Akka Java Documentation

Release 2.2.3

Terminology, Concepts

- *Concurrency vs. Parallelism*
- *Asynchronous vs. Synchronous*
- *Non-blocking vs. Blocking*
- *Deadlock vs. Starvation vs. Live-lock*
- *Race Condition*

Non-blocking Guarantees (Progress Conditions)

Non-blocking Guarantees (Progress Conditions)

- *Wait-freedom*

```
public void wait_free_method ()
{  // Every call takes
   // finite number of steps
   --> Never blocking (No deadlocks)
   --> No starvation
}
```

Non-blocking Guarantees (Progress Conditions)

- *Wait-freedom*

```
public void wait_free_method ()
{  // Every call takes
   // finite number of steps
   --> Never blocking (No deadlocks)
   --> No starvation
}
```

- *Lock-freedom*

```
public void lock_free_method ()
{  // Often calls take
   // finite number of steps
   --> No deadlocks
   --> Starvation possible
}
```

Non-blocking Guarantees (Progress Conditions)

- *Wait-freedom*

```
public void wait_free_method ()
{  // Every call takes
   // finite number of steps
   --> Never blocking (No deadlocks)
   --> No starvation
}
```

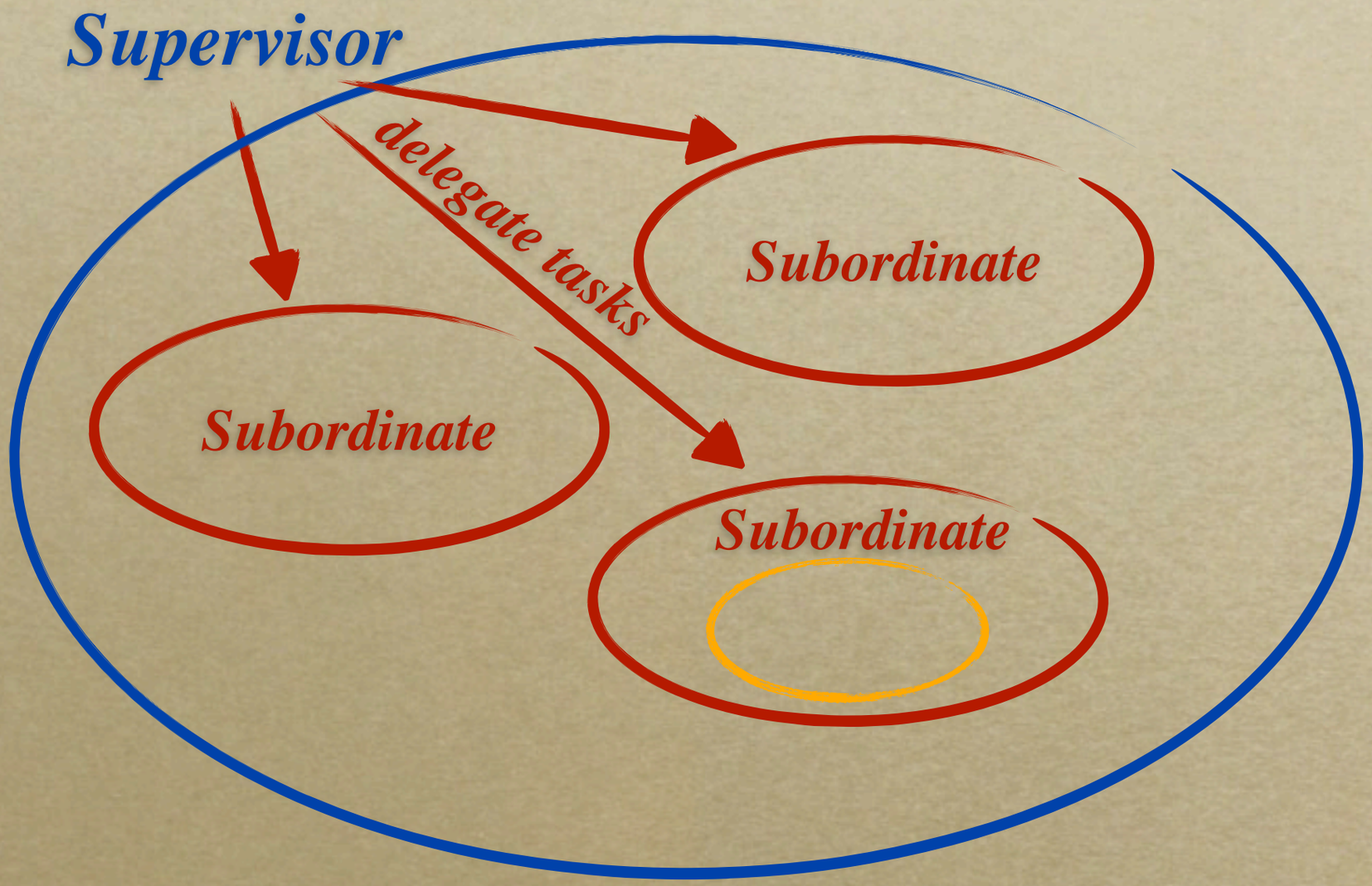
- *Lock-freedom*

```
public void lock_free_method ()
{  // Often calls take
   // finite number of steps
   --> No deadlocks
   --> Starvation possible
}
```

