Actor programming model in Akka.NET

BY RICCARDO TERRELL. - @TRIKACE

The issue is Shared of Memory

Shared Memory Concurrency

Data Race / Race Condition

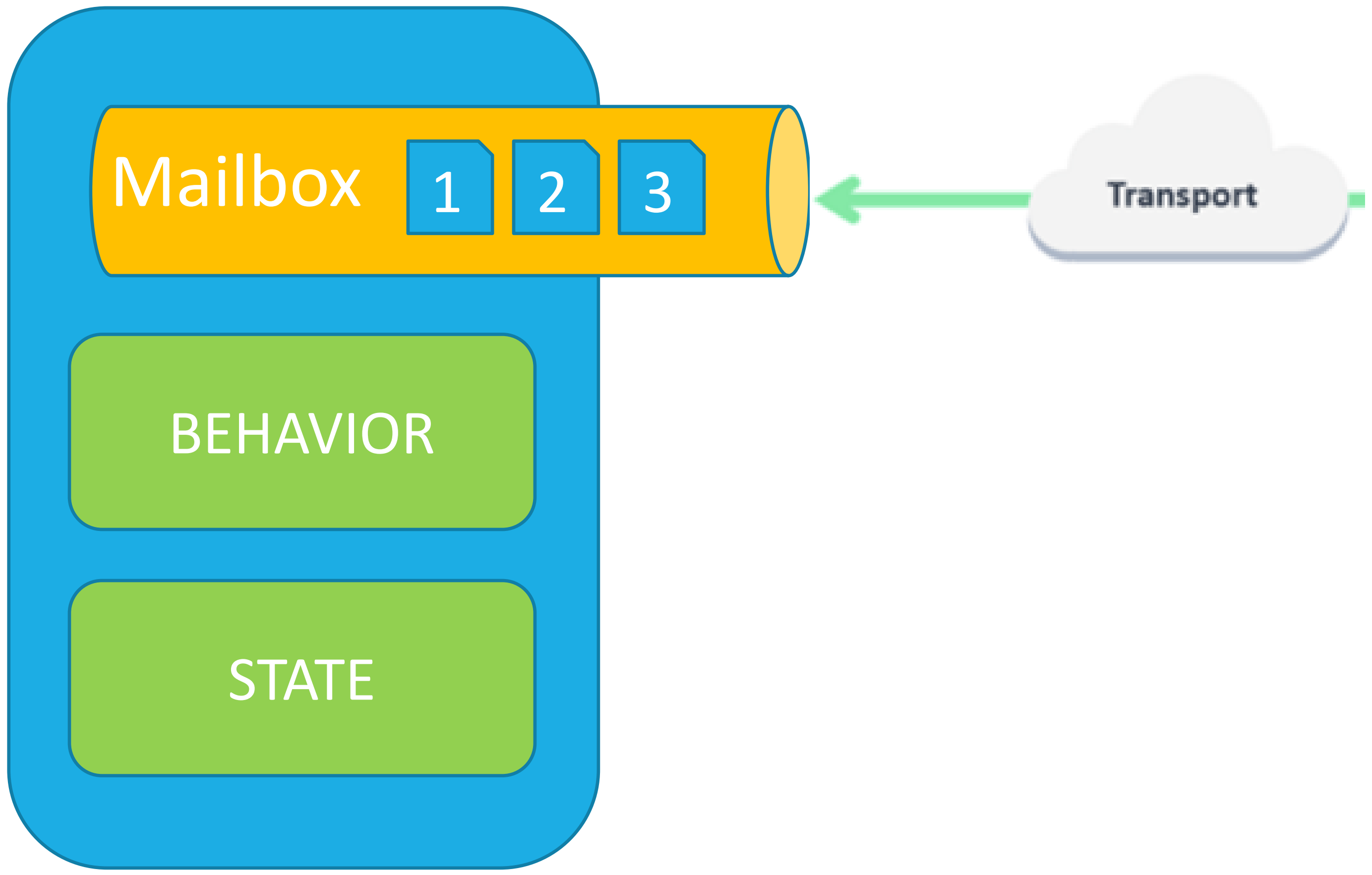
Works in sequential single threaded environment

Not fun in a multi-threaded environment

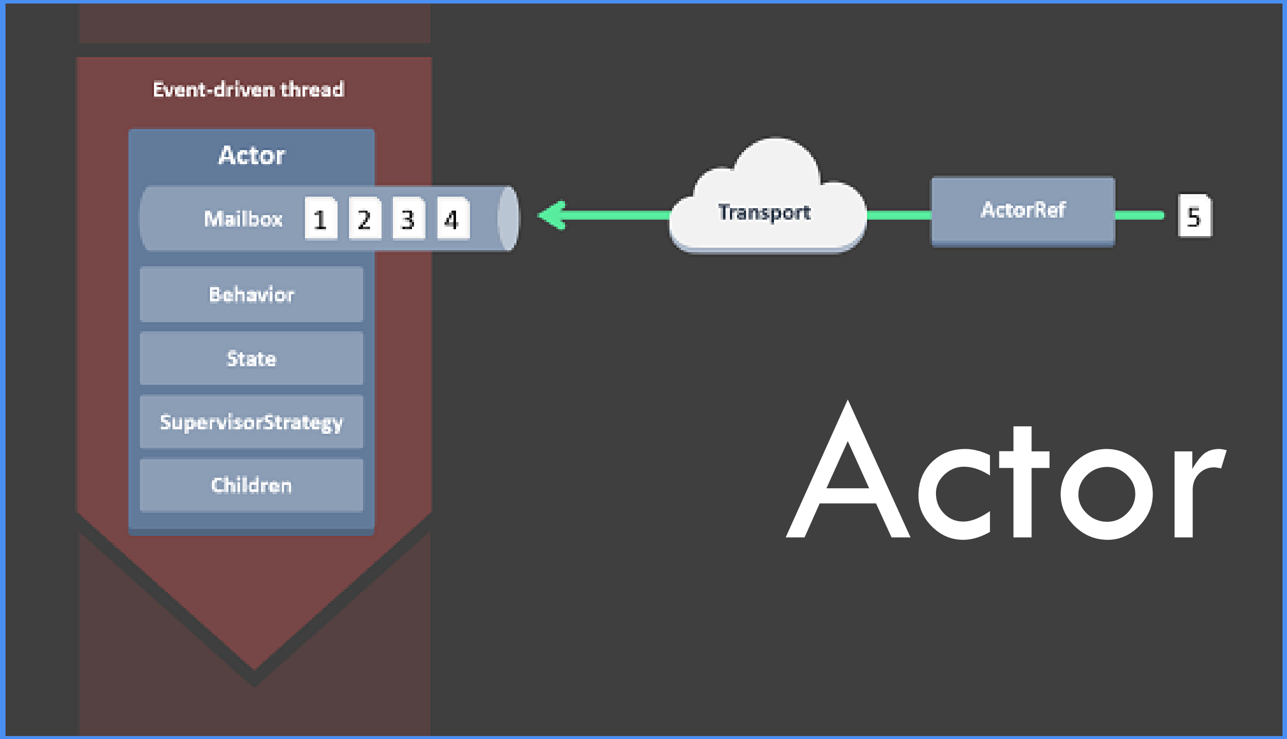
Not fun trying to parallelize

- Locking, blocking, call-back hell

Message Passing based concurrency



- Processing
- Storage – State
- Communication only by messages
- Share Nothing
- Message are passed by value
- Lightweight object
- Running on it’s own thread
- No shared state
- Messages are kept in mailbox and processed in order
- Massively scalable and lightening fast because of the small call stack



Actor Model

Actor Model Three axioms:

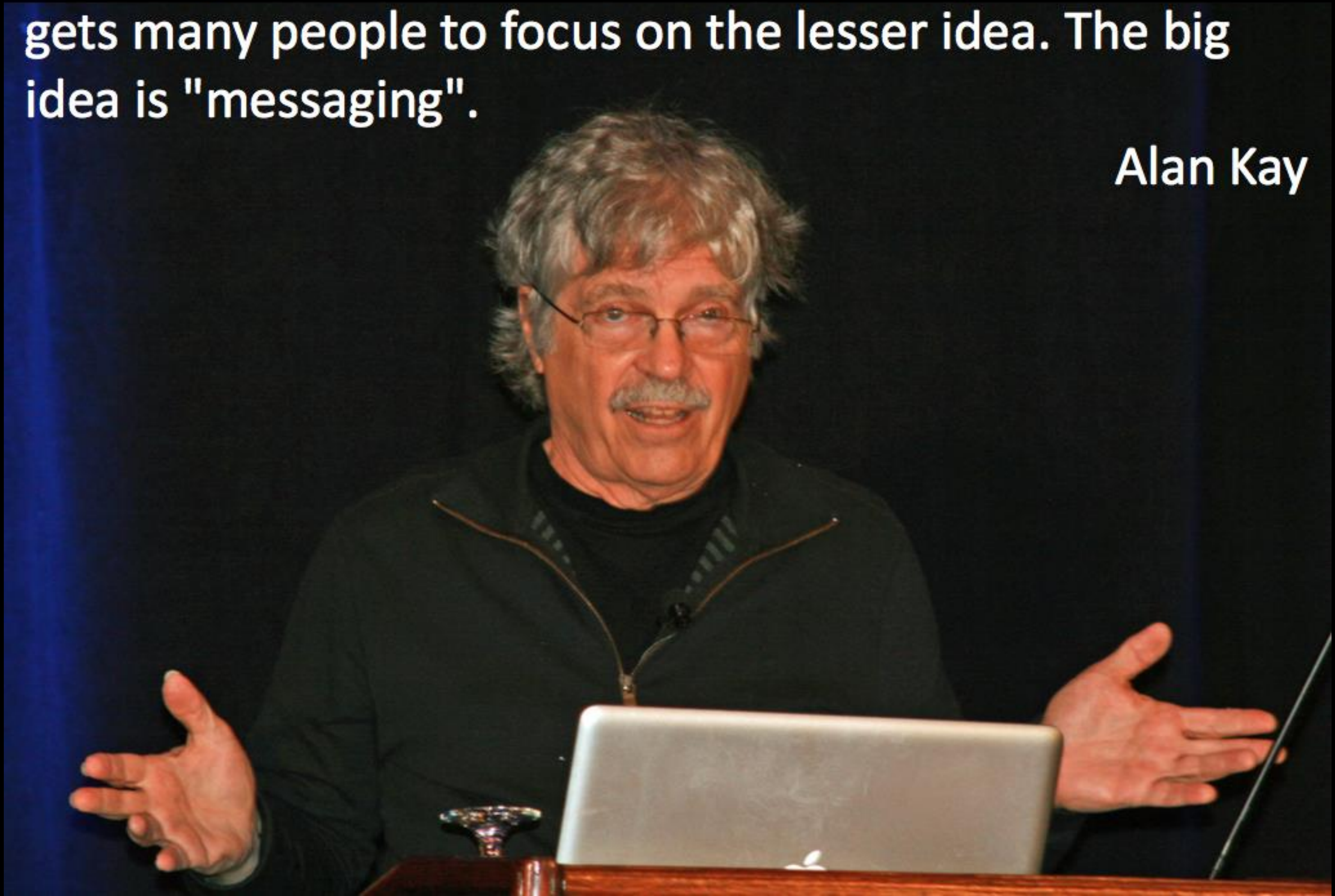
1. Send messages to other Actors

   - One Actor is not Actor -

2. Create other Actor
3. Decide how to handle the next message

I'm sorry that I coined the term "objects", because it gets many people to focus on the lesser idea. The big idea is "messaging".

