IN.5022 — Concurrent and Distributed Computing

Series 3 and 4

Due dates: 12.10.2023 (soft deadline) and 19.10.2023, 12:00, on Moodle

Exercise 1 (due 19.10.2023, feedback if provided by 12.10.2023)

Write an Erlang (or Elixir) program, based on your pseudo code from the previous exercise series, that creates a ring of *N* processes and sends *M* times a token with an associated counter in the ring.

The following constraints must be fulfilled:

1. All ring processes have exactly the same code (within the module `tr`), except the starting process *s*: `tr:start(N,M)` that launches the application.
2. Each time a process *p* receives a token, *p* sends the associated counter incremented by one to its successor. It also prints the value of the counter.
3. When there are no more tokens, each process terminates gracefully. Optionally have each node write the last value of the counter into a file, together with its process identifier and the identifier of its successor in the ring.

Hints:

- Only simple data structures (atoms, integers, tuples and simple variables) are necessary in this exercise. Also, only simple arithmetic, receive and send expressions, elementary pattern matching and tail recursion are necessary: no case or if expressions. Even so, it is a quite challenging exercise for a first Erlang (or Elixir) program. Pay special attention to having a functional and declarative approach, and not a procedural one.
- The easiest way to deploy the token ring is to do it with an extra process that will start the other processes. Another way is that all processes in the ring act as peers and spawn their successor except for the last one that closes the ring.
- If the constraints are too difficult to be satisfied, you can relax them to come up with a working solution.
- As a step-by-step process, follow these points in the creation of your program to make your life easier:
    1. Make a sketch of your processes and communication between them, clearly separating the phases (e.g., deployment, token activity, termination). This is your documentation.
    2. Write a high-level language-independent pseudo code (based on the previous exercise series).
    3. Write a (high-level) pseudo code.

Exercise 2 (due 19.10.2023)

Write the same program in the other language (Elixir or Erlang) than you used for the first exercise.