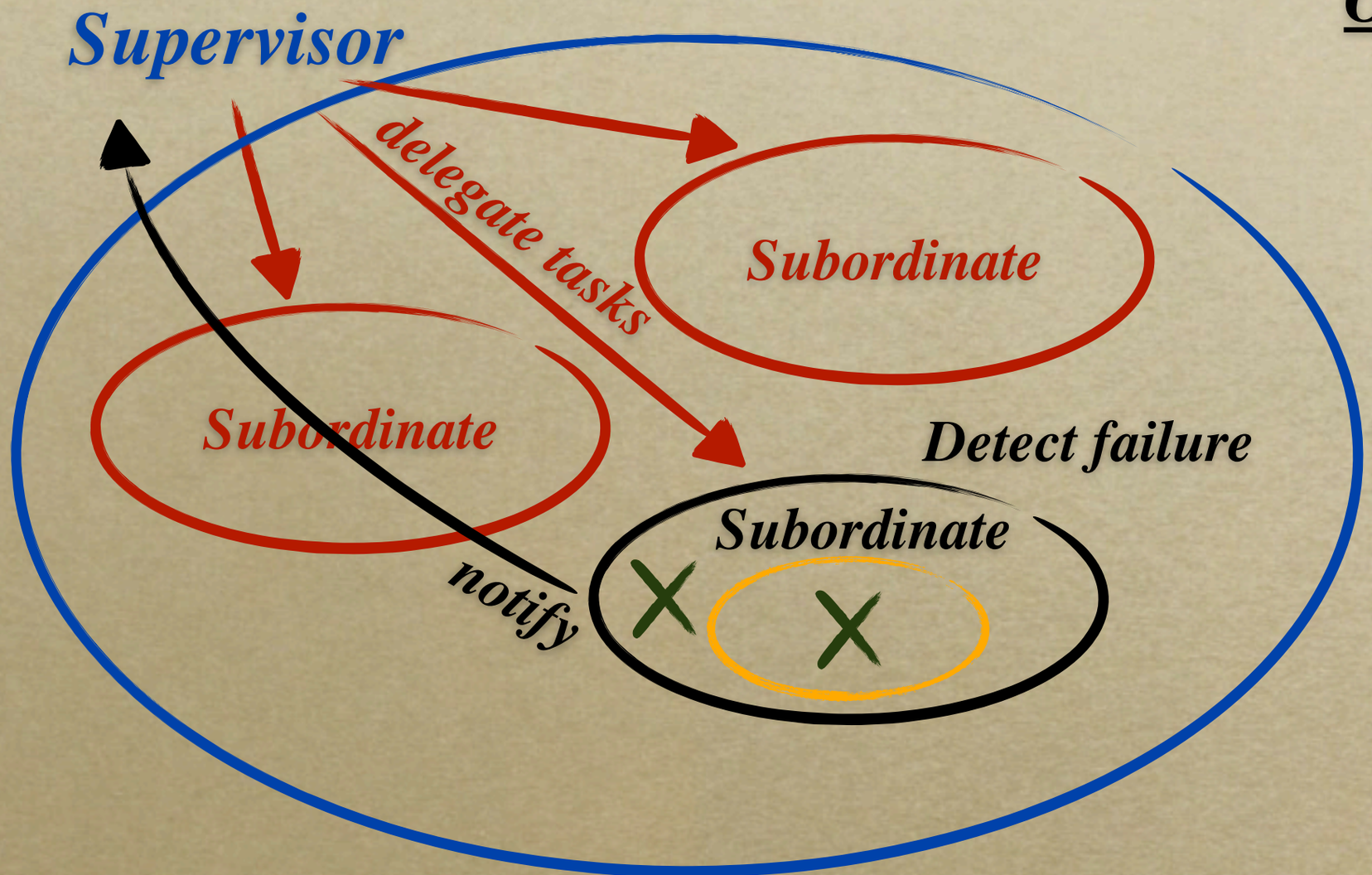
- *Obstruction-freedom*

```
public void obstruction_free_method ()
{  // If at some point in time
   // it executes in isolation
   // (others become suspended)
}
```

