idles. After all our trials we now have one solution but it would require to refactor our code by using agents instead of only enums. We chose to keep our solid basis and not take the risk of adding a bug, taking in account the time left.

Architecture Overview



Scenario



How to launch

Launch server : sh ./server.sh

Connect a new client : sh ./client.sh