POLITECNICO
MILANO 1863

Actor Model – Akka Evaluation Lab

Luca Mottola

luca.mottola@polimi.it

http://mottola.faculty.polimi.it

Rules

- Complete the README.md file with
  - Your group identifier
    - From the group registration document
  - Name of each group member
  - A 200-word (max) description of the message flows in your solution
    - What actor talks to what other actor using what message, when, …
- Create and submit a single zip file with the entire code of your project
  - Name of the file: akka-groupXX.zip
    - XX is the group identifier from the group registration document
  - Submit by the user corresponding to the contact email specified in the group registration document

Preliminaries

- You are to create a simple **sensor data processing** system using actors
- The system shall be composed of (at least) three different actor types
  - A variable number of **sensor data processors** that maintain the average of the sensor readings they receive
  - A single **dispatcher** actor that distributes incoming sensor readings according to two different policies (see next)
  - A variable number of **temperature sensor** actors that generate temperature readings

Dispatching Logic: Load Balancing

- The dispatcher permanently associates a given temperature sensor to a given sensor data processor
- It maintains the number of temperature sensors funneling data to the same sensor data processor as balanced as possible
  - Example: with five temperature sensors (A, B, C, D, E) and two sensor data processors (P1, P2)
    - C, B, E → P1; A, D → P2 is acceptable
    - C, B, A, E → P1; D → P2 is **not** acceptable

Dispatching Logic: Round Robin

- The dispatcher decides on an (any) ordering among data processors
- As a temperature reading is received, the dispatcher forwards the reading to the next sensor data processor in a circular order
  - Example: with five temperature readings coming in at time t0, t1, t2, t3, t4 and three sensor data processors
    - Say the ordering is P1, P2, P3
    - t0 → P1; t1 → P2; t2 → P3; t4 → P1 is acceptable
    - t0 → P1; t1 → P2; t2 → P1; t4 → P3 is **not** acceptable

Message Types

- The system shall use (at least) three types of messages
  - **TemperatureMsg** carries temperature readings
  - **DispatchLogic** tells the dispatcher what dispatching logic to use between Load Balancing and Round Robin
  - **GenerateMsg** tells the temperature sensors to generate a new reading

Failures

- When a sensor data processor receives a negative temperature reading, **it fails**
- Handling the failure must ensure that the average temperature before the failure is **retained**, yet the faulty temperature reading is excluded from the computation

Code

- In the assignment, you also find
  - A definition of the three basic message types
    - You are free to **extend** the message definitions, but you **cannot change** the existing code
    - You can of course define more message types, if needed
  - Two **working actors** corresponding to the (faulty) temperature sensors
    - You should not normally change the code here, but you may need to extend it
  - Templates for the **dispatcher** and the **processor** actors
  - A **template** for a test main method
    - This is necessarily incomplete!!
    - It cannot run as it is!!
    - It must run when you submit