Supervision and Monitoring



Supervision and Monitoring



Options

1. Resume the subordinate, keeping its accumulated internal state

2. Restart the subordinate, clearing out its accumulated internal state

3. Terminate the subordinate permanently

4. Escalate the failure, thereby failing itself

Top-Level Supervisors

"the one who walks the bubbles of space-time"

ActorSystem's status "isTerminated" = True

SupervisorStrategy.stoppingStrategy

"root guardian"

/

--> Orderly shut-down

--> Logging

akka.actor.guardian-supervisor-strategy

"guardian"

"system guardian"

shutdown order

user

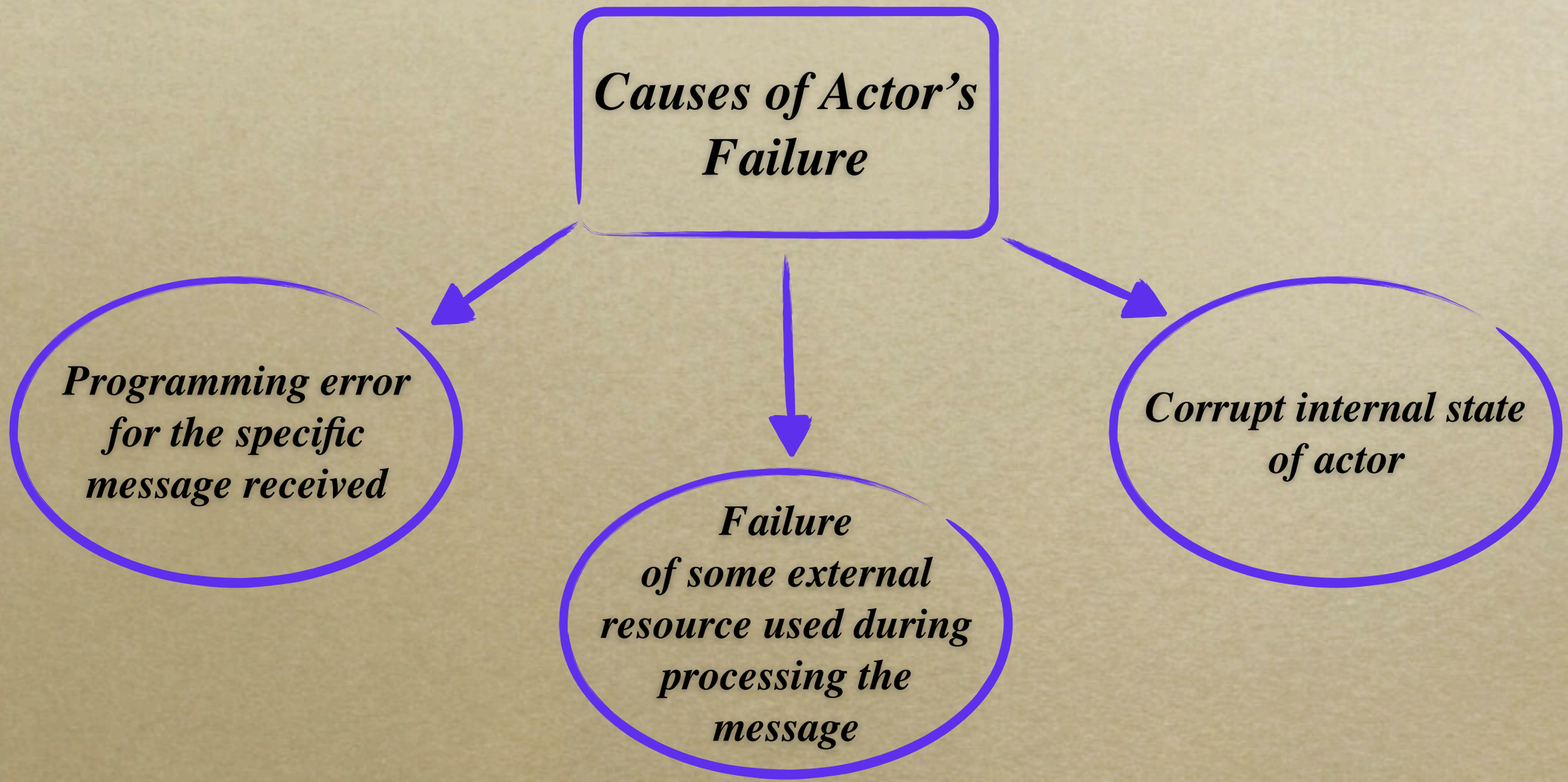
system

your actor hierarchy

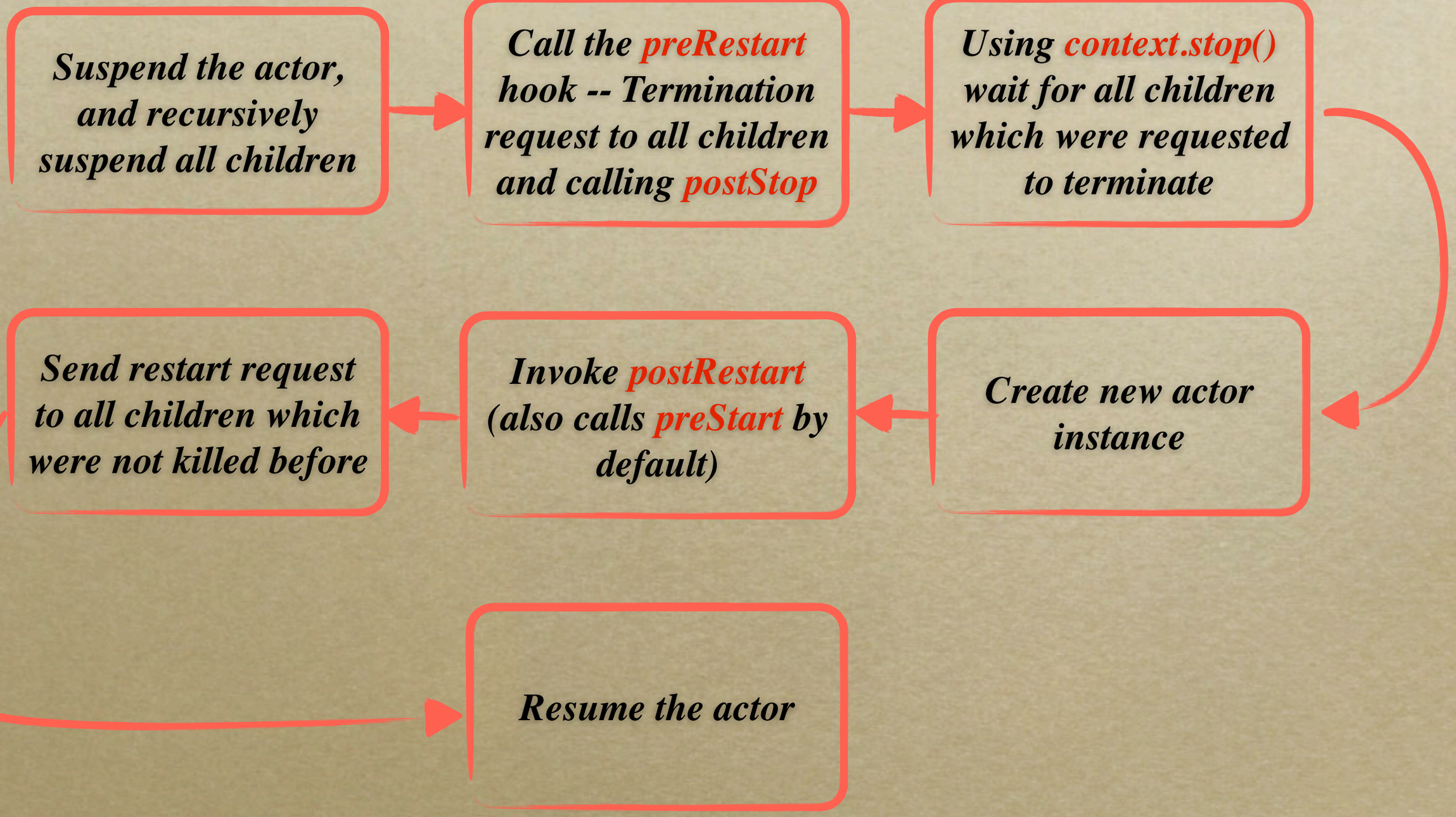
sys support hierarchy

system.actorof()