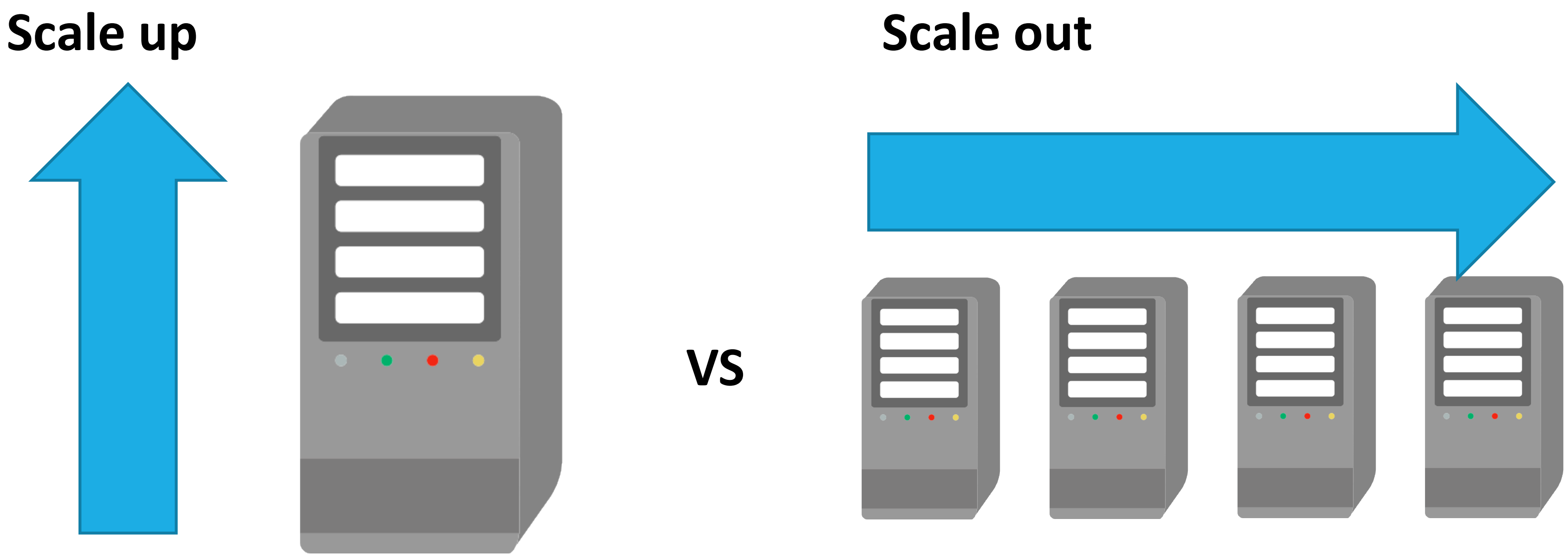
Alan Kay



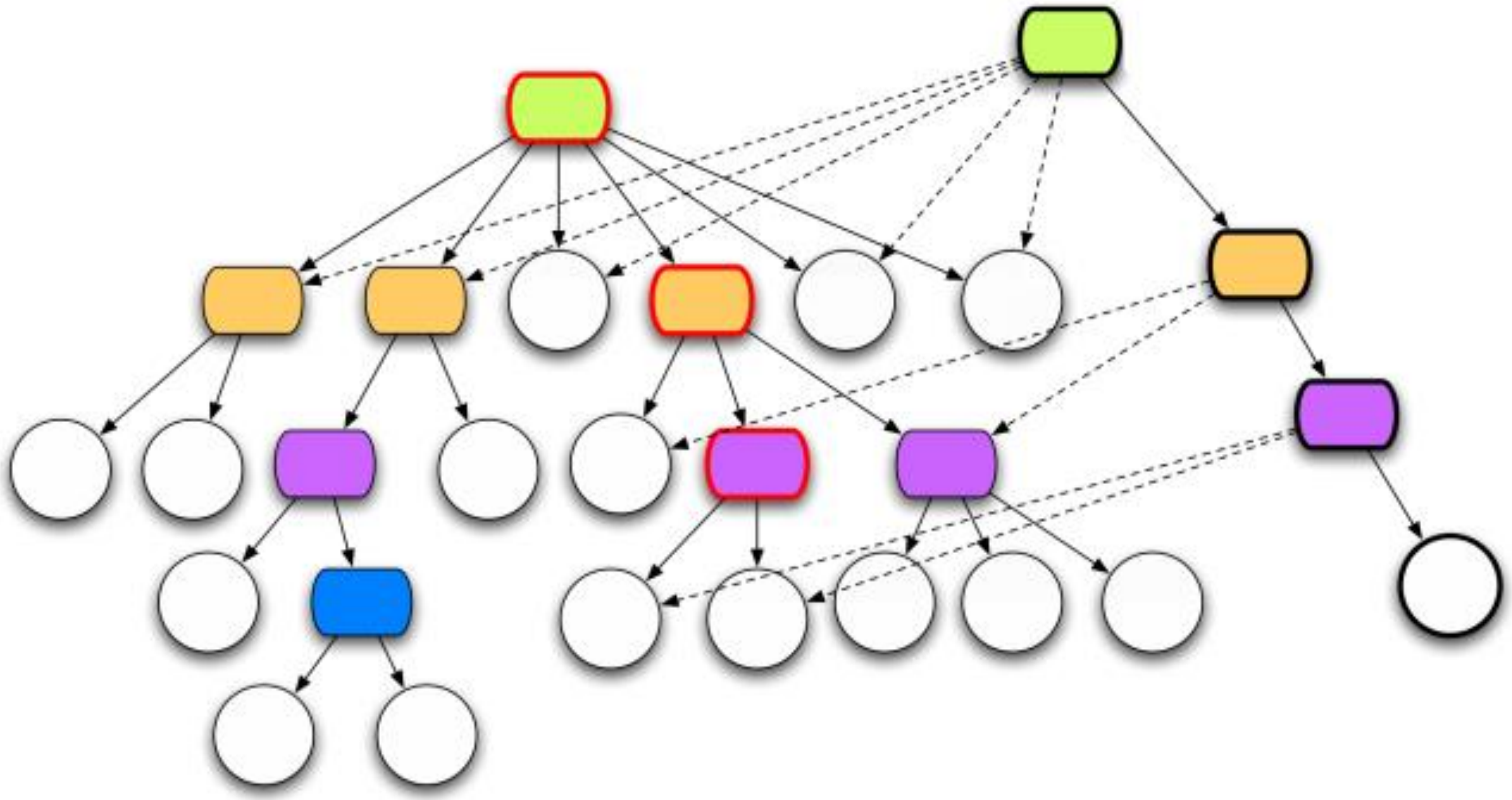
Reactive Manifesto & Actor Model

Responsive	Event Driven
Message-Driven	Communication by messages
Resilient	Fault tolerant by Supervision
Elastic	Expand and de-contract on demand

Its about maximizing resource use



Actor System



Akka.NET

- Akka.Remote
- Akka.Cluster
- Akka.Persistence
- Akka.Streams

* *Concurrent*

* *Resilient*

* *Distributed*

* *Scalable*

What is Akka.Net

Akka.NET is a port of the popular Java/Scala framework Akka to .NET.

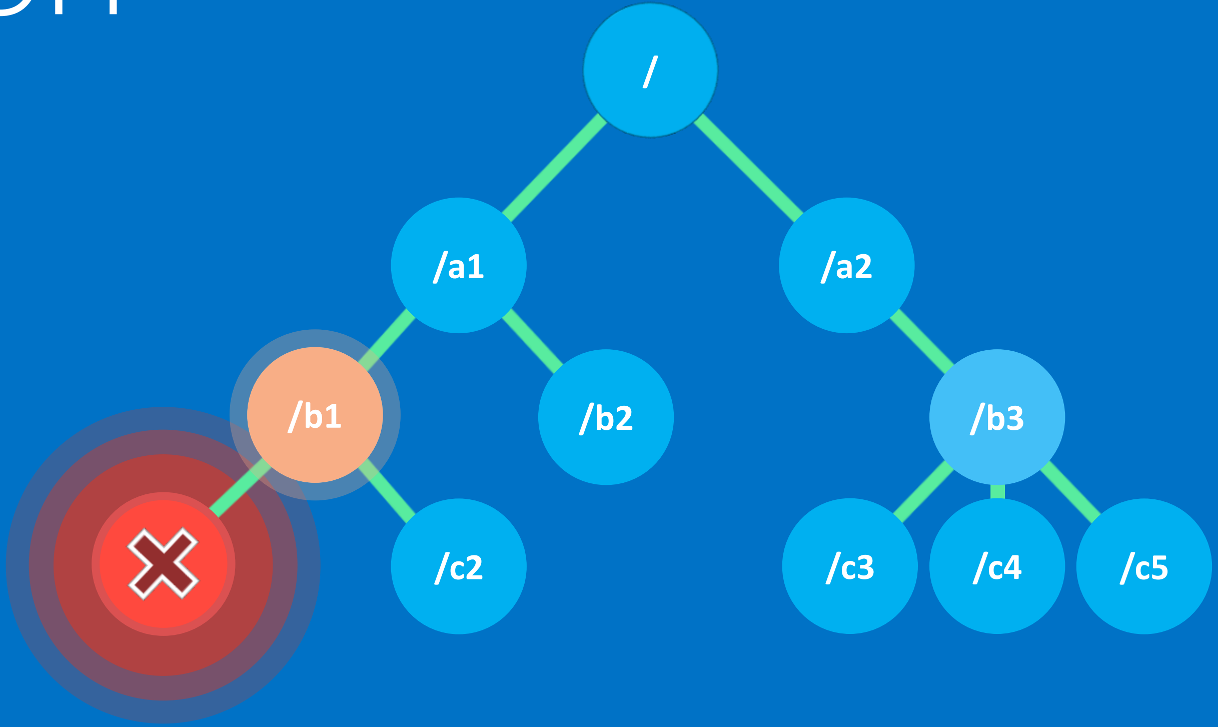
"Akka is a toolkit and runtime for building highly concurrent, distributed, and resilient message-driven applications on the JVM."

