arcadia
software country

Akka.NET

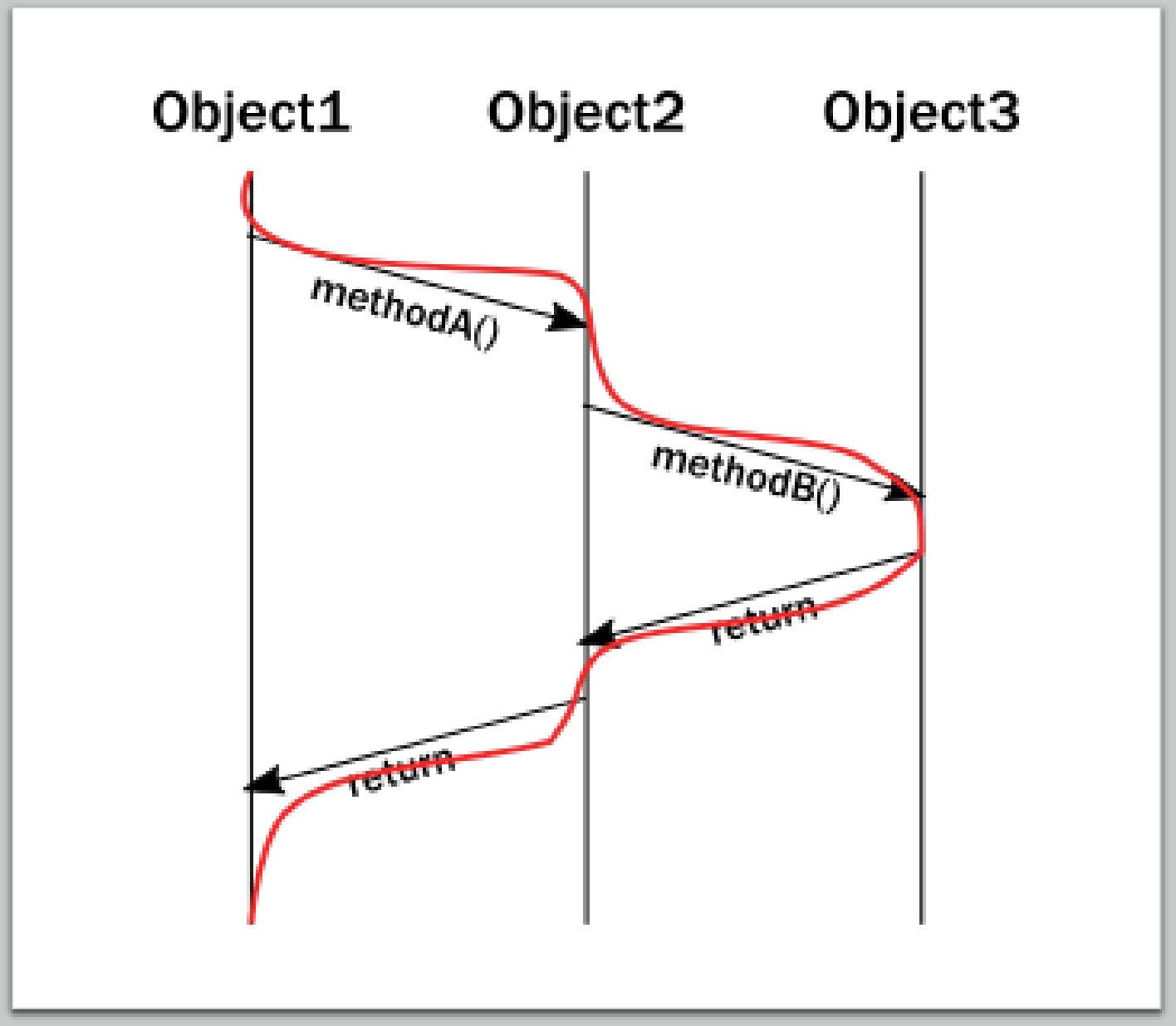
Alexander Shevnin

TECHNOLOGY BY PEOPLE
FOR BUSINESS

Myself

- C++, C#, TypeScript, ASP.NET, Angular
- Delivery Manager
- In Arcadia since 2012
- https://github.com/santee

Classical model



https://getakka.net/articles/intro/what-are-actors.html

What about concurrency?

- Objects can only guarantee encapsulation in the face of single-threaded access
- Locks are bad.
- Distributed locks are nightmare

Top Rock'n'Roll Hits

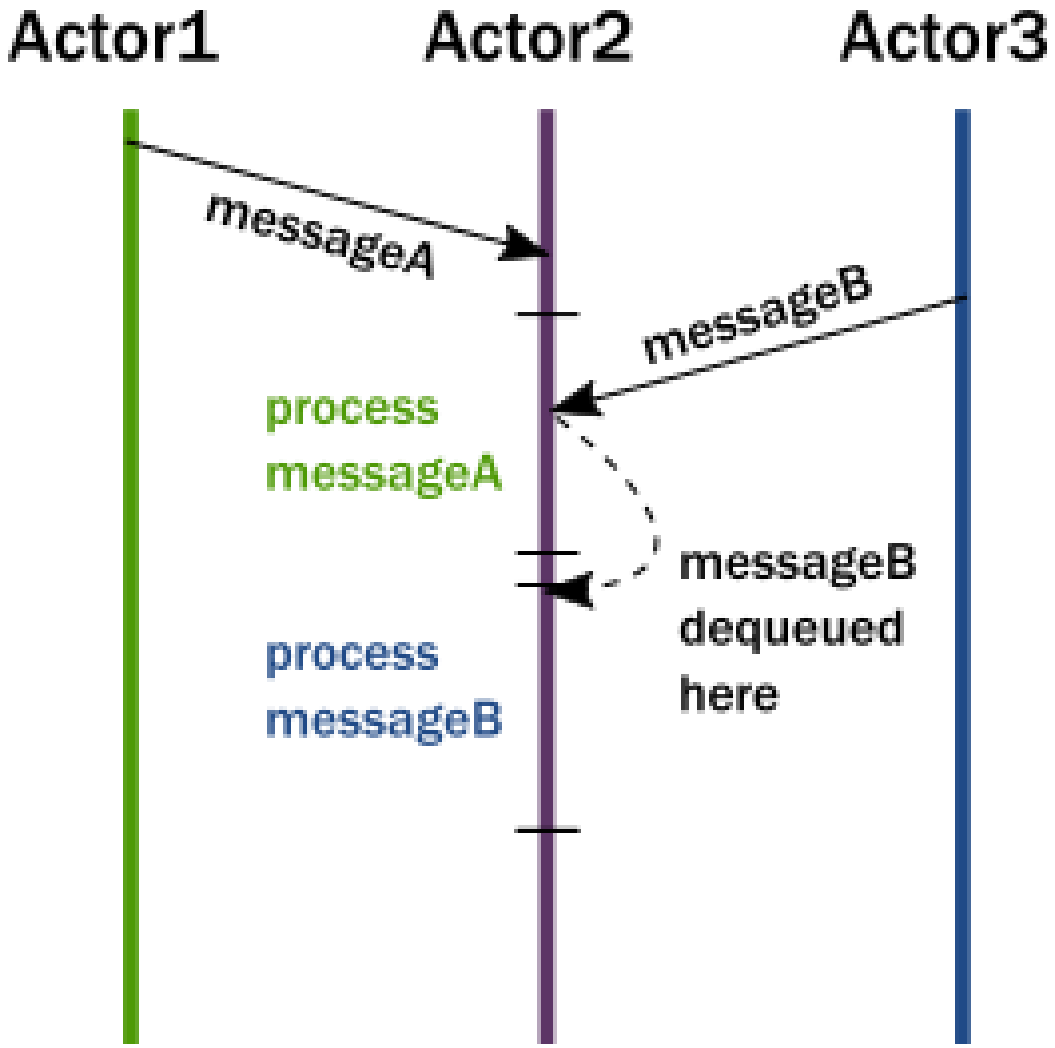
Presented By Joel Whitburn Author Of Pop Annual

