Particular Software

Welcome

Async / Await

Chain of Responsibility


Pattern
Build it
WrapUp


Pattern
Build It
WrapUp

OWIN

```
appBuilder.Use(async (ctx, next) =>
{
  // do some things here
  await next();
  // or here
});
```

ASP.NET Core Middleware

```
public class Startup {
    public void Configure(IApplicationBuilder app)    {
        app.Use(async (context, next) =>
        {
            // do some things here
            await next.Invoke();
            // or here
        });
    }
}
```

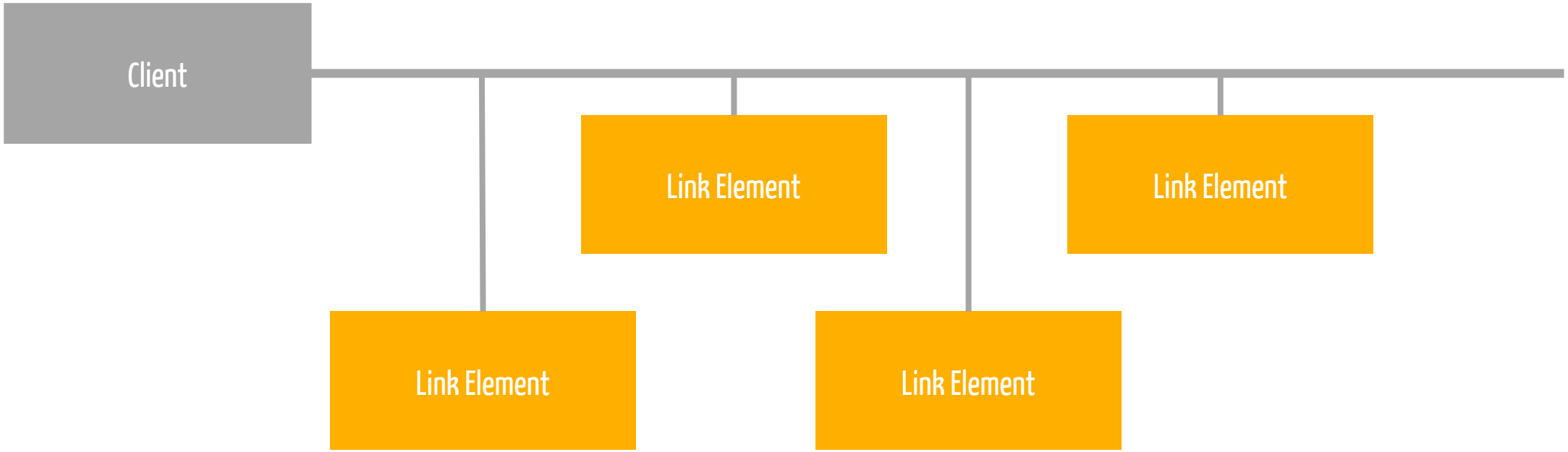
ASP.NET Core Filters

```
class ActionFilter : IAsyncActionFilter  {
 public async Task OnActionExecutionAsync(
   ActionExecutionContext context, ActionExecutionDelegate next)
  {
     // do some things here
     var resultContext = await next();
     // or here
   }
 }
```