- Typesafe

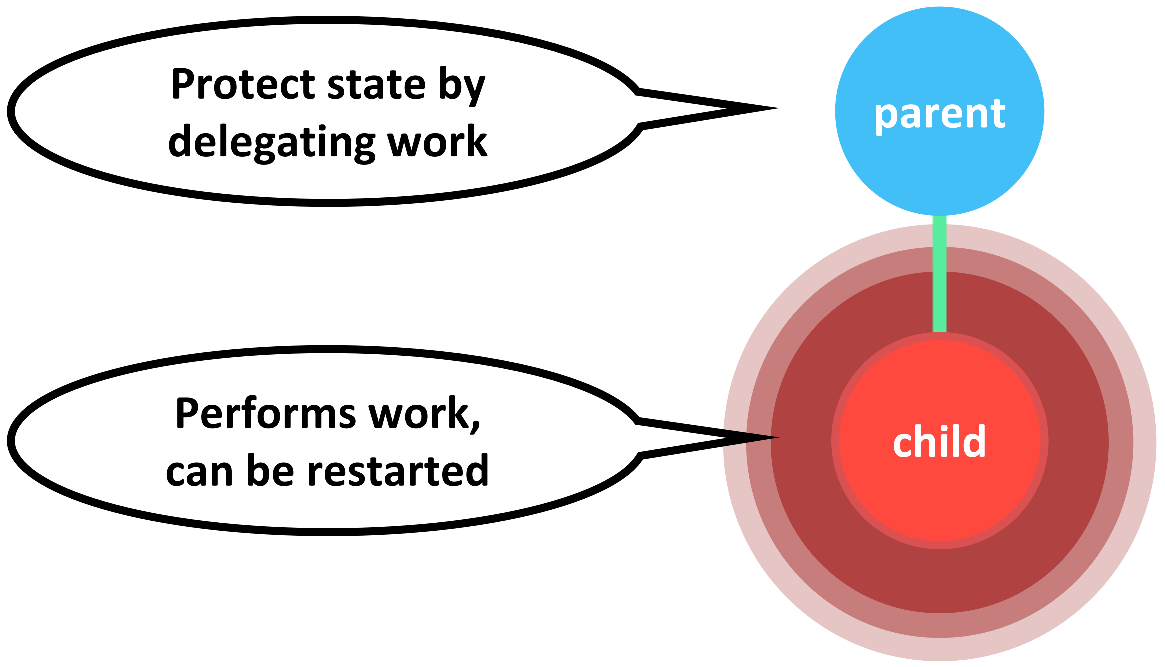
Actor – Akka.NET in C#

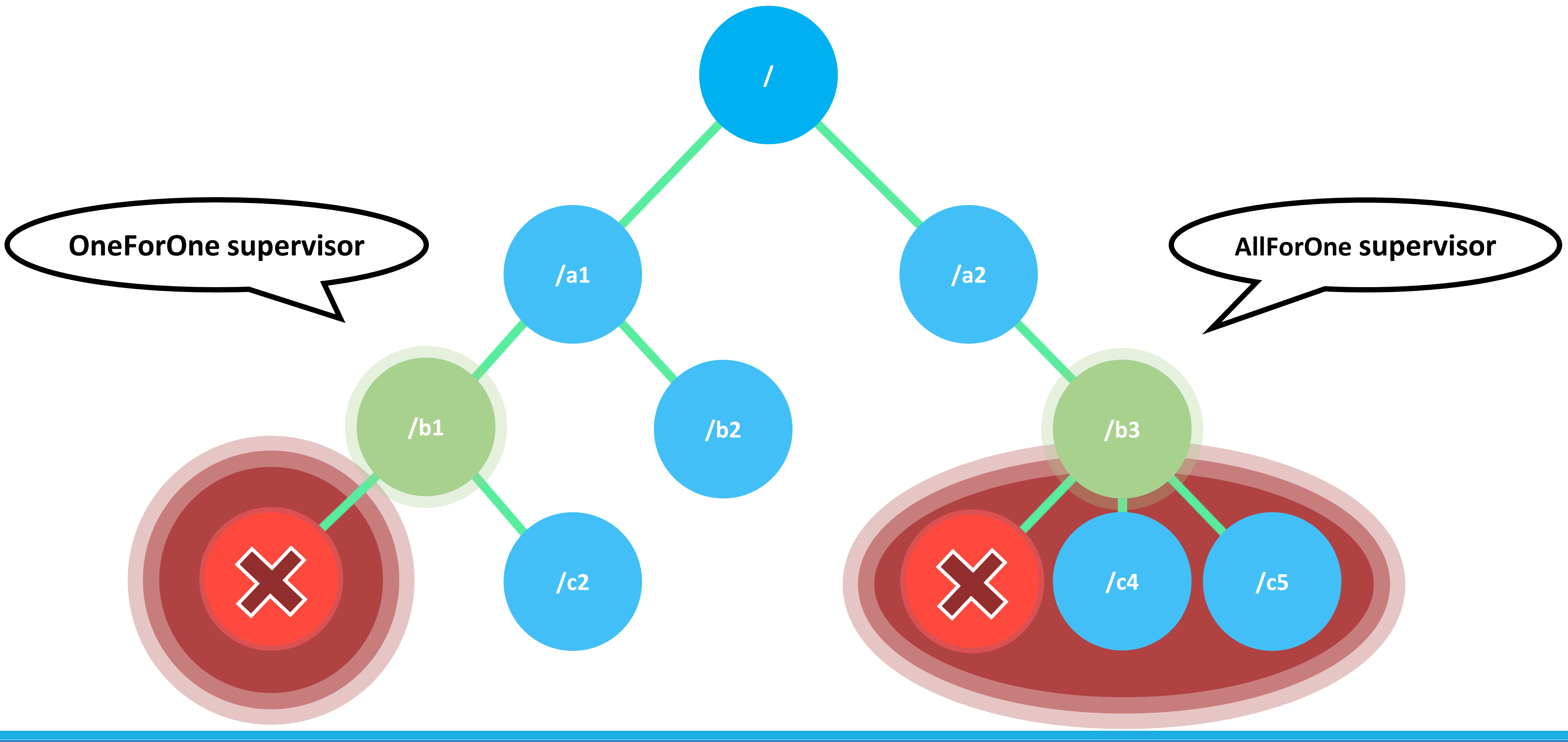
```
var system = ActorSystem.Create("fizz-buzz");

public class FizzBuzzActor : ReceiveActor {
    public FizzBuzzActor()     {
        Receive<FizzBuzzMessage>(msg => {
            // code to handle the message
        });
    }
}

var actor = system.ActorOf(Props.Create<FizzBuzzActor>(), "fb-actor");
actor.Tell(new FizzBuzzMessage(5));
```

