Disclaimer

Today’s session is only to be done by those who have completed the previous one. If you haven’t finished implementing the sliding window algorithm, please do that first. Make sure to implement the simulation of failures part as well.

Topic

In this session we will discuss about the deployment of Erlang programs. So far, you saw how to spawn processes in an Erlang context. You also saw how to have those processes communicate with each other. However, building complex distributed systems requires much more.

Today, you will have several increasingly difficult goals. The objective here is not only to implement simple features, but also to think about the technical elements that have to be implemented to achieve the more difficult goals.

Question 1

a) Calling erlang:node/0 gives me nonode@nohost. What does that mean? Why is node naming important? How to solve that?

b) Try running two Erlang contexts on your machine. Use net_adm:ping/1 to check if the connection can be established. You should get a pong, if you get a pang then something is not working properly.

c) Using SSH, try running two Erlang contexts on two different machines. Can you establish a connection? If not try using a full name to solve the DNS issues.

d) Have a look at the functions erlang:register/2, erlang:registred/0, erlang:whereis/1 and erlang:unregister/1. What do you think they are used for?

e) [hard] Have a look at erlang:spawn/4 and experiment with it. What do you think of its behaviour?

f) [hard] How does erlang:spawn/4 handles custom modules? Discuss about a bootstrapping method to send code to remote servers.

g) [harder] Implement everything needed to compile code, launch and register processes remotely. You should make sure that all cases are handled (unregister a name, attempt to register a name that is already used, etc.). You should start by defining the communication interface.

h) [hardest] Deploy your code and show how it can be used.