Restarting (1)



Restarting (2)



Lifecycle Monitoring (DeathWatch)

- *Each actor may monitor any other actor*
  - *Useful in the cases when supervisors have to terminate the children*
  - *Restarts are not visible outside the affected supervisors*
    - *Transition from **Alive** to **Dead** can be only monitored using Terminated message*
    - *ActorContext.watch(targetActorRef)*
    - *ActorContext.unwatch(targetActorRef)*

Message Delivery (1)

General Rules

at-most-once delivery

no guaranteed delivery

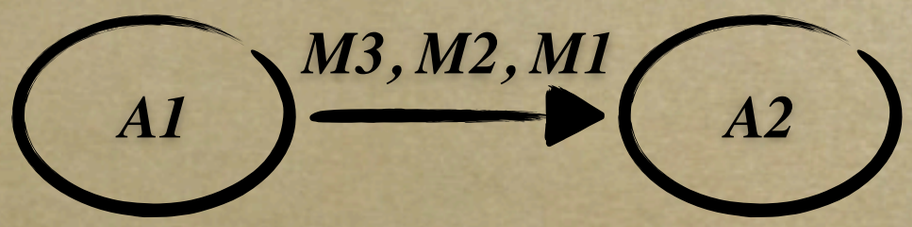
- ***at-most-once*** *-- cheapest - highest performance, least implementation overhead*
- ***at-least-once*** *-- acknowledgement*
- ***exactly-once*** *-- most expensive - worst performance*

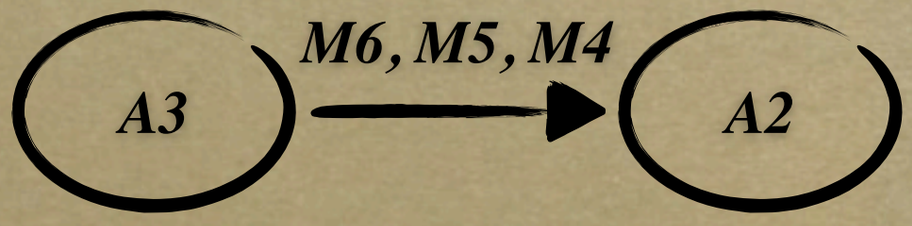
Message Delivery (2)

General Rules

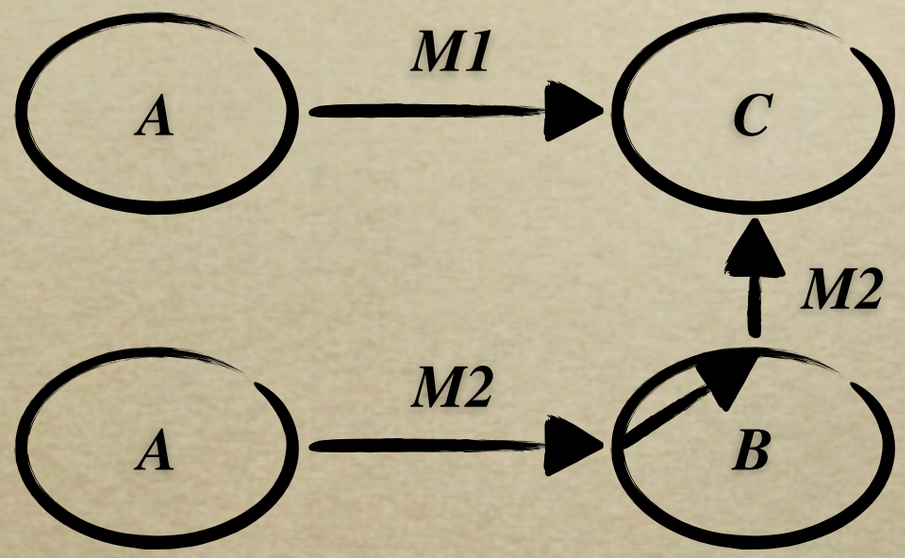
message ordering per sender-receiver pair