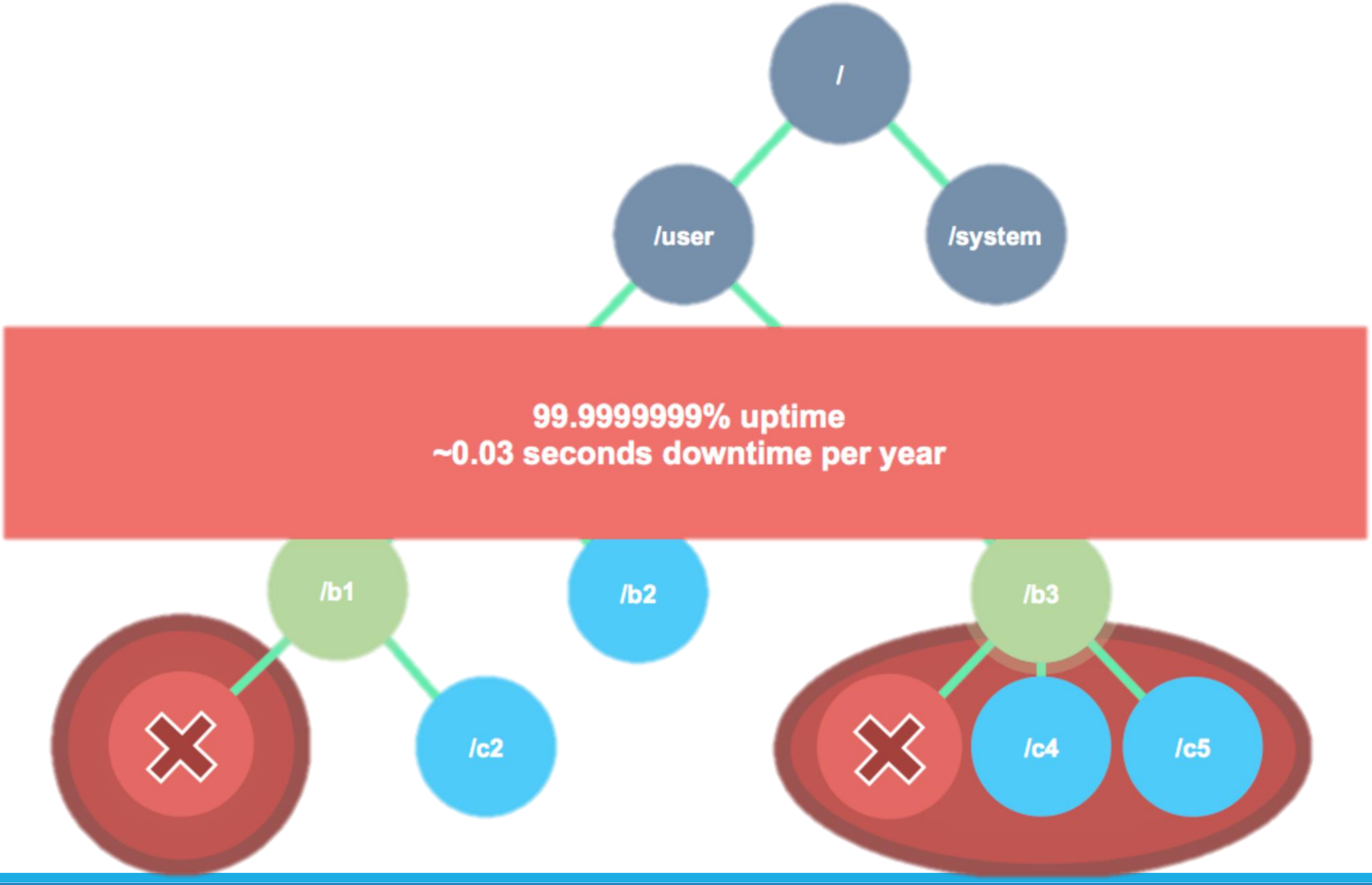
Supervision



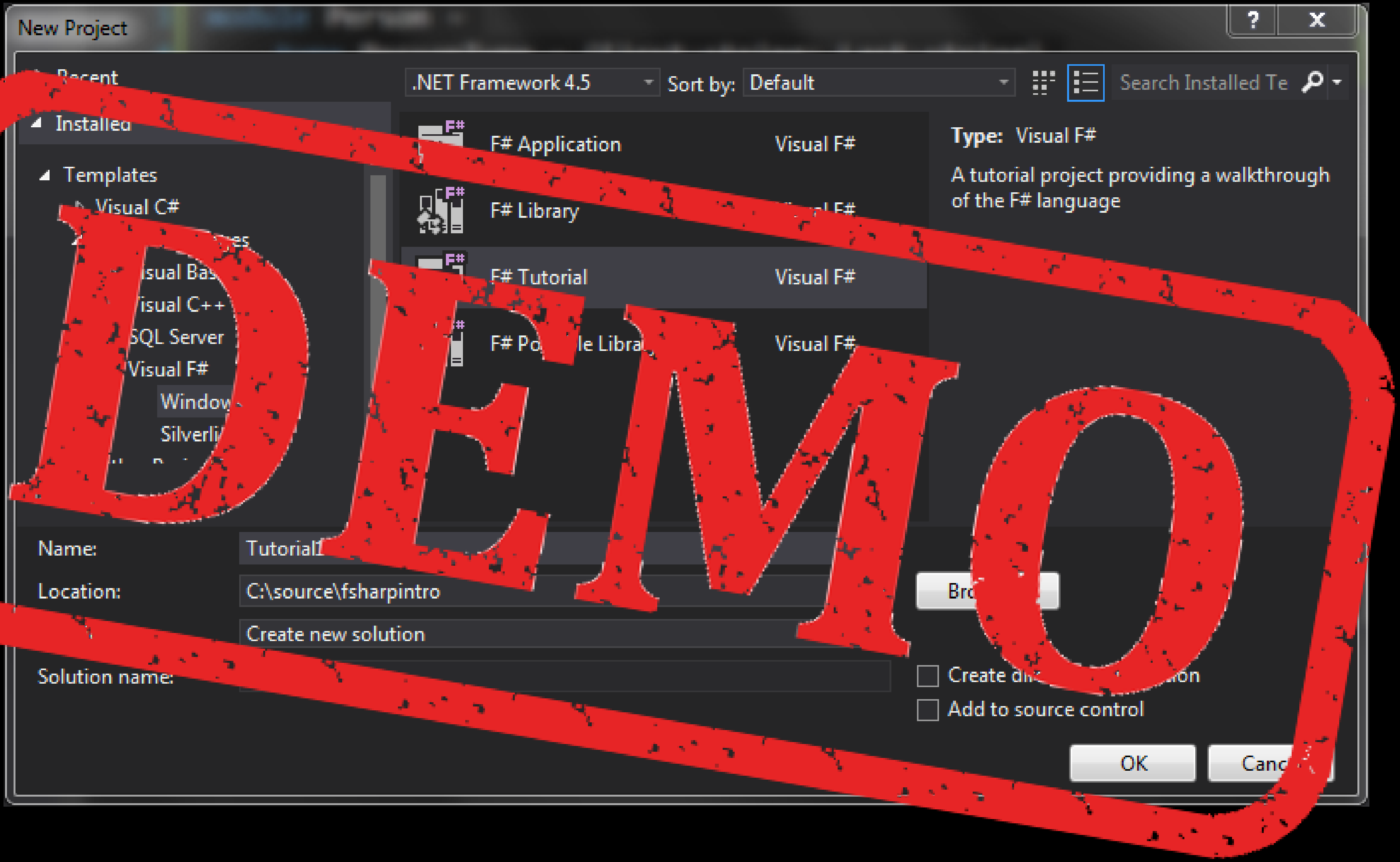
Let it crash


Protect state by delegating work
parent
Performs work, can be restarted
child

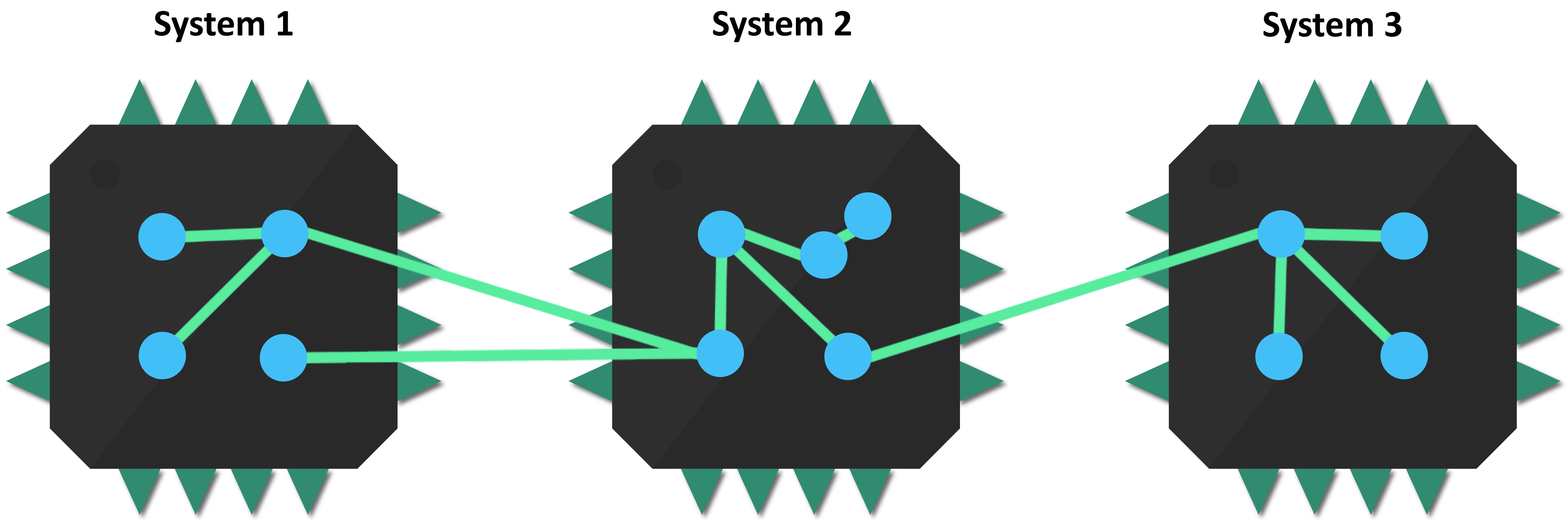

OneForOne supervisor
/
/a1
/a2
AllForOne supervisor
/b1
/b2
/b3
/c2
/c4
/c5



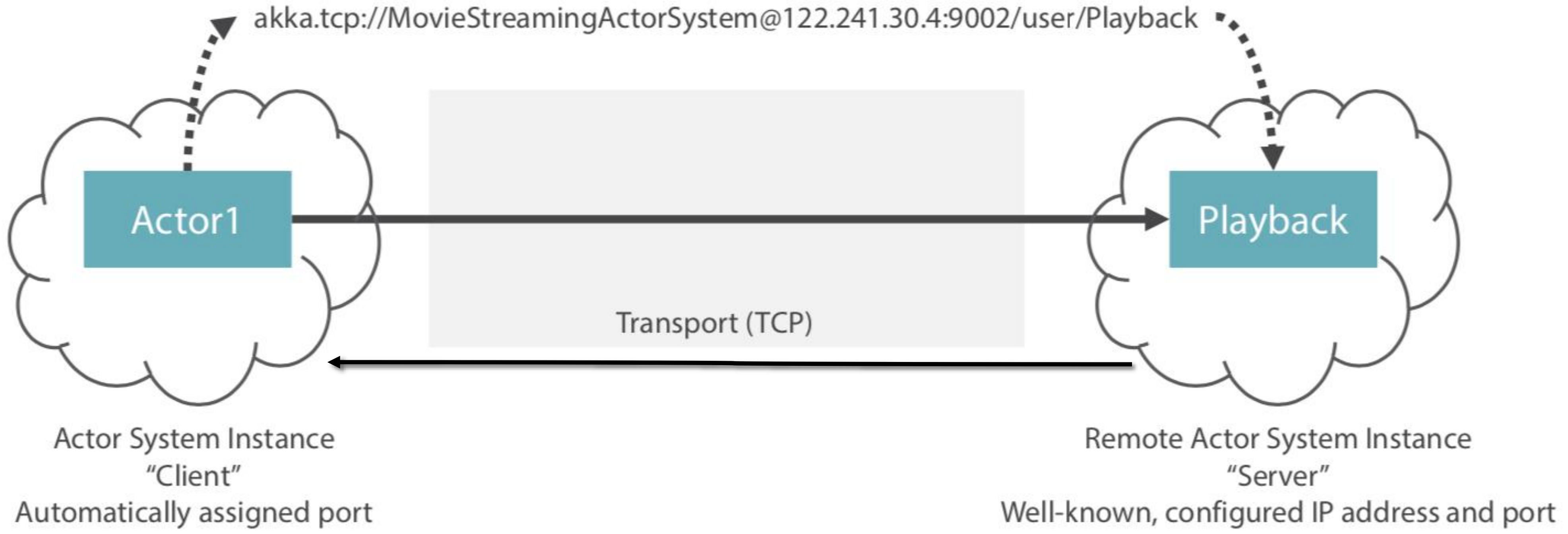
99.9999999% uptime
~0.03 seconds downtime per year


DEMO
New Project
Recent
Installed
Templates
Visual C#
Visual Basic
Visual C++
SQL Server
Visual F#
Windows
Silverlight
.NET Framework 4.5
Sort by: Default
Search Installed Te
F# Application
Visual F#
F# Library
F# Tutorial
Visual F#
Visual F#
Type: Visual F#
A tutorial project providing a walkthrough of the F# language
Name:
Tutorial1
Location:
C:\source\fsharpintro
Create new solution
Solution name:
Add to source control
OK

Remoting

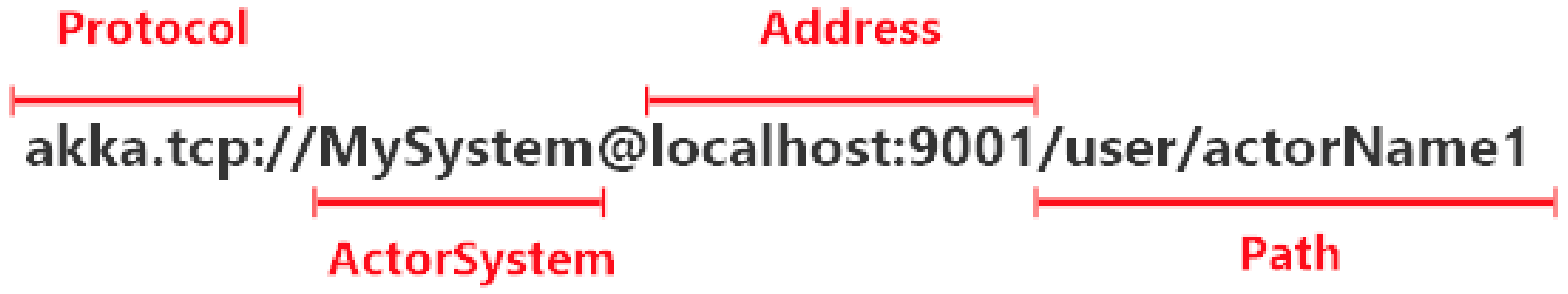

System 1
System 2
System 3

Remoting


akka.tcp://MovieStreamingActorSystem@122.241.30.4:9002/user/Playback
Actor1
Playback
Transport (TCP)
Actor System Instance
"Client"
Automatically assigned port
Remote Actor System Instance
"Server"
Well-known, configured IP address and port

Location Transparency

What location transparency means is that whenever you send a message to an actor, you don't need to know where they are within an actor system, which might span hundreds of computers. You just have to know that actors' address.


akka.tcp://MySystem@localhost:9001/user/actorName1

Remotely Deploying Actors

Deploying an actor means two things simultaneously:

Creating an actor instance with specific, explicitly configured properties

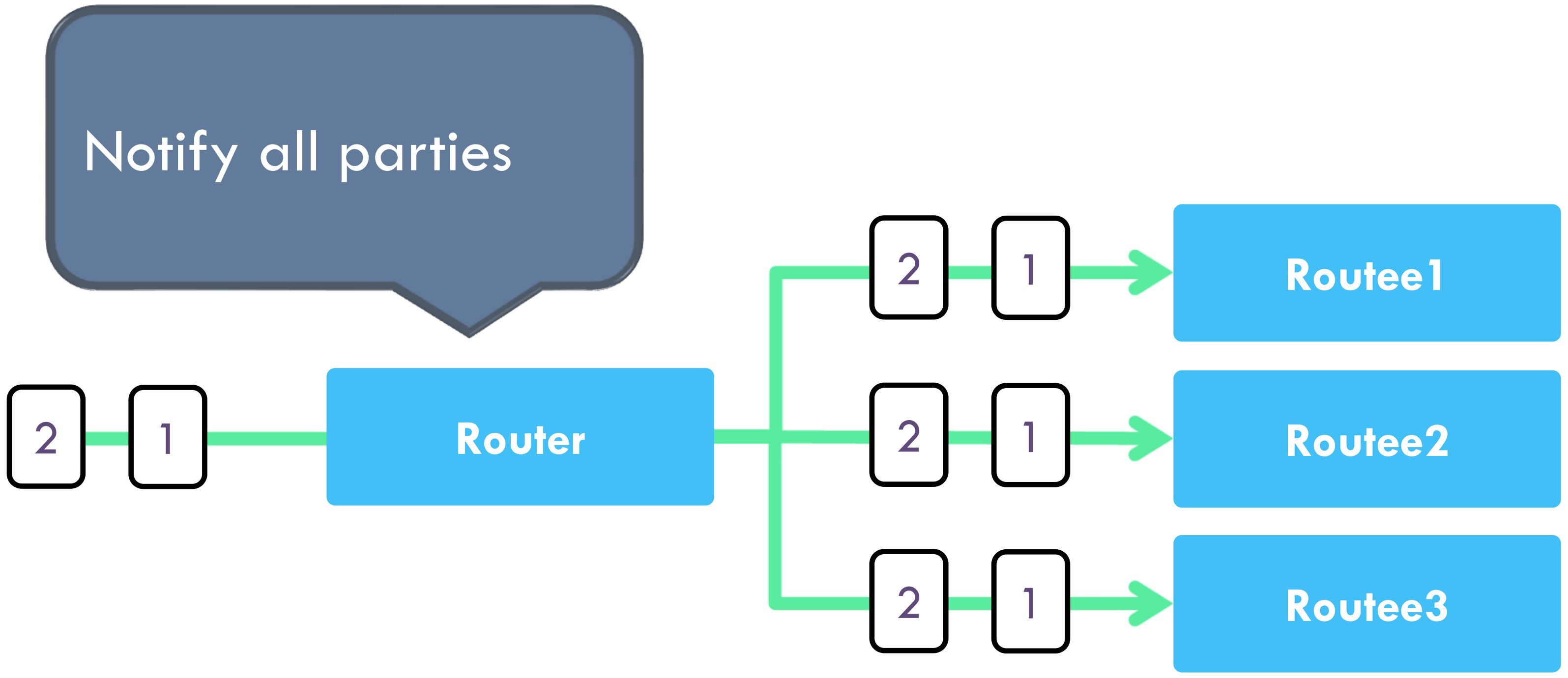
Getting an ActorRef to that actor

```
using (var system = ActorSystem.Create("Deployer", ConfigurationFactory.ParseString(@” // CLIENT
        akka {  ... CONFIG AS BEFORE ... }
           remote {
              helios.tcp {
                 port = 0
                 hostname = localhost
              }
           }
        }")))
     {
              var remoteEcho1 = system.ActorOf(Props.Create(() => new EchoActor()), "remoteecho");

              var echoActor = system.ActorOf(Props.Create(() => new HelloActor(remoteEcho1)));
              echoActor.Tell(“Hello”)
```