Goals

target

Chain of Responsibility


Client
Link Element
Link Element
Link Element
Link Element

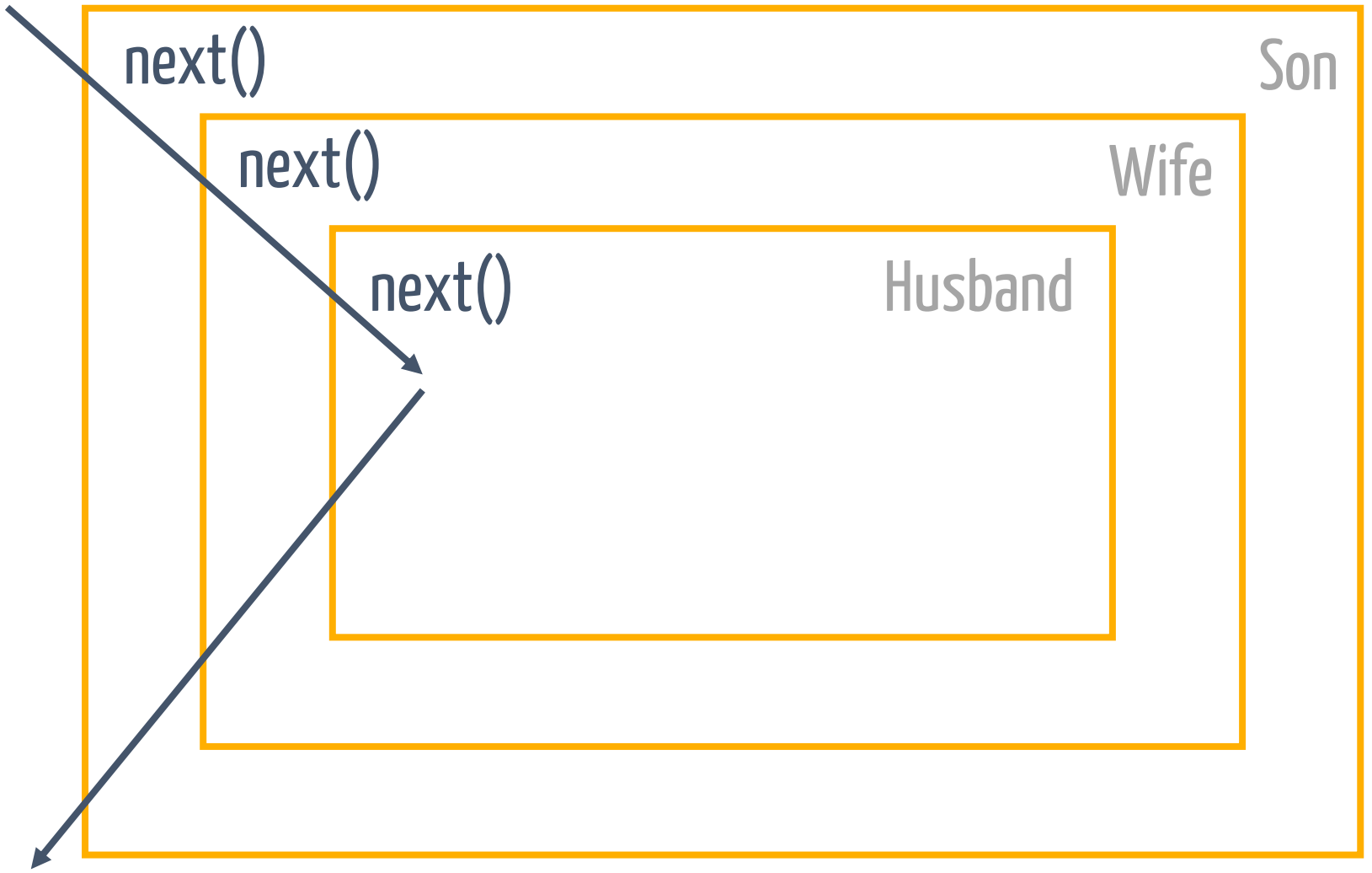
son wife husband

son wife husband

```
static void Person(Action next)
{
    // Implementation
    next();
}
```

```
public void ManualDishwasherUnloading()
{
    Son(() => Wife(() => Husband(() => Done())));
}
```