1	**Crocodile Rock** Elton John	6	**Will It Go Round In Circles** Billy Preston

Actor

- Receives/sends messages
- Can create child actors
- Maintains its own state
- Can only affect each other through messages
- Location transparency


CARLY RAE JEPSEN
Call Me Maybe



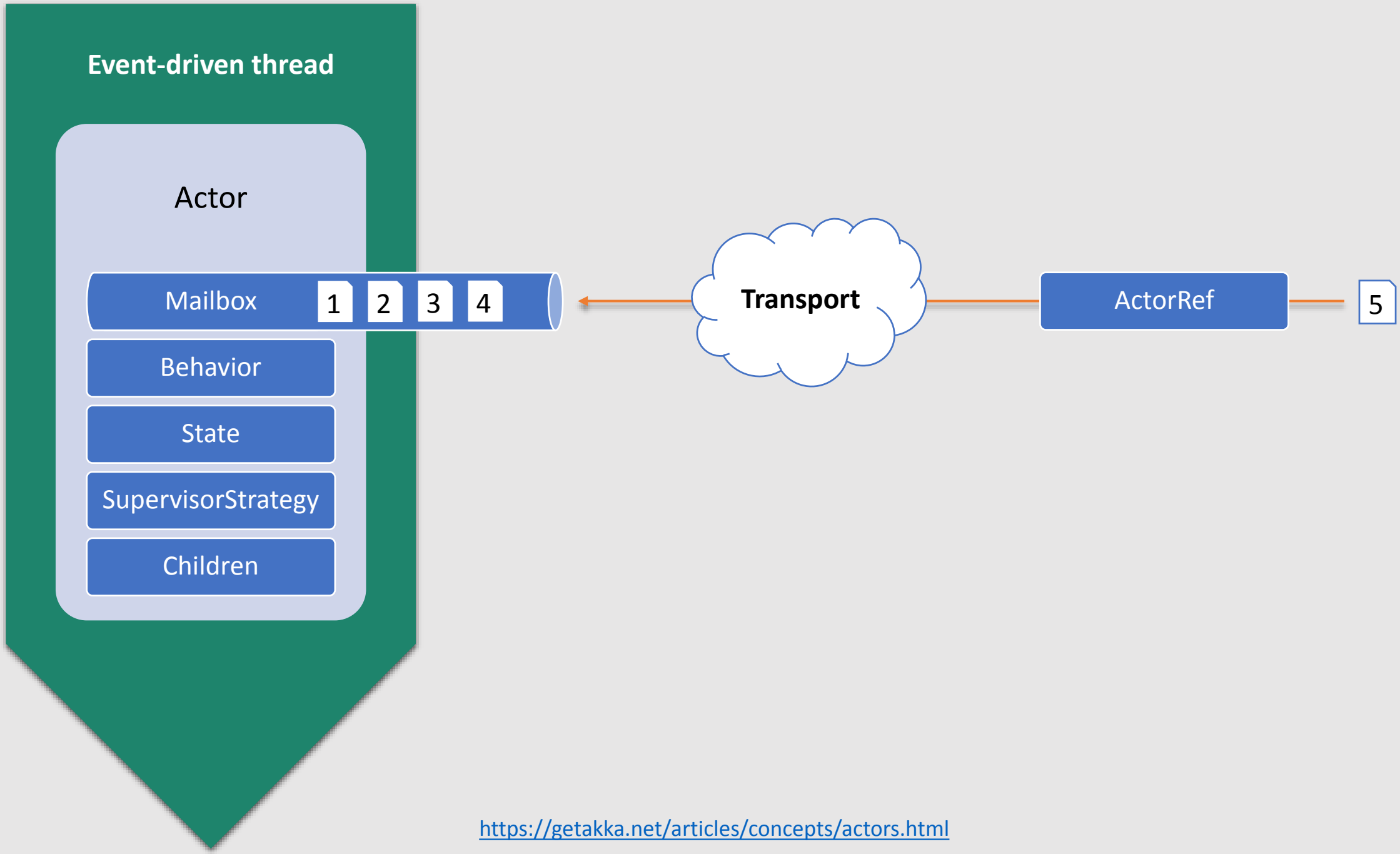
https://getakka.net/articles/intro/what-are-actors.html

Usages

- Erlang
  - RabbitMQ
  - CouchDB
- Akka / Scala
  - LinkedIn
  - Walmart
  - Blizzard

Akka.NET

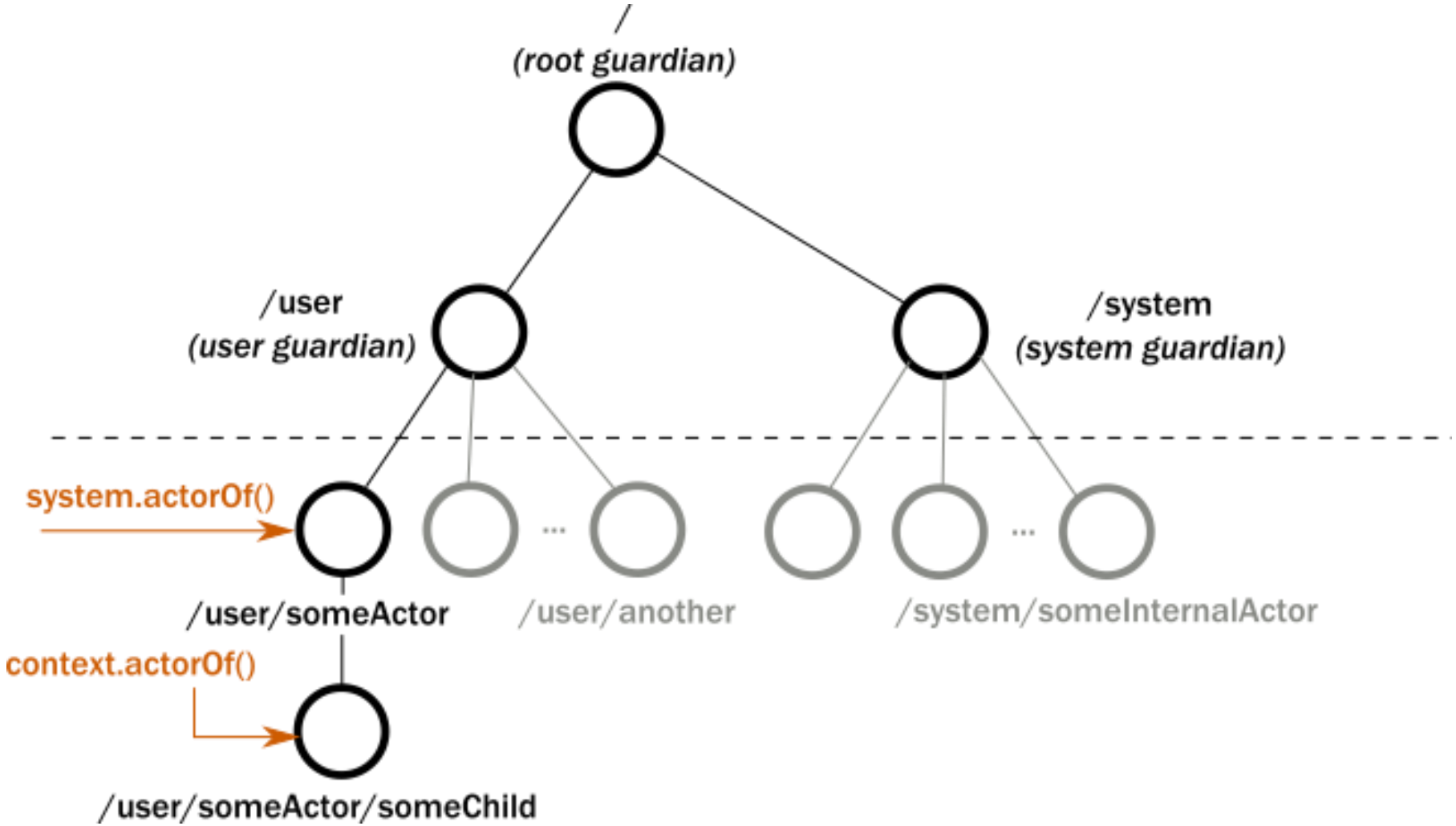
- Actors model for .NET
- A port of Akka (JVM-based Scala framework)
- 2013-2014
- Petabridge - https://petabridge.com/about/