Routing

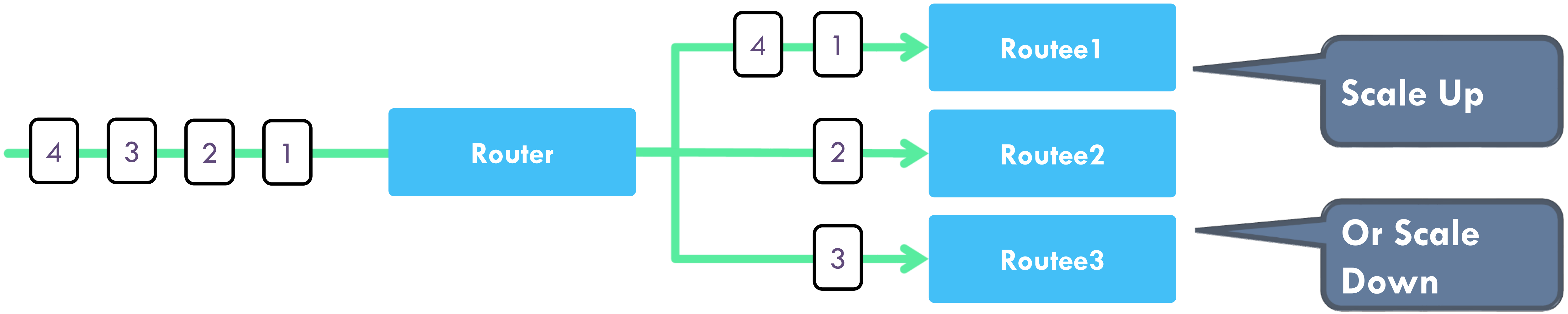
Akka.Net Routing Strategies

Broadcast router will as the name implies, broadcast any message to all of its routees



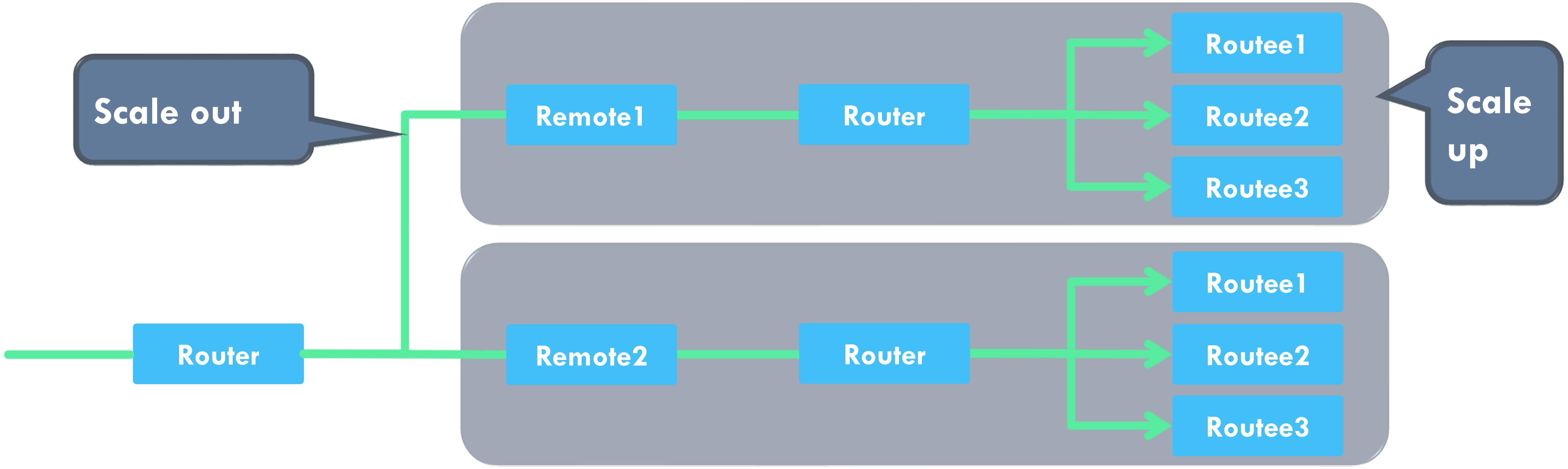
Akka.Net Routing Strategies

RoundRobin-Pool router uses round-robin to select a connection. For concurrent calls, round robin is just a best effort.

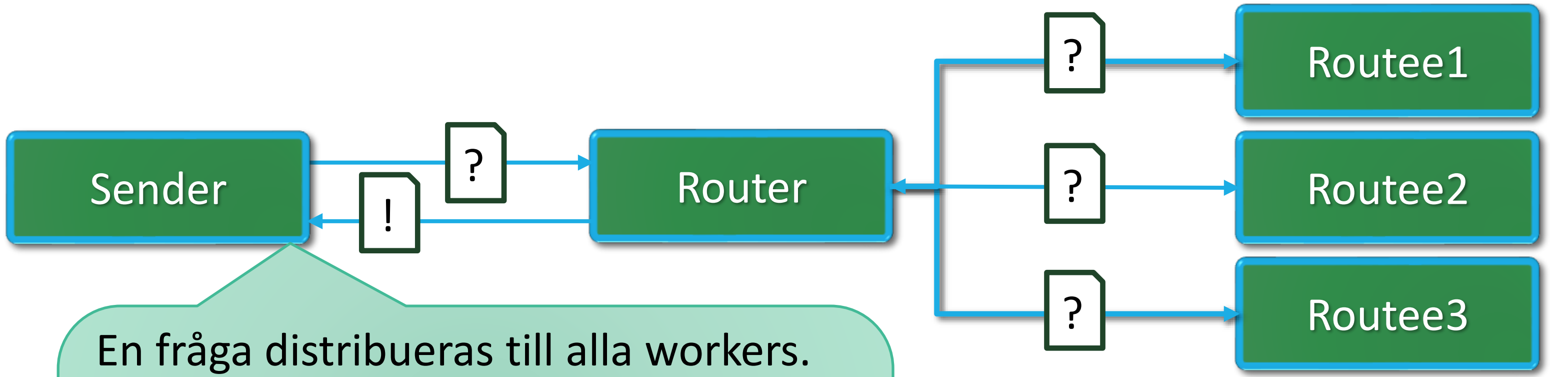


Akka.Net Routing Strategies

RoundRobin-Group router uses round-robin to select a connection. For concurrent calls, round robin is just a best effort.



ScatterGatherFirstCompletedRouter



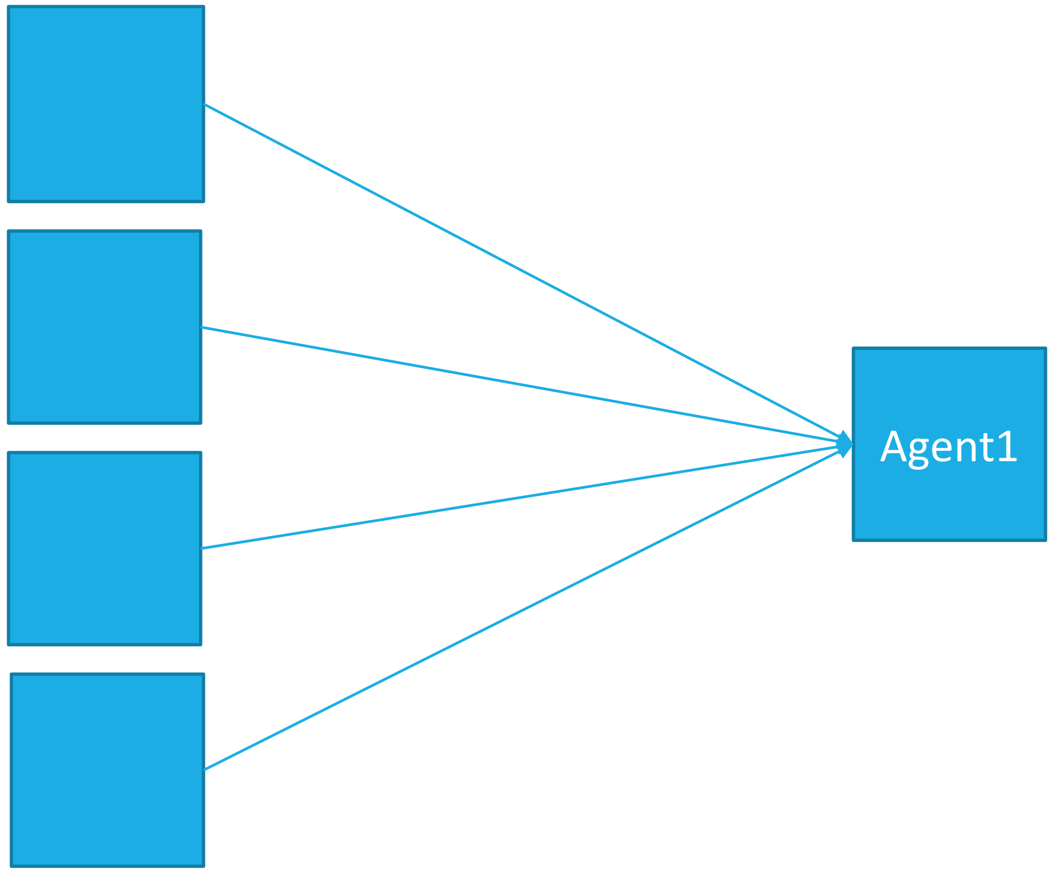
En fråga distribueras till alla workers.
Första svaret returneras till avsändaren

Konceptuellt samma som TPL
Task.WaitAny(tasks)