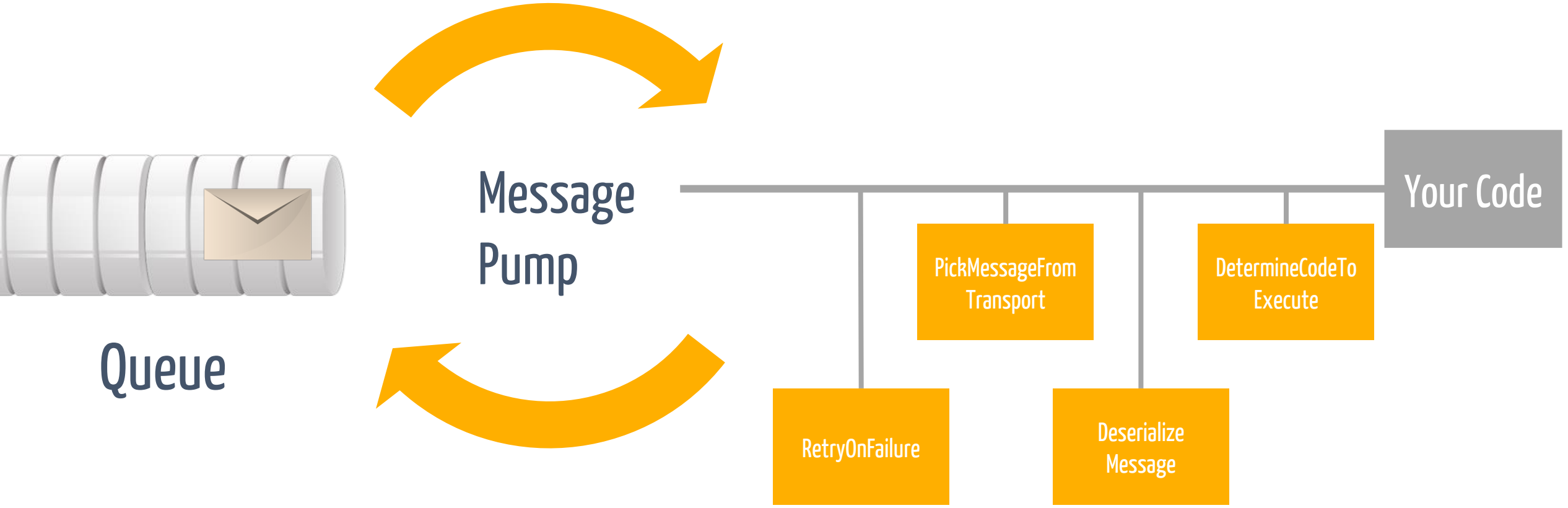
Demo


next()
Son
next()
Wife
next()
Husband

cumbersome

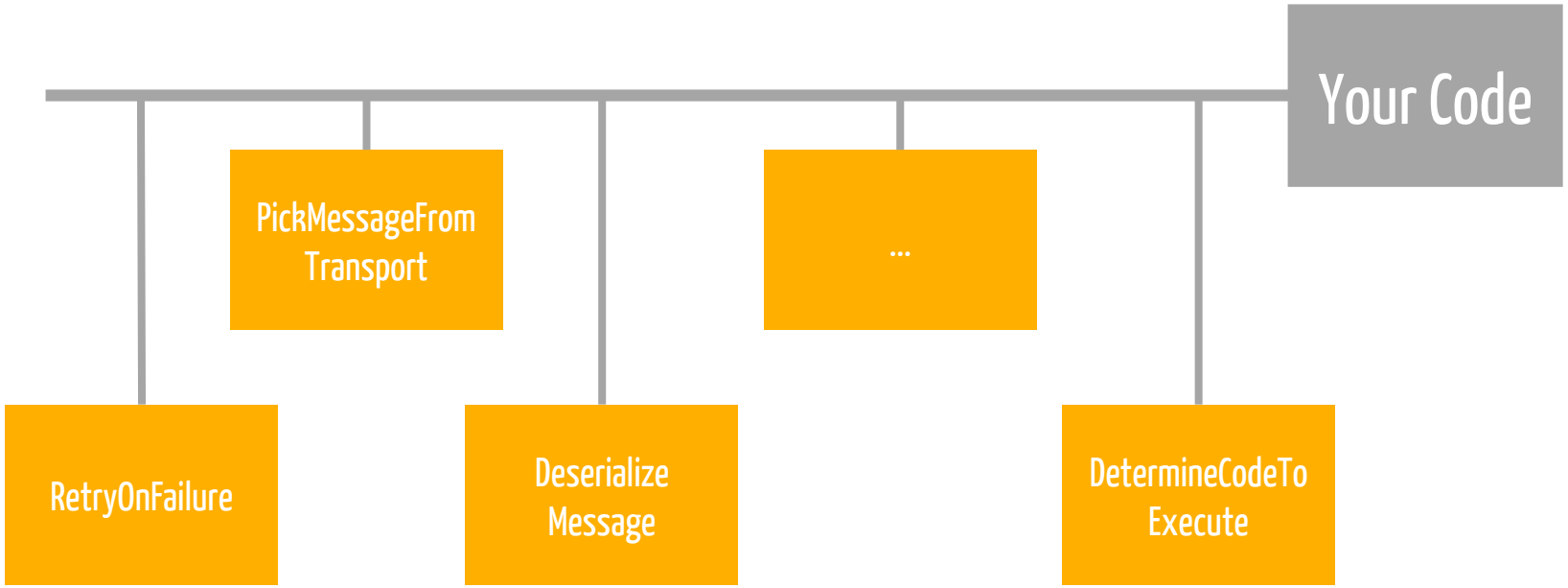
Demo

```
static void IgnoreDishStillWetException(Action next))
{
    try {
        next();
    }
    catch(DishStillWetException) { }
}
```