in-order delivery

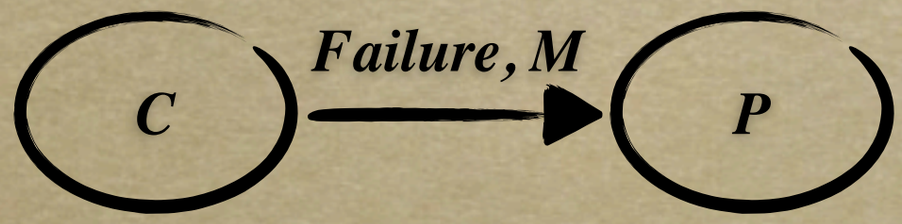




this rule is not transitive

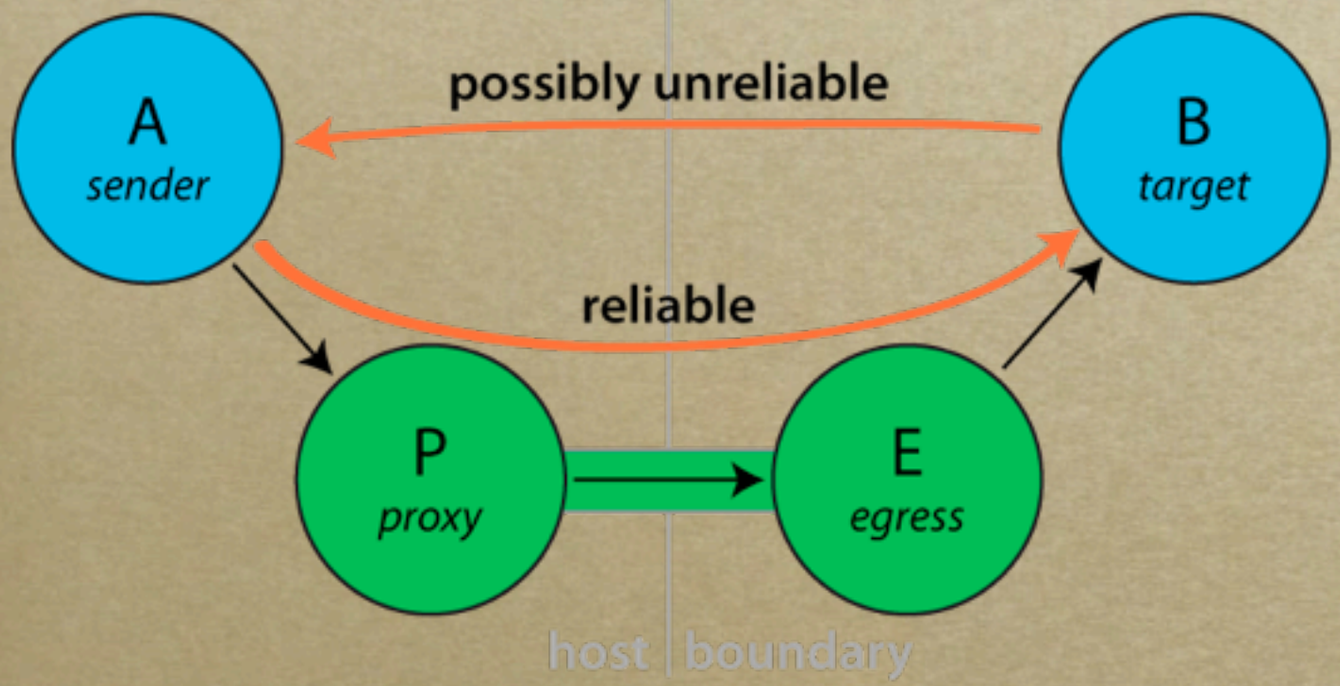


any order



Guaranteed Delivery

ACK-RETRY Ptotocol



A way to identify individual messages to correlate message with acknowledgement

A retry mechanism which will resend messages if not acknowledged in time

A way for the receiver to detect and discard duplicates

Mailbox with Explicit Acknowledgement

```
class MyActor extends Actor {
  def receive = {
    case msg ⇒
      println(msg)
      doStuff(msg) // may fail
      PeekMailboxExtension.ack()
  }

  // business logic elided ...
}

object MyApp extends App {
  val system = ActorSystem("MySystem", ConfigFactory.parseString("""
    peek-dispatcher {
      mailbox-type = "akka.contrib.mailbox.PeekMailboxType"
      max-tries = 2
    }
    """))

  val myActor = system.actorOf(Props[MyActor].withDispatcher("peek-dispatcher"),
    name = "myActor")

  myActor ! "Hello"
  myActor ! "World"
  myActor ! PoisonPill
}
```

Actors, STM and the Java Memory Model

- *The actor send rule:*
- *The actor subsequent processing rule:*
- *The transactional reference rule:*