



Actors are Distributed (Part II)

Programming Reactive Systems

Roland Kuhn

Cluster needs Failure Detection

Consensus is unattainable if some members are unreachable.

Every node is monitored using heartbeats from several others.

A node unreachable from one other is considered unreachable for all.

Nodes can be removed to restore the cluster consensus.

Cluster and DeathWatch

Actors on nodes which are removed from the cluster must be dead.

- ▶ allows clean-up of remote-deployed child actors
- ▶ decision must be taken consistently within the cluster
- ▶ once Terminated was delivered the actor cannot come back

Lifecycle monitoring is important for distributed fail-over:

- ▶ delivery of Terminated is guaranteed
- ▶ this is only possible because it can be synthesized when needed

Applying it to ClusterWorker

```
val cluster = Cluster(context.system)
cluster.subscribe(self, classOf[ClusterEvent.MemberUp])
val main = cluster.selfAddress.copy(port = Some(2552))
cluster.join(main)

def receive = {
  case ClusterEvent.MemberUp(member) =>
    if (member.address == main) {
      val path = RootActorPath(main) / "user" / "app" / "receptionist"
      context.actorSelection(path) ! Identify("42")
    }
  case ActorIdentity("42", None)      => context.stop(self)
  case ActorIdentity("42", Some(ref)) => context.watch(ref)
  case Terminated(_)                => context.stop(self)
}
```