Actor Patterns

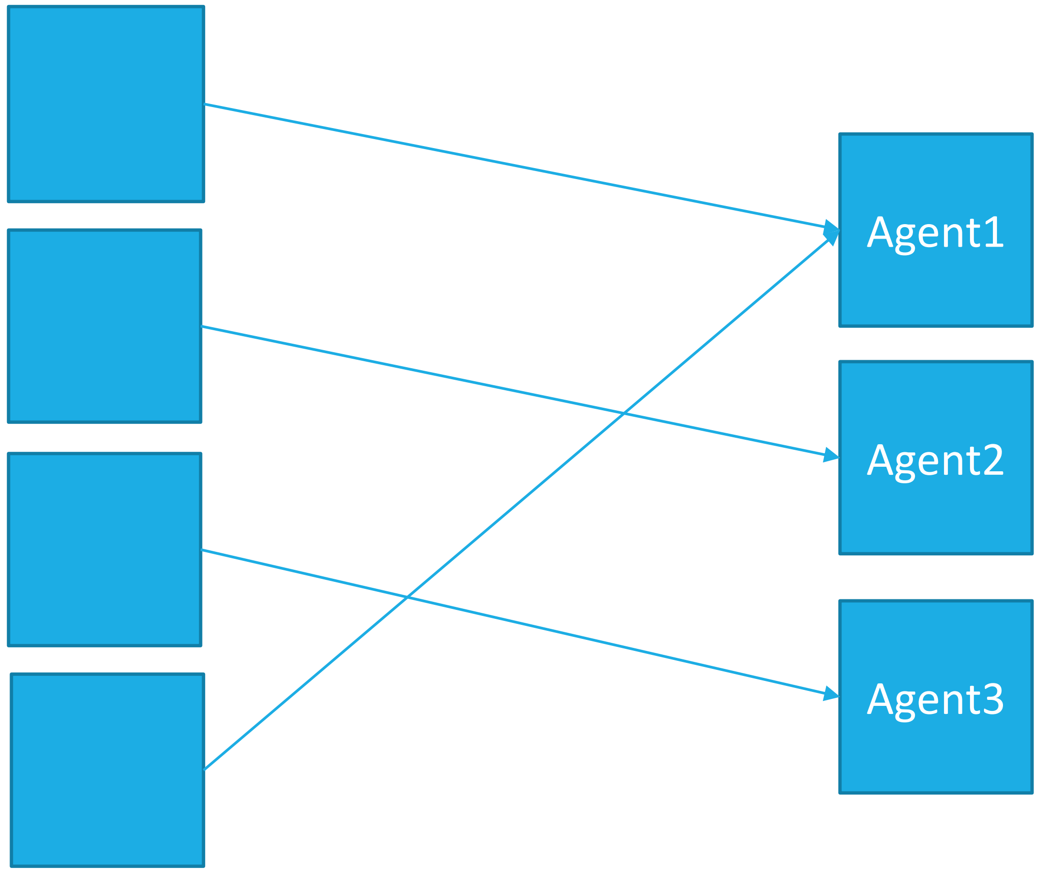
Actor patterns

Singleton



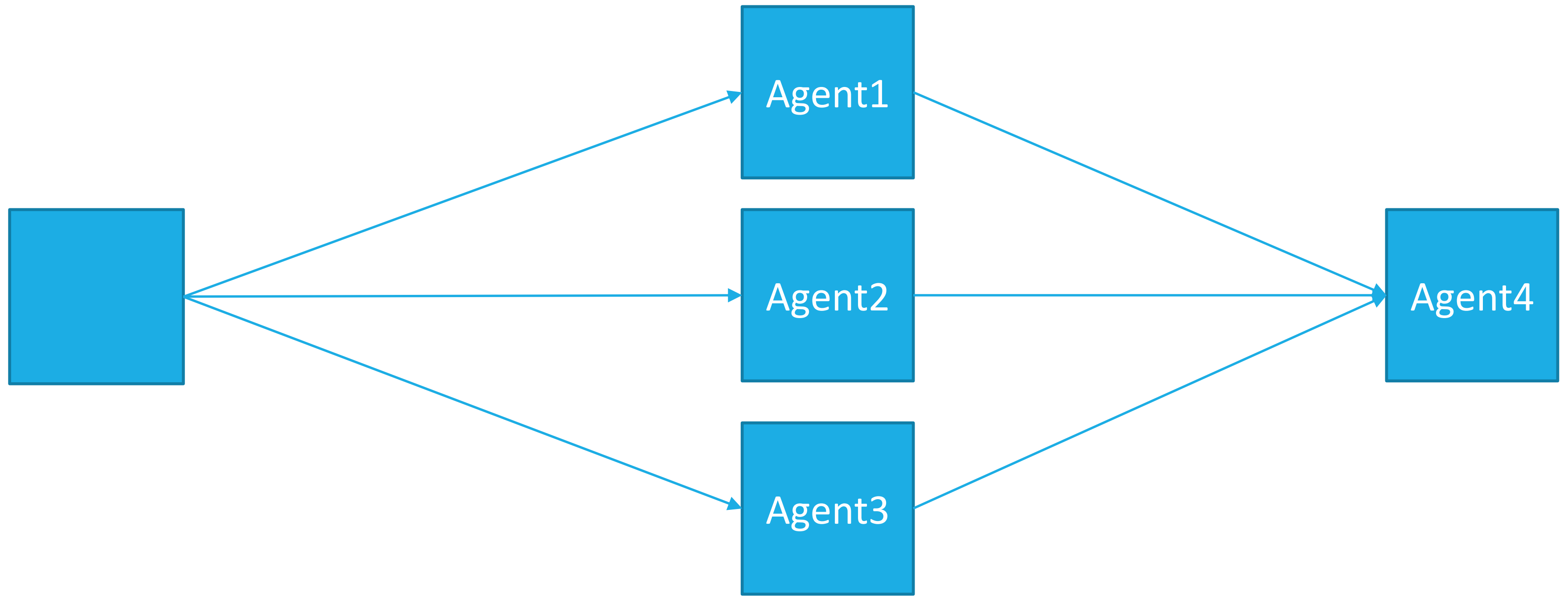
Actor patterns

Pool



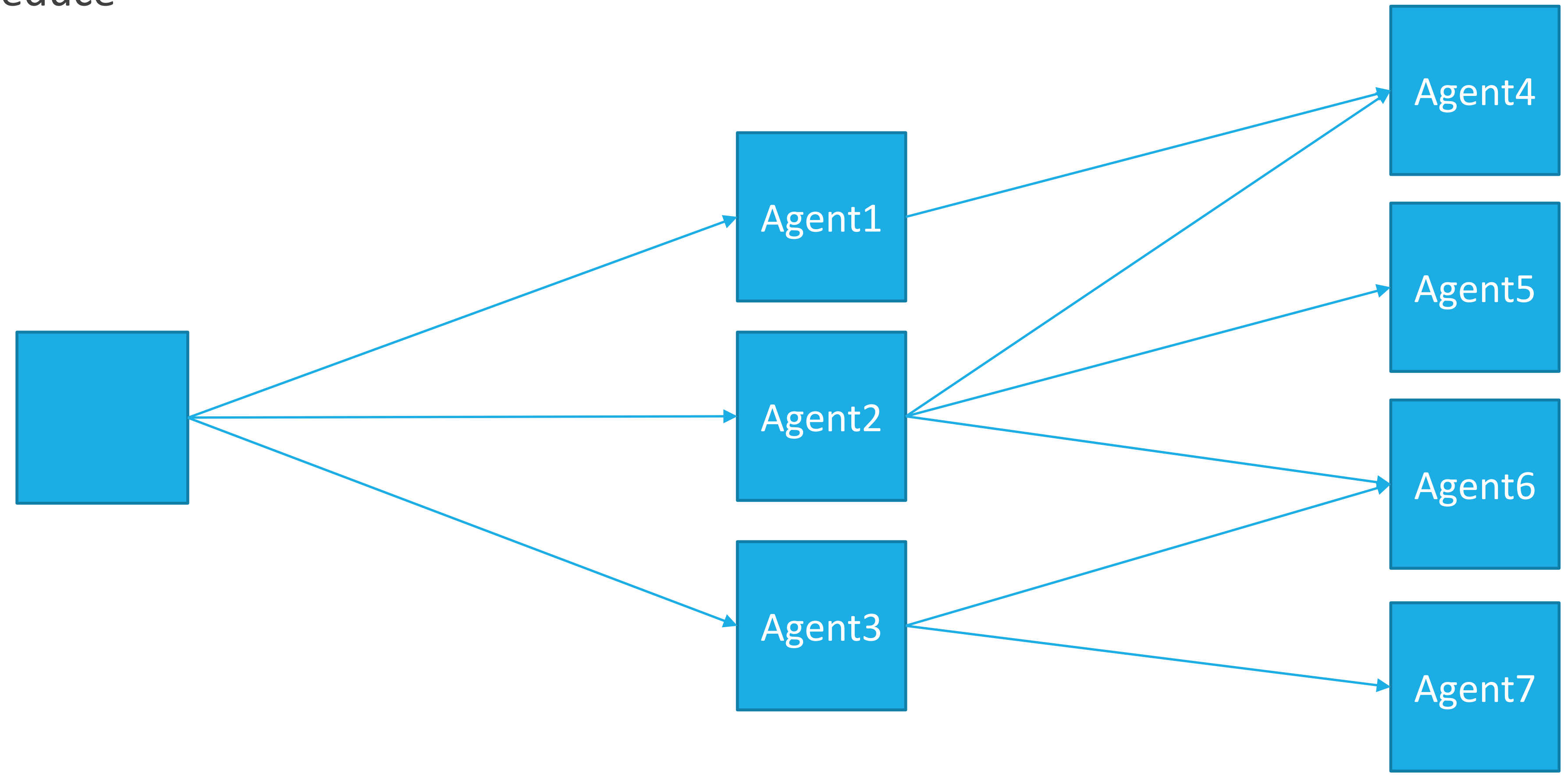
Actor patterns

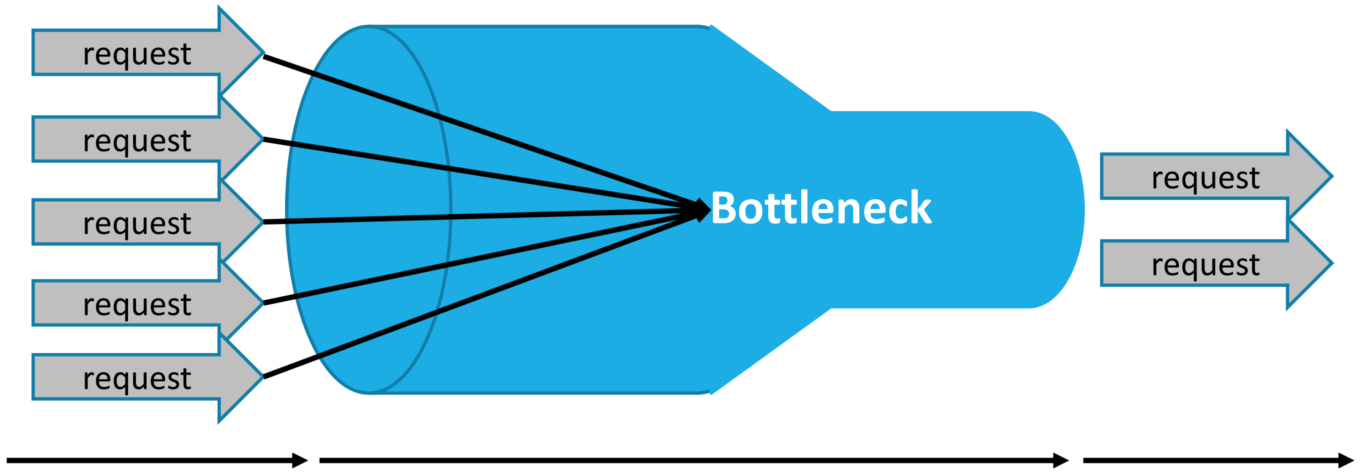
Fork Join



Actor patterns

Map Reduce

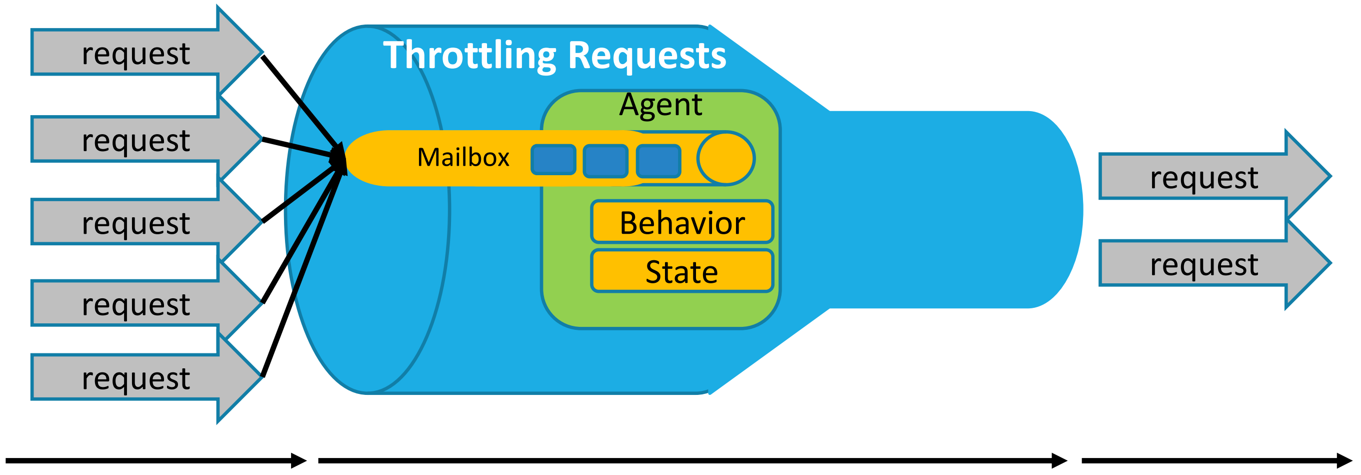




Multiple application requests for Database

Contention due to a limited number of available connections, causing waits and queuing of the requests