https://getakka.net/articles/concepts/actors.html

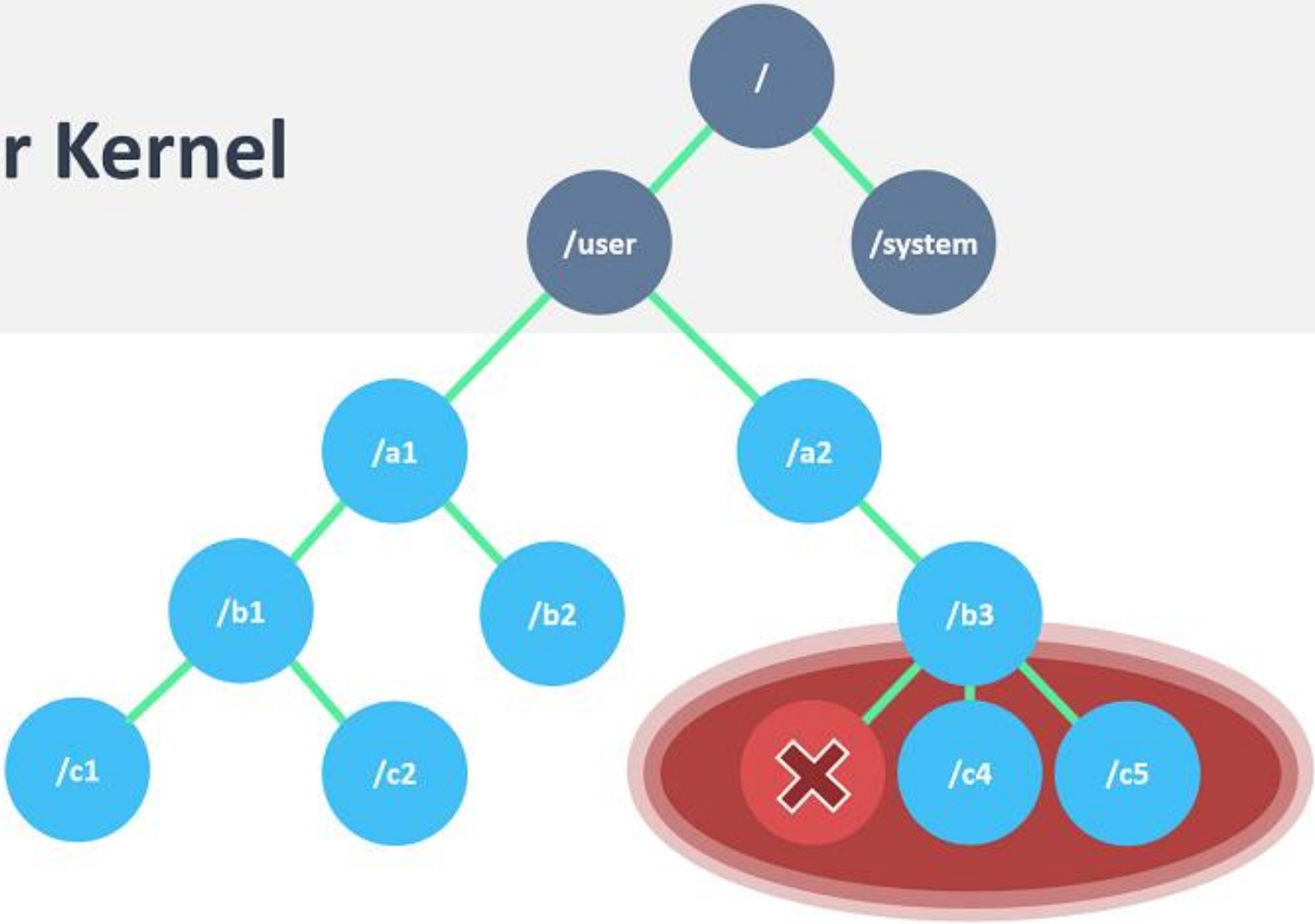
```
public class PrintMyActorRefActor : UntypedActor
{
    protected override void OnReceive(object message)
    {
        switch (message)
        {
            case PrintSomething msg:
                Console.WriteLine(msg.Text);
                this.Sender.Tell('I made dis!');
            break;
        }
    }
}
```

```
var props = Props.Create(() => new PrintMyActorRefActor());

var firstRef = Sys.ActorOf(props, "first-actor");
Console.WriteLine($"First: {firstRef}");


firstRef.Tell(new PrintSomething("Hello World"));
var result = await firstRef.Ask<string>(new PrintSomething("repeat!"),
                                        timeout,
                                        cancellationToken);
```



https://getakka.net/articles/intro/tutorial-1.html

Error Kernel

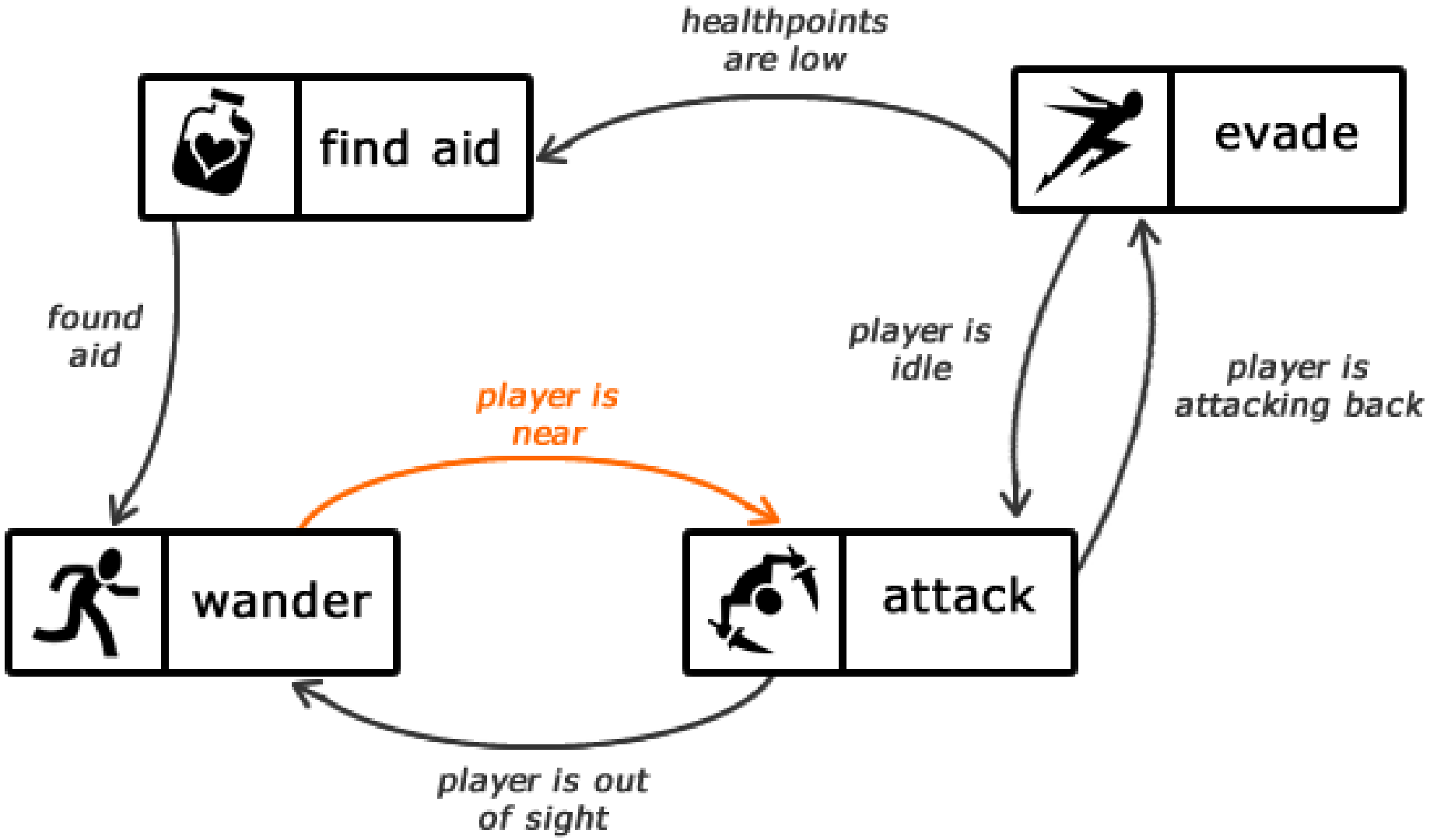


https://getakka.net/articles/concepts/supervision.html

Benefits

- Encapsulation is preserved by decoupling execution from signaling
- No need for locks – messages are processed one at a time
- State is truly private
- Failure is expected and dealt with via messages

