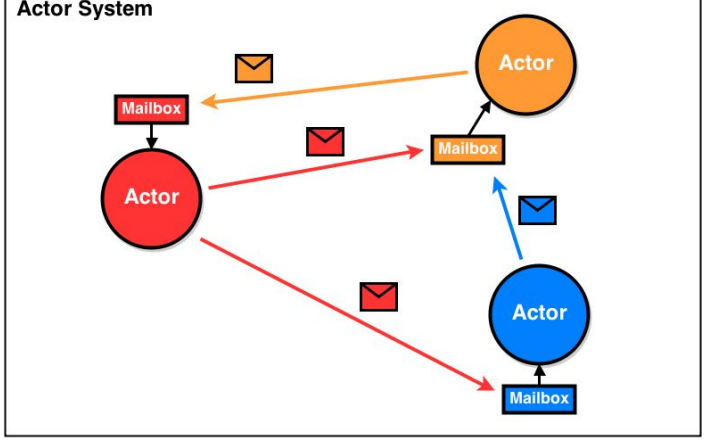
Programmation distribuée par acteur

Par ou commencer ?

Akka is a collection of tools that helps you write concurrent and distributed applications. The central piece is its implementation of the Actor Model, invented by Carl Hewitt in 1973, where Actors are independent agents of computation that only communicate with each other by sending messages

Les acteurs ?

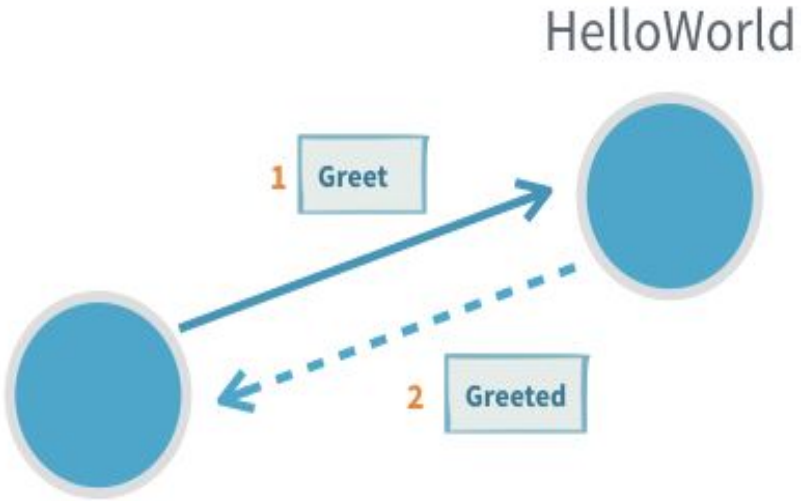


- un modèle de programmation concurrent
- la charge de travail est répartie entre des entités, les acteurs
- les acteurs sont isolés, pas d'état partagé
- éléments légers
- Communication sous forme de messages
- abstraction des verrous et threads = moins complexe pour développer

Akka ?

- framework pour la construction de systèmes distribués
- tolérance au panne = “ **let it crash** “
- Supervision : chaque acteur est supervisé par son parent (**auto-réparant**)
- acteurs asynchrones , relocalisation possible
- APi en java , Scala ...

Envoyer et Recevoir



- **ActorRef** ≈ Pointeur
- **Tell**(Object message, ActorRef sender)

  (non bloquante)

- On peut forwarder un msg
- **Receive** , reçoit un msg et adopte un comportement
- messages acceptés + type de réponse= protocole de l'acteur

Qu’en est t-il des données ?

- Les données échangés devraient être immuables
- Messages traités pas forcément dans le même thread
- Les opérations bloquantes sont à proscrire
- un thread à la gestion de ressources bloquantes

Garanties de livraison

- **Au plus une fois** 

- **Au moins une fois**
- **Exactement une fois** (coûteux)

Gestions des erreurs

- Chaque acteur est le superviseur de ses acteurs fils
- Remonté de message système
- Agit selon une stratégie de supervision

-> peut agir sur un fils ou tous les fils

-> Par défaut:

- ***ActorInitializationException, ActorKilledException***: l'acteur est stoppé.
- ***Exception*** : l'acteur est redémarré
- ***autres Throwable*** : ils sont remontés au parent