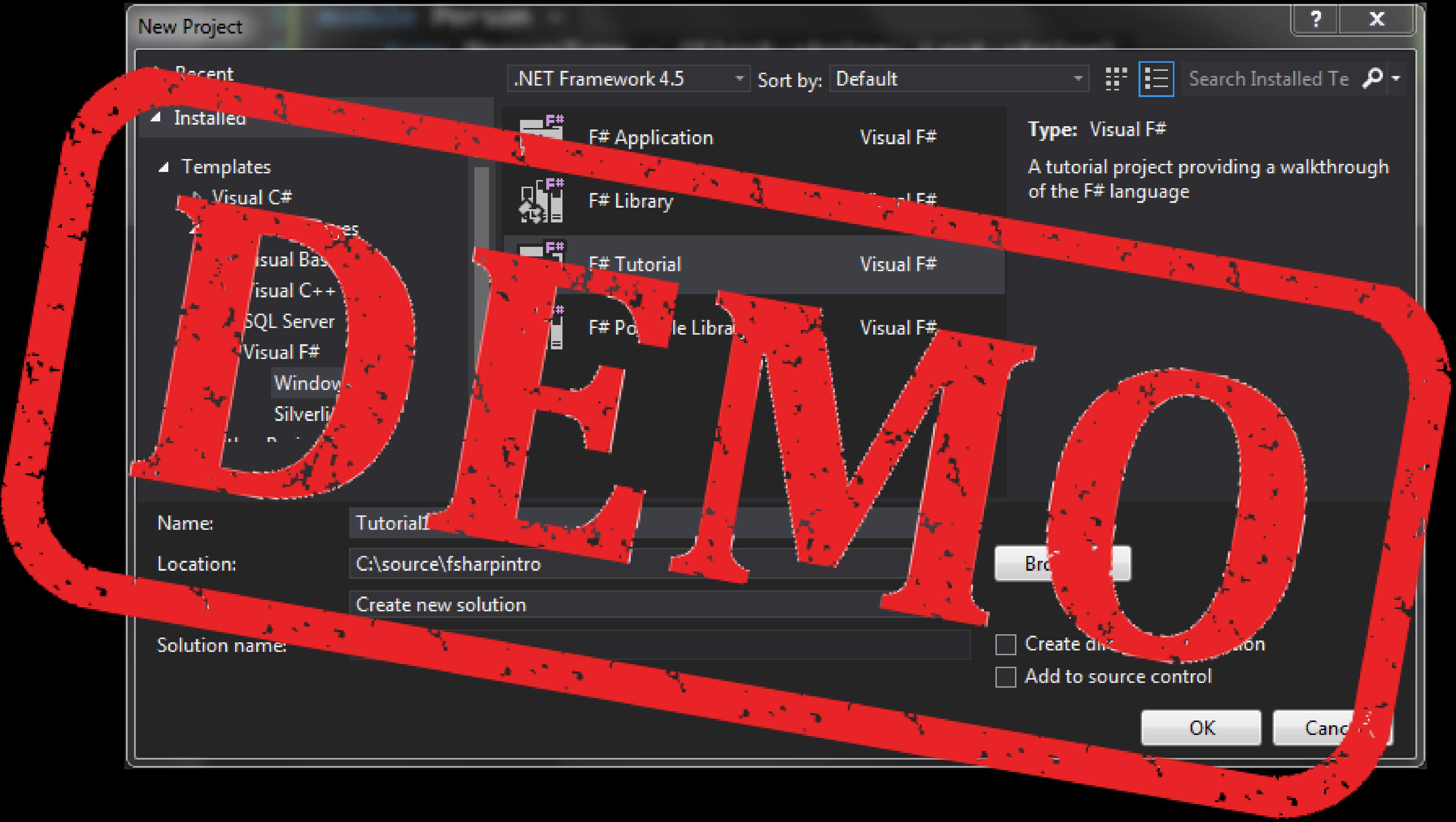
The number of concurrent requests is reduced



Multiple Application requests for Database

The Actor buffering and enqueing the incoming requests ensuring a granular controller over the Database connection pool

The number of concurrent requests is tamed



New Project

Recent
Installed
Templates
Visual C#
Visual Basic
Visual C++
SQL Server
Visual F#
Windows
Silverlight

.NET Framework 4.5
Sort by: Default
Search Installed Te

F# Application	Visual F#
F# Library	Visual F#
F# Tutorial	Visual F#
F# Portable Library	Visual F#

Type: Visual F#

A tutorial project providing a walkthrough of the F# language

Name: Tutorial1
Location: C:\source\fsharpintro
Create new solution
Solution name:

☐ Create directory for solution
☐ Add to source control

Clustering

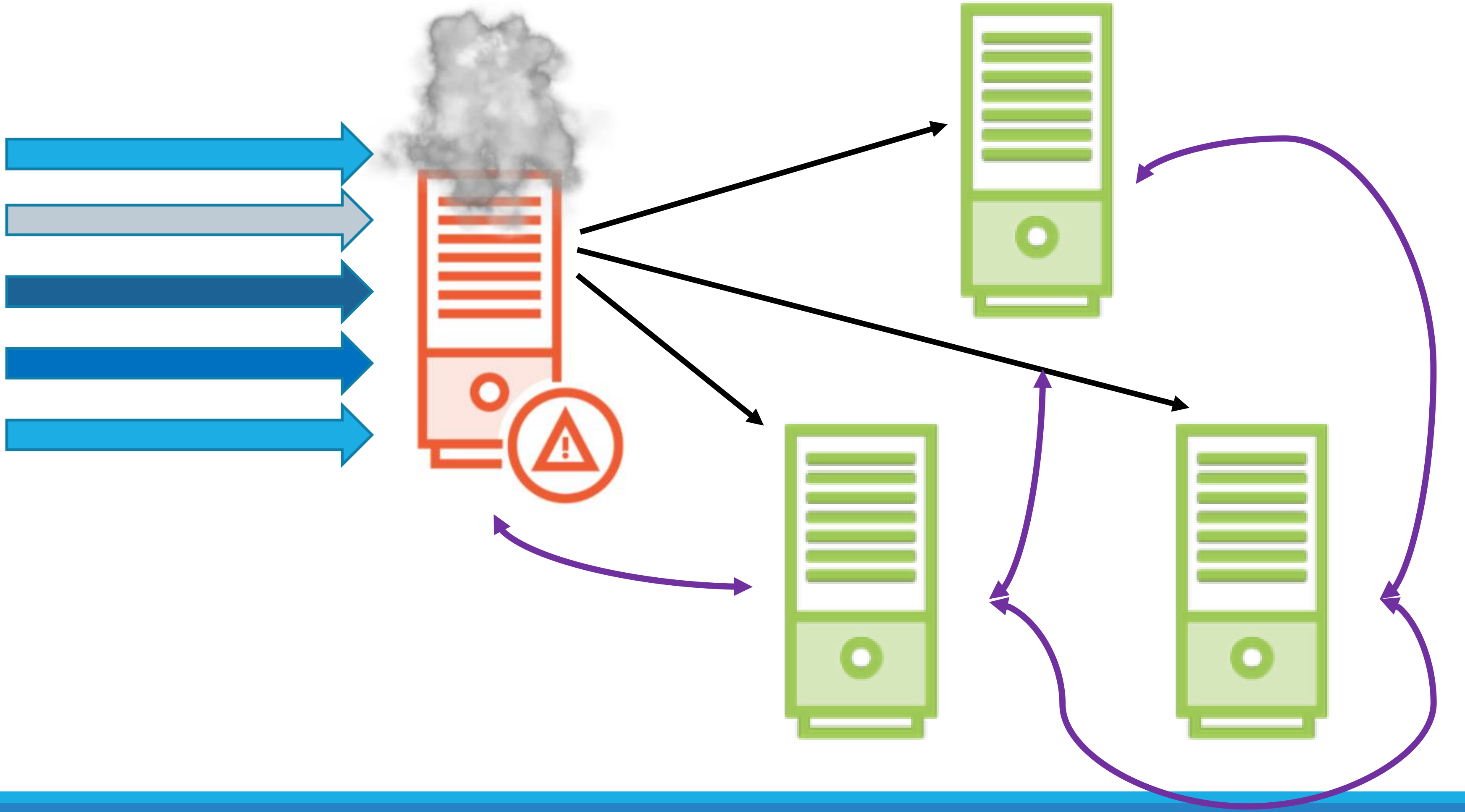
- Load balancing
- Fault-tolerant
- Scalability

Actor Clustering

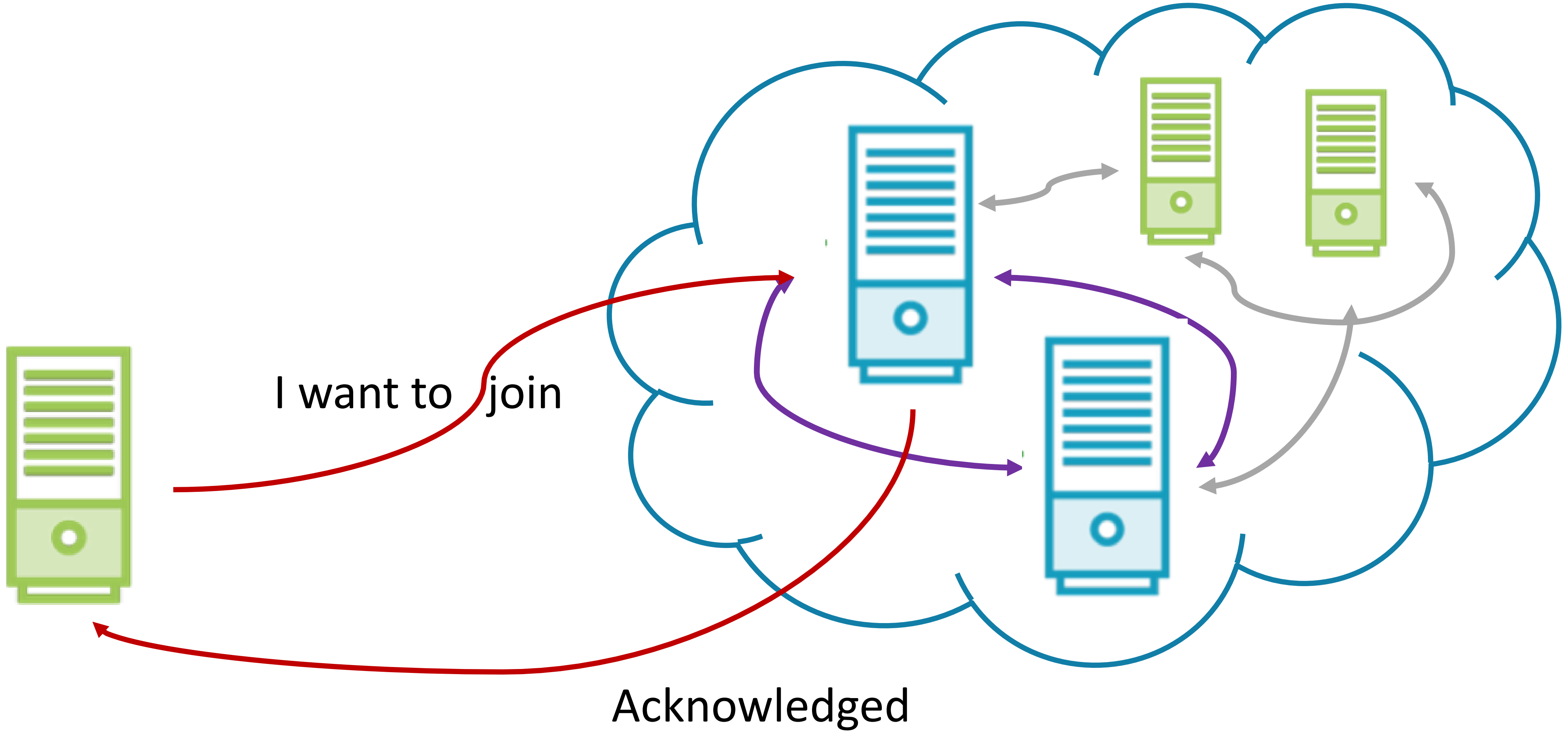
“Anything that can go wrong, will go wrong”

-- Murphy’s Law

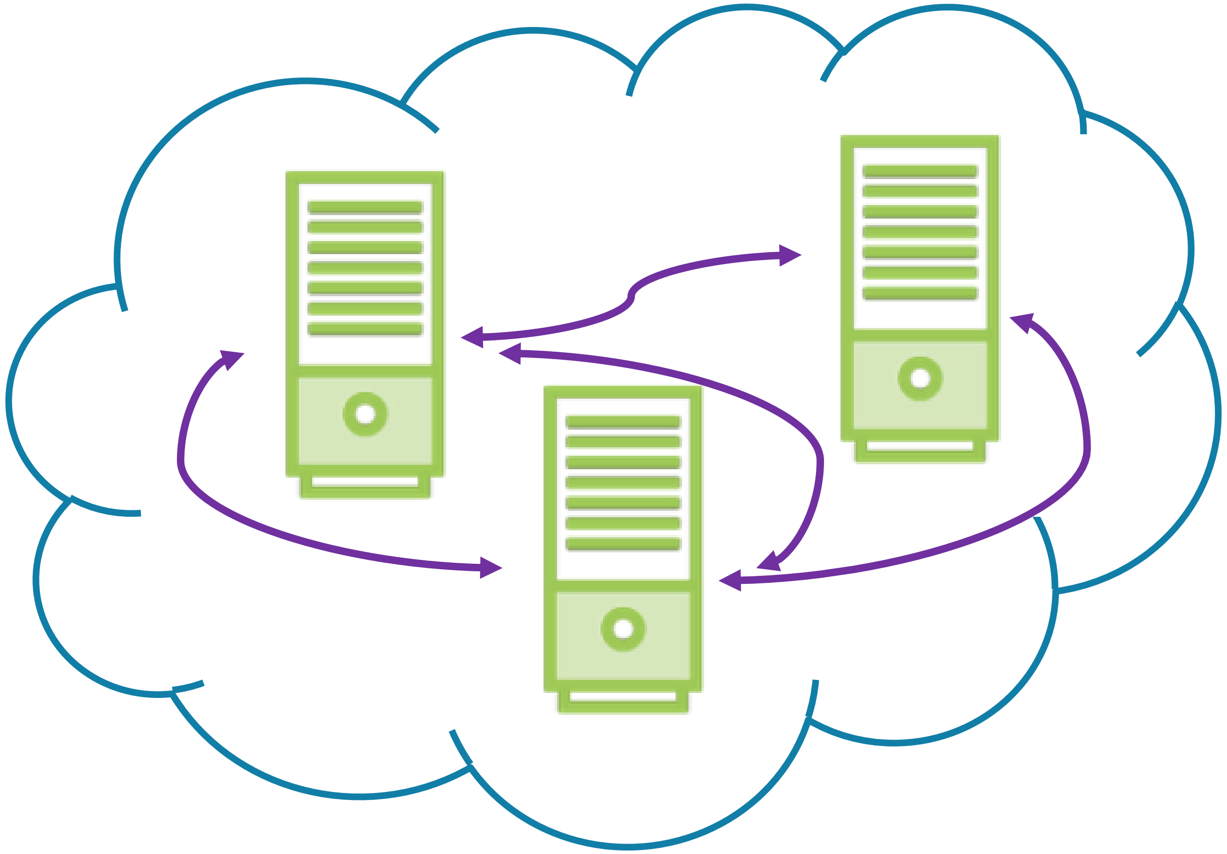
Actor Clustering



Joining the Actor Cluster



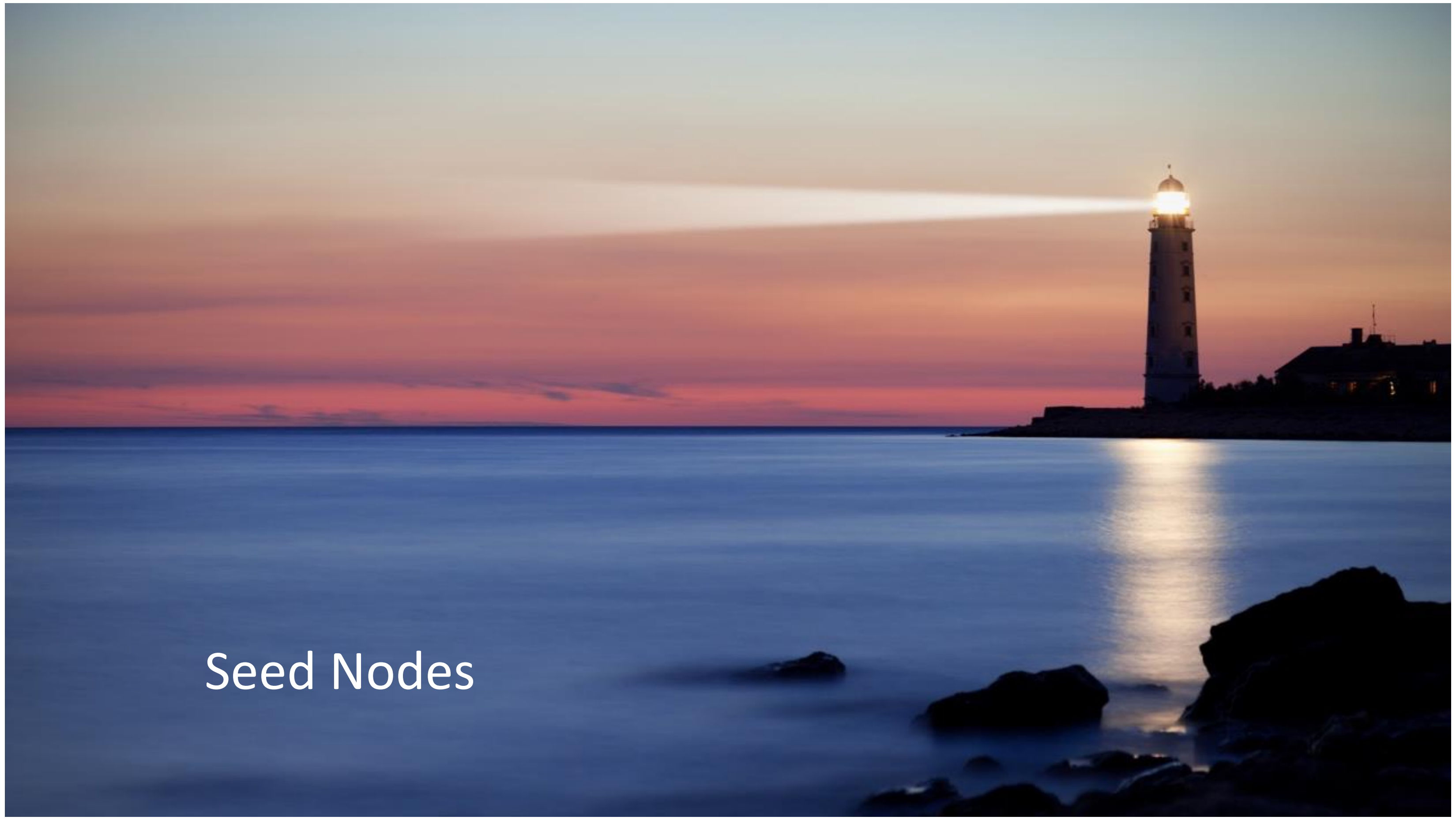
Joining the Actor Cluster

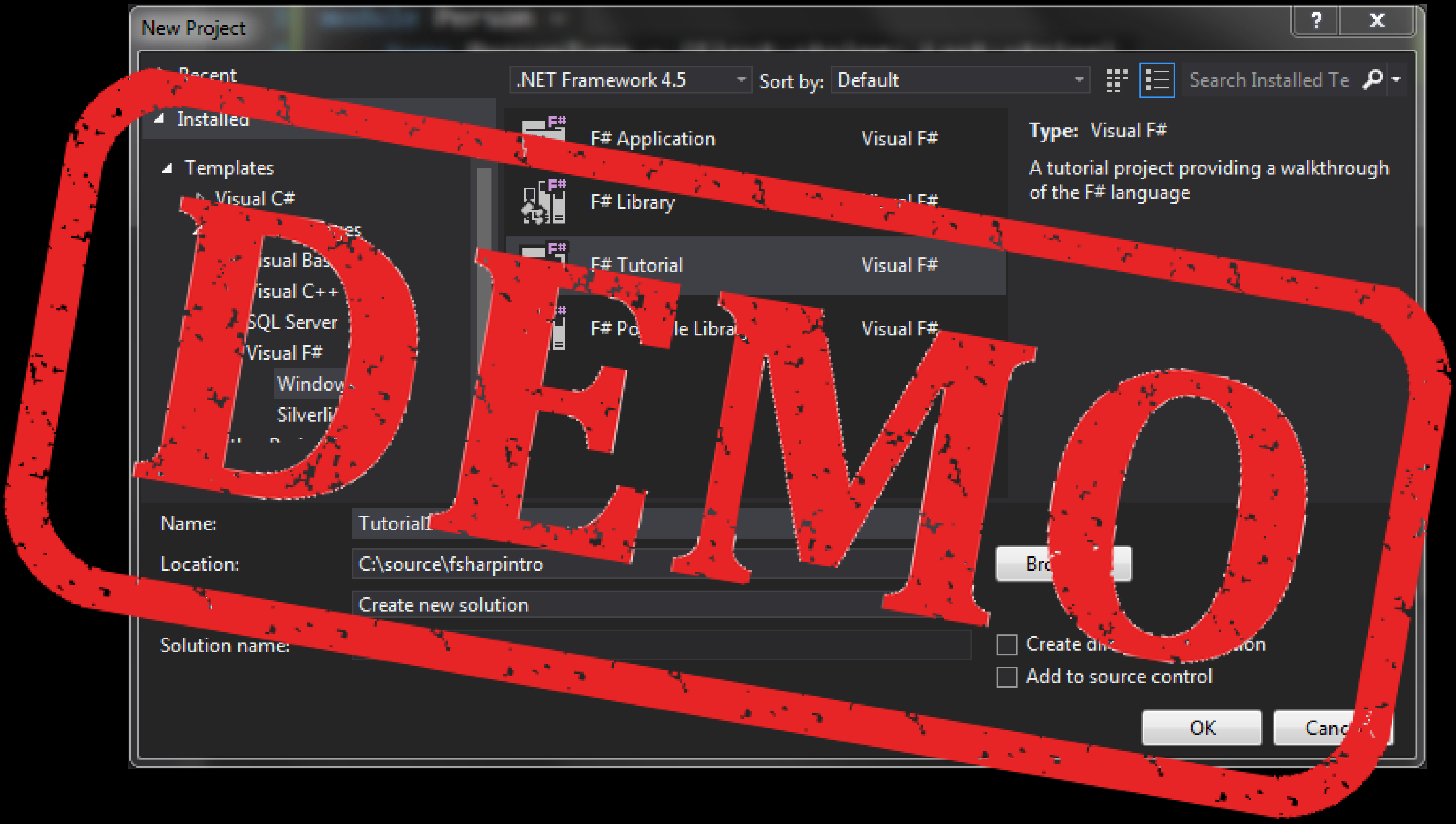


Gossip

Lighthouse

Lighthouse dedicated is a simple but effective seed node





.NET Framework 4.5

Sort by: Default

Type: Visual F#

A tutorial project providing a walkthrough of the F# language

Name: Tutorial1

Location: C:\source\fsharpintro

Create new solution

Solution name:

The tools we use have a profound (and devious!) influence on our thinking habits, and, therefore, on our thinking abilities.

-- Edsger Dijkstra