Finite State Machine



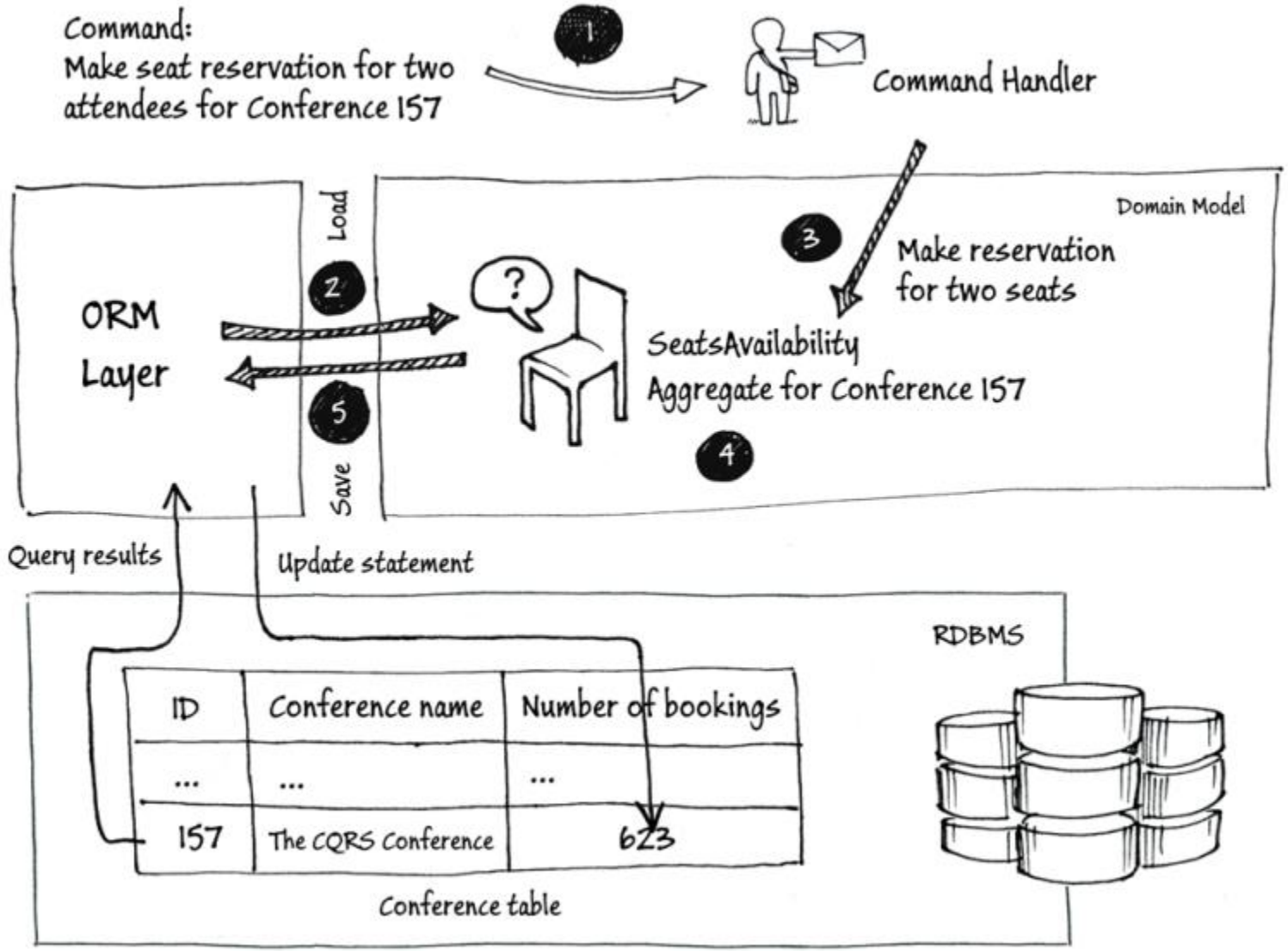
https://gamedevelopment.tutsplus.com/tutorials/finite-state-machines-theory-and-implementation--gamedev-11867

```
public class MyActor : UntypedActor, IWithUnboundedStash
{
    protected override void OnReceive(object message)
    {
        switch (message)
        {
            case OtherMsg _:
                //do smthg
                break;

            case SwitchMe _:
                Context.Become(OtherBehavior);
                break;
            default:
                this.Unhandled();
                break;
        }
    }
    //...
```

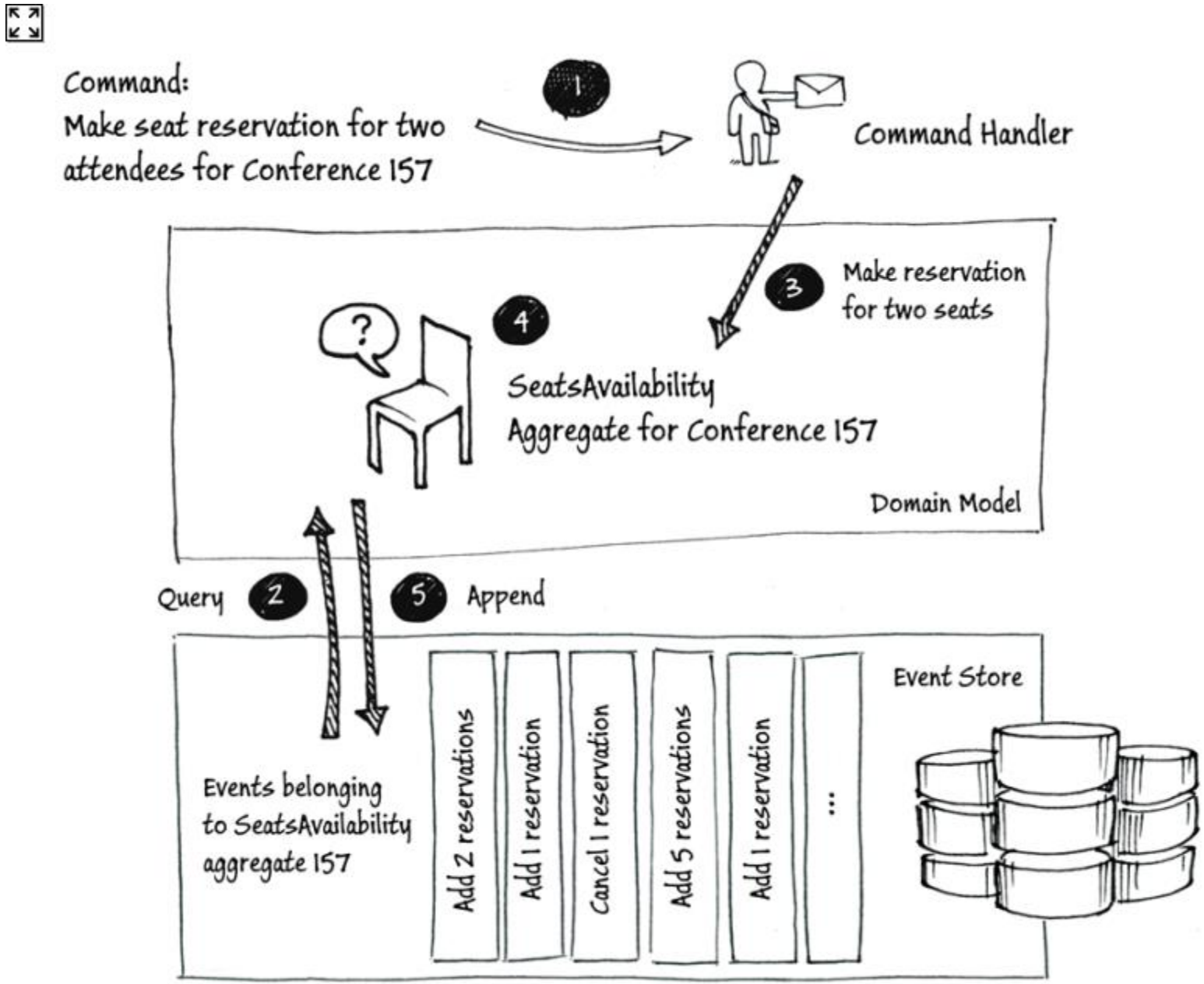
```
public IStash Stash { get; set; }

private void OtherBehavior(object message)
{
    if (message is SwitchMeBack)
    {
        // switch back to previous behavior on the stack
        this.Stash.UnstashAll();
        Context.Unbecome();
    }
    else
    {
        this.Stash.Stash();
    }
}
```

