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 **event-based communication system** using actors
- The system shall be composed of (at least) five actors
  - Two **worker** actors matching events against subscriptions
  - A **broker** actor coordinating the operation of **worker**(s)
  - A **subscriber** actor that issues subscriptions and receives notifications
  - A **publisher** actor that generates events

Preliminaries (1/2)

- The system shall use (at least) four types of messages
  - **SubscribeMsg** to issue subscriptions
  - **PublishMsg** to generate events
  - **NotificationMsg** to notify subscribers of events
  - **BatchMsg** to change the broker message policy

Brokers and Workers

- The **broker** splits the matching process between the two **worker** actors as follows
  - Splitting is based on a partitioning of the **key** attribute in the **Subscribe** message
    - Even keys go to one worker, odd keys go to the other
  - When a worker actor receives a **Publish** message for a topic it is not aware of, it fails
    - Handling the failure must ensure that the set of active subscriptions before the failure is retained
  - You can assume that **at most one subscriber exist for a given topic**

BatchMsg

- Whenever the broker receives a BatchMsg, it looks at the **isOn** attribute
  - If it is false, the broker shall immediately process every subsequent message it receives
  - If it is true, the broker shall buffer all event messages it receives since that time, and process them in a batch as soon as it receives another **BatchMsg** with **isOn** set to false
- You may begin solving this exercise without this feature, then you add it later on

Code

- In the assignment, you also find
  - A definition of the four 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
  - A **template** for a test main method
    - This is necessarily incomplete!!
    - It cannot run as it is!!
    - Uncomment the Java lines to test
    - It must run when you submit