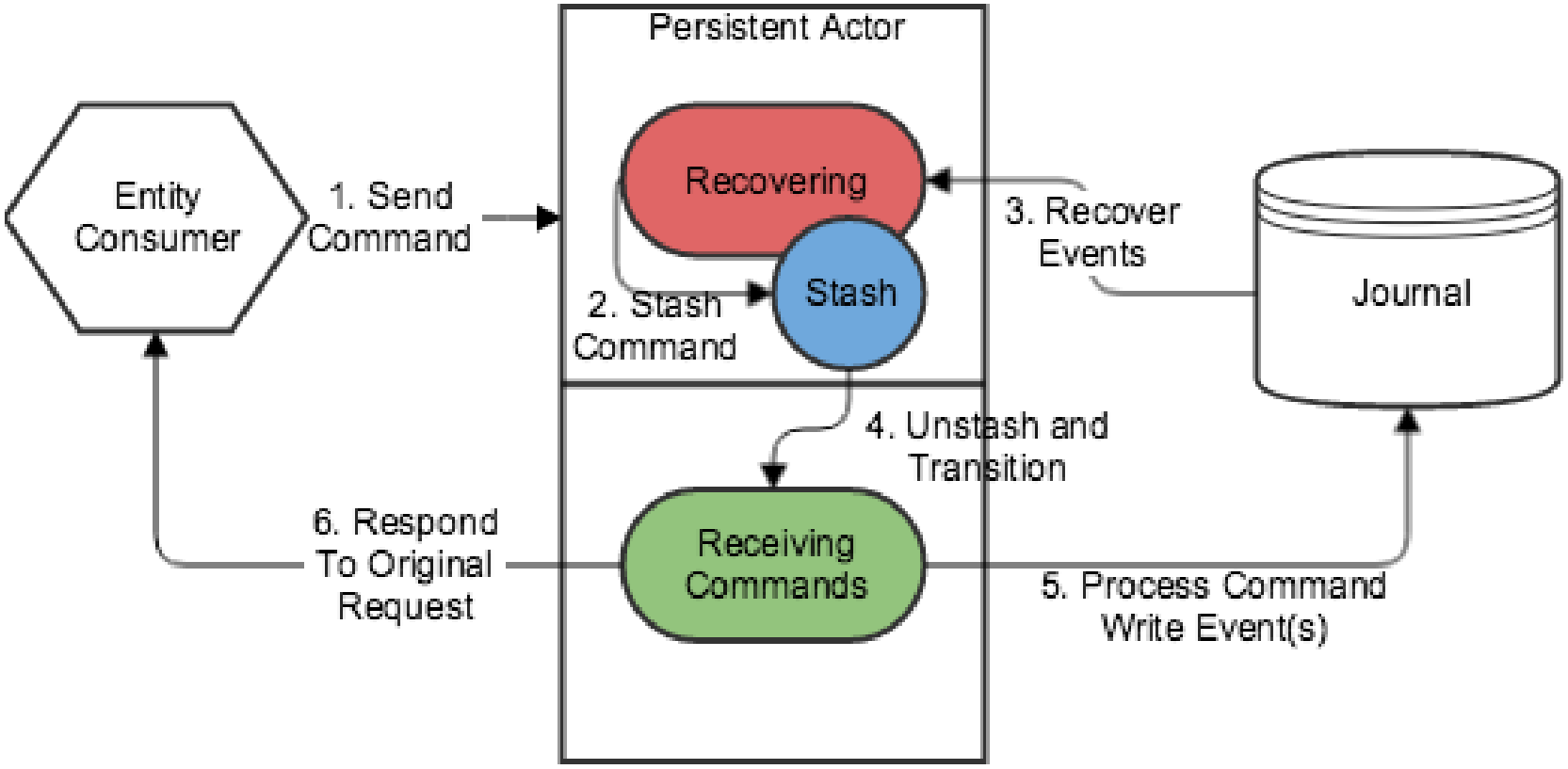
```
private UntypedRecieve PrintAndWaitForResponse(string printMe)
{
    this.otherActor.Tell(new PrintSomething(printMe));

    void OnMessage(object message)
    {
        switch (message)
        {
            case MessagePrinted _:
                Context.Become(SomethingElse);
                break;
            default:
                break;
        }
    }

    return OnMessage;
}
```

Persistence



https://docs.microsoft.com/en-us/previous-versions/msp-n-p/jj591559(v=pandp.10)


Command:
Make seat reservation for two attendees for Conference 157
1
Command Handler
3
Make reservation for two seats
4
SeatsAvailability Aggregate for Conference 157
Domain Model
Query
2
5
Append
Events belonging to SeatsAvailability aggregate 157
Add 2 reservations
Add 1 reservation
Cancel 1 reservation
Add 5 reservations
Add 1 reservation
...
Event Store



https://hub.packtpub.com/making-history-event-sourcing/

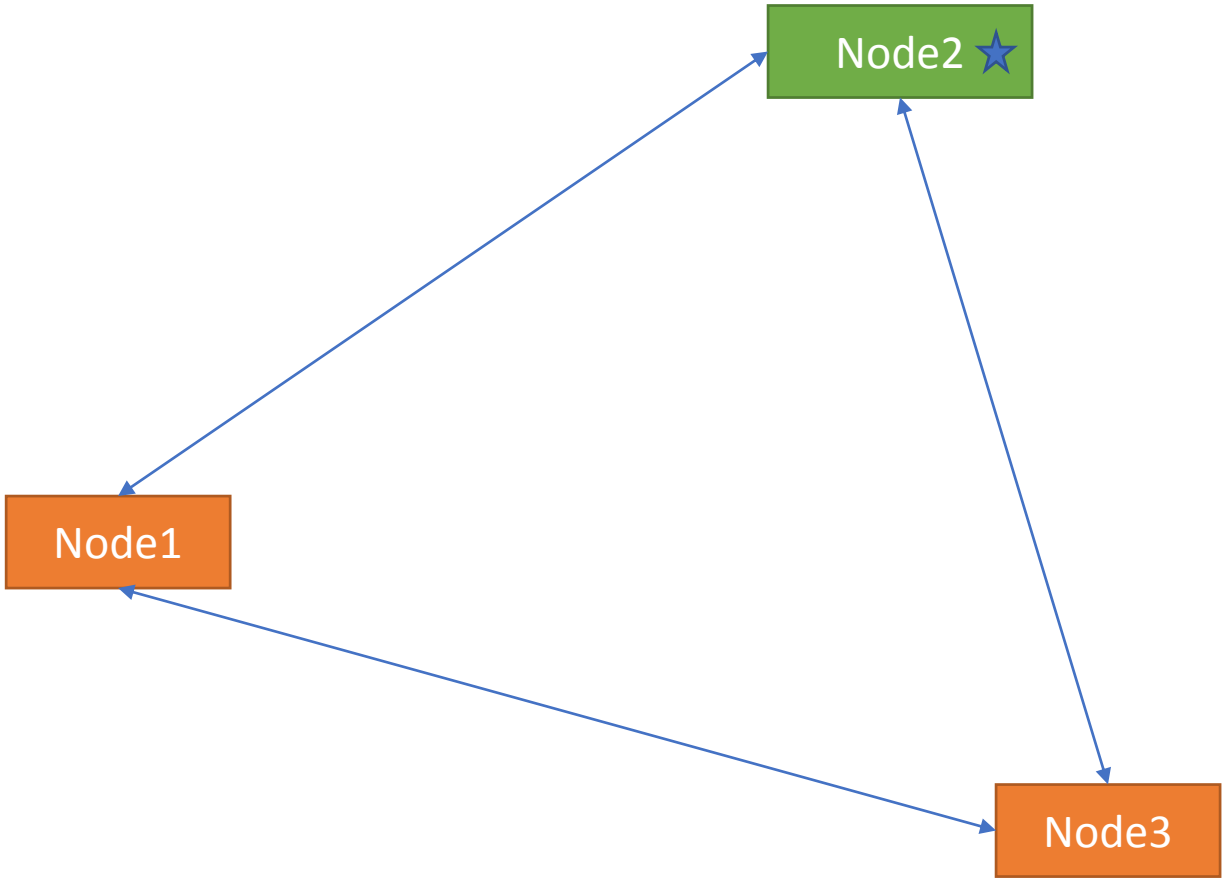
```
public class ReservationsActor : UntypedPersistentActor
{
    private ReservationsState state = new ReservationsState();

    private void OnSeatReserved(SeatReserved evt)
    {
        state = state.Updated(evt);
    }

    protected override void OnRecover(object message)
    {
        switch (message)
        {
            case SeatReserved evt:
                this.OnSeatReserved(evt);
                break;
        }
    }
    //...
```

```
protected override void OnCommand(object message)
{
    switch (message)
    {
        case ReserveSeat cmd:
            this.notificationsActor.Tell(new SendReservationNotification(cmd));
            this.Persist(new SeatReserved(cmd.Seat), this.OnSeatReserved);
            break;
    }
}
```

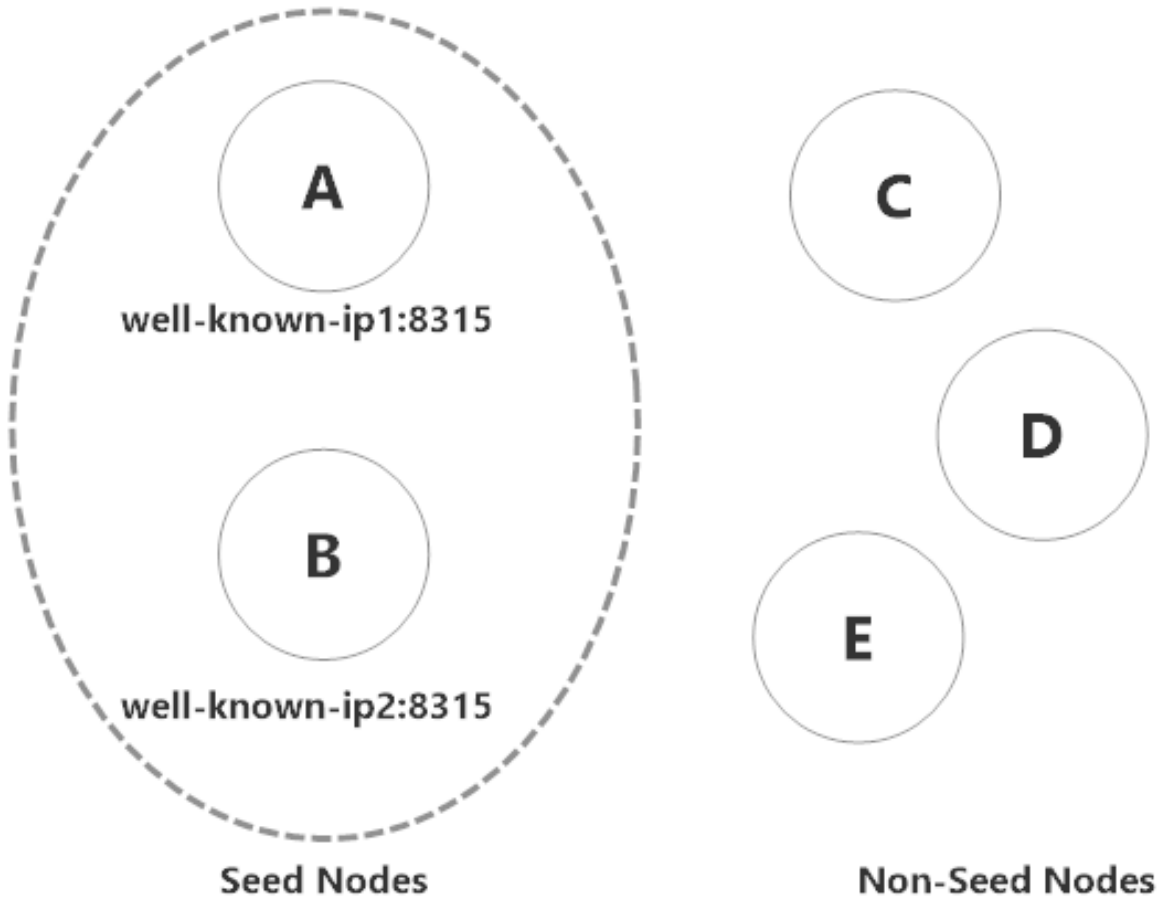
```
case SnapshotOffer snapshot when snapshot.Snapshot is ReservationsState:
     this.state = (ReservationsState)snapshot.Snapshot;
     break;
```

- Event adapters
  - version migrations
  - separate domain and data models
- At-least-once delivery
- Databases support (found in Nuget)
  - In Memory
  - RavenDB
  - Marten
  - Postgresql
  - MongoDB
  - SQLServer
  - AzureTable
  - Generic ADO.NET
  - SQLite
  - Redis
  - Cassandra
  - DocumentDB
  - MySQL
  - Oracle

Akka Clustering

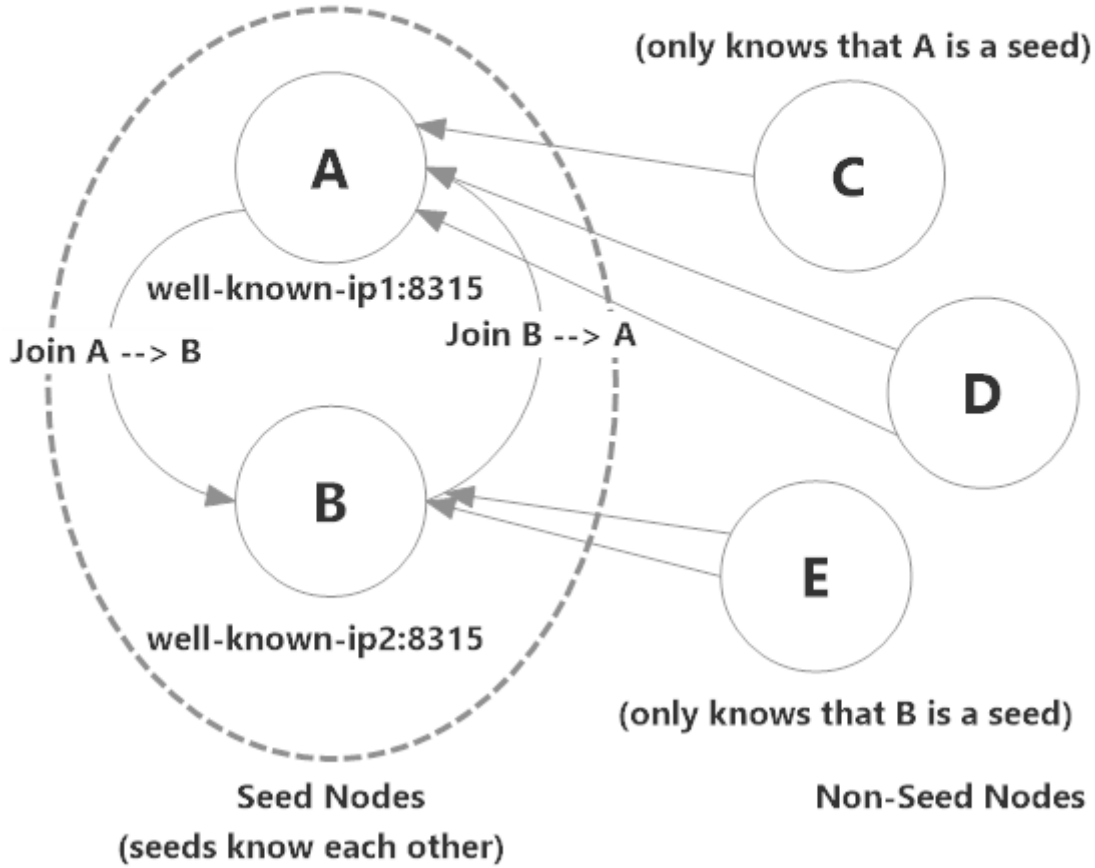

Node2
Node1
Node3

Initial Cluster State
(Deploying 5 Nodes)



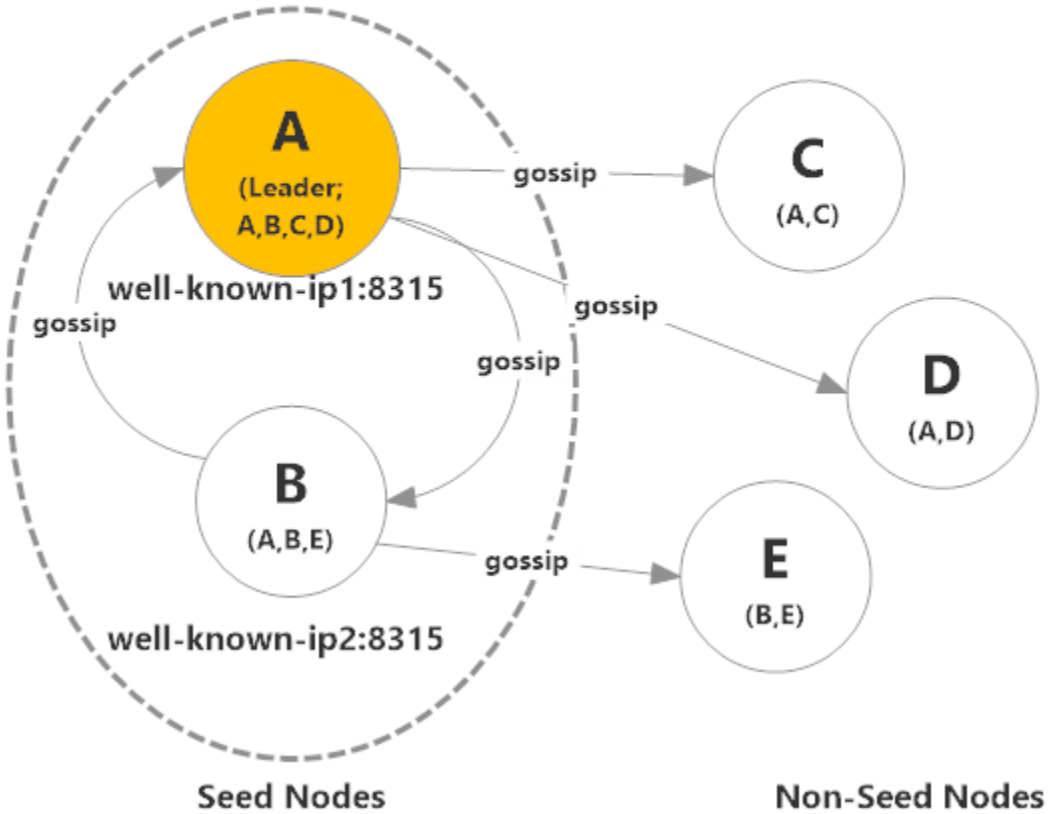
https://getakka.net/articles/clustering/cluster-overview.html

State 1 - Joining the Cluster
(Everyone attempts to Join Seed Nodes)



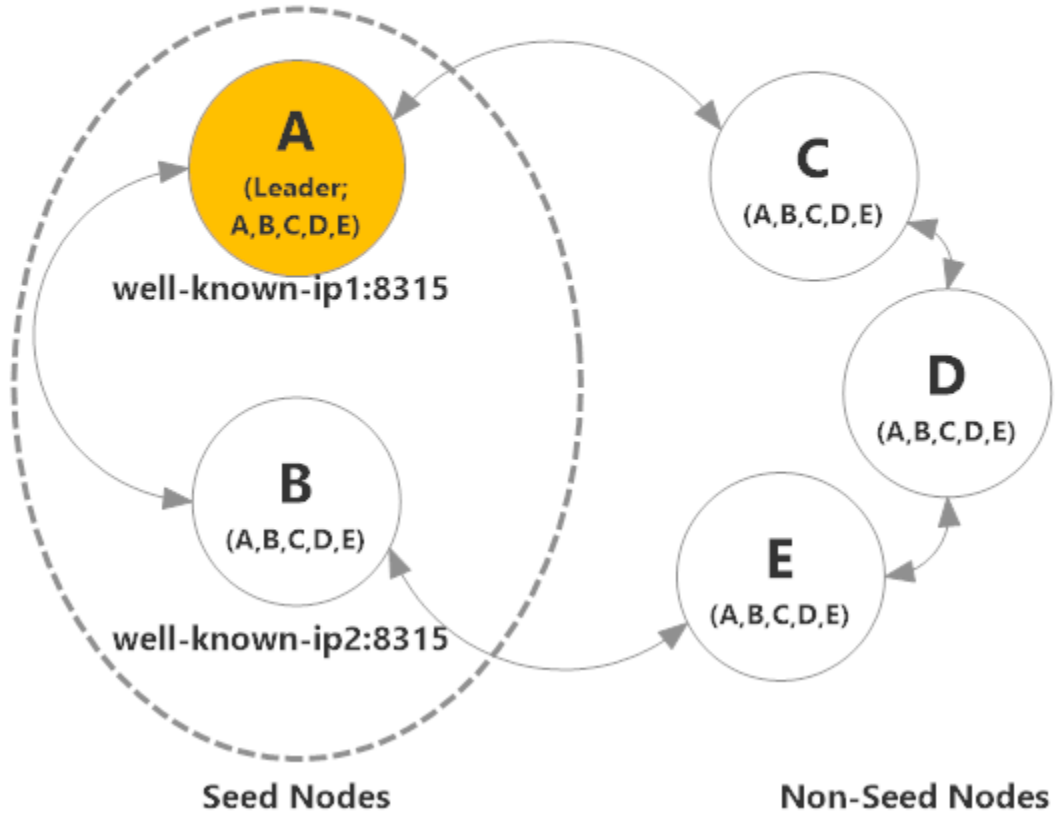
https://getakka.net/articles/clustering/cluster-overview.html

State 2 - Leader Elected, Marking Nodes Up (Gossip Begins)



https://getakka.net/articles/clustering/cluster-overview.html

State 3 - Gossip Spreads, Ring is Formed (Communication Established between Non-Seeds)



https://getakka.net/articles/clustering/cluster-overview.html

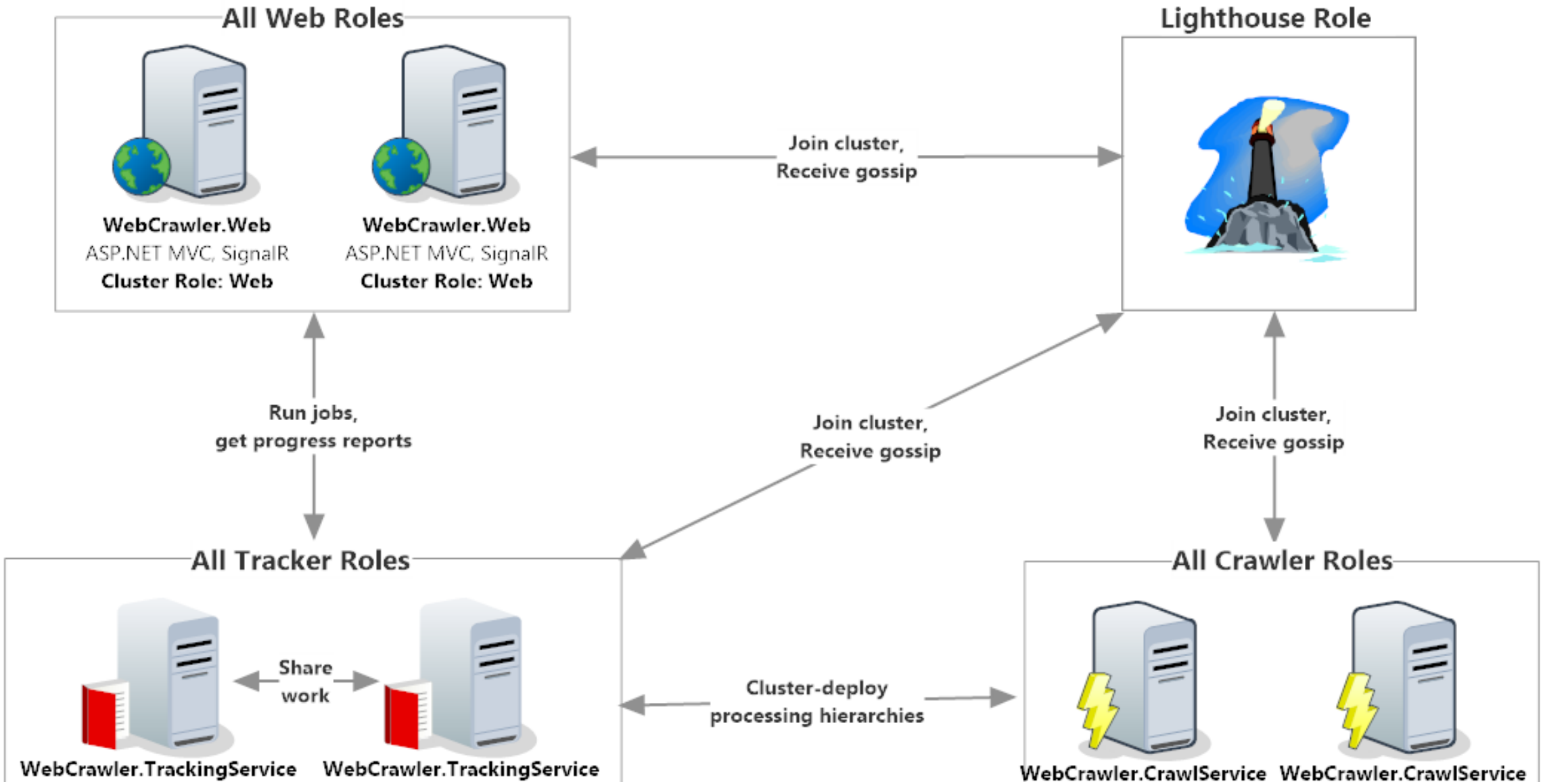
Summary

- Fault-tolerant
- Elastic
- Decentralized
- Peer-to-Peer
- No single point of failure
- Easy microservices

Use-cases

- Analytics Systems
- Marketing Automation
- Multi-player Games
- Device / IoT Tracking
- Alerting & Monitoring Systems
- Recommendation engines
- Dynamic pricing

https://youtu.be/mUTKvGyxbOA?t=294

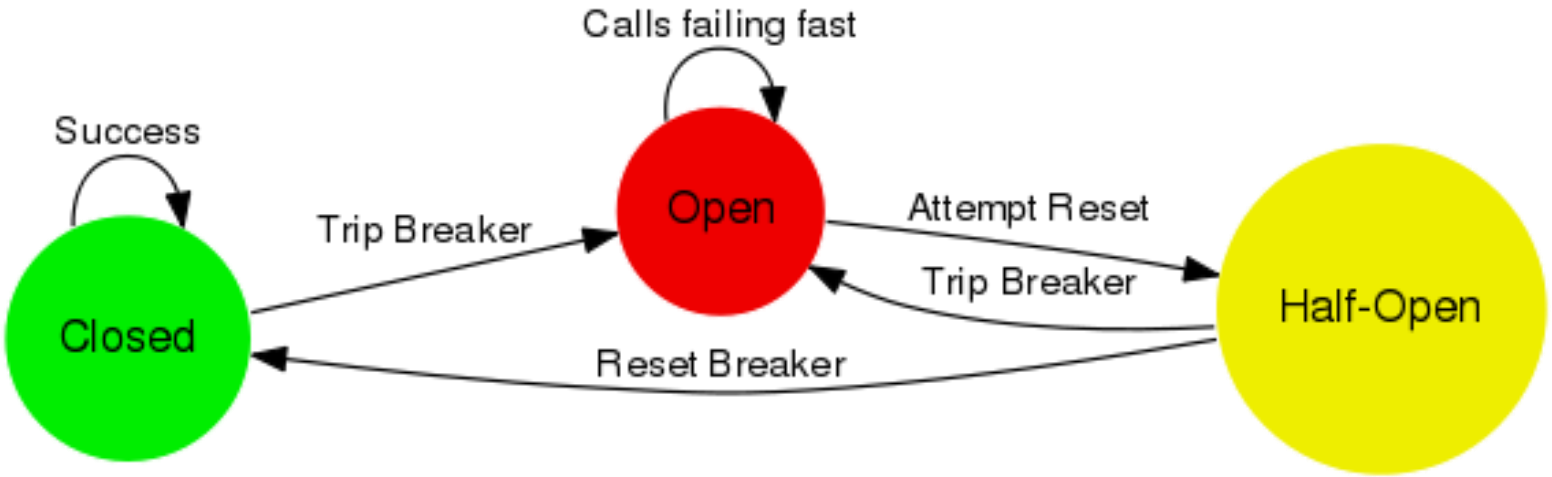


https://github.com/petabridge/akkadotnet-code-samples/tree/master/Cluster.WebCrawler

Additional features

- EventBus
- Logging (+Serilog)
- Scheduler (both in-memory and persistent for long-running tasks)
- Timeouts / cancellations
- Monitoring (AppInsights, performance counters, visualizer)
- Circuit Breaker

Circuit Breaker



https://getakka.net/articles/utilities/circuit-breaker.html

Important details

- Works on
  - .NET Core
  - .NET Framework 4.5+
  - Docker
  - Mono
- DotNetty as a transport
- Protobuf, JSON + custom serializers
- Hyperion (1.5+) (https://github.com/akkadotnet/Hyperion), fork of ‘Wire’

Problems / Gotchas

- Messages have to be serializable
- Props closures have to be serializable
- Lack of type-safety
- Awful Dependency Injection
- Simple things are hard
- Jumping around TPL
- Not easy.

Where do I learn more?

https://getakka.net/

https://petabridge.com/bootcamp/

https://github.com/petabridge/akkadotnet-code-samples/tree/master/Cluster.WebCrawler

https://www.amazon.com/Reactive-Messaging-Patterns-Actor-Model/dp/0133846830

Alternatives

- https://github.com/dotnet/orleans
- https://github.com/akka/akka-meta/blob/master/ComparisonWithOrleans.md
- Service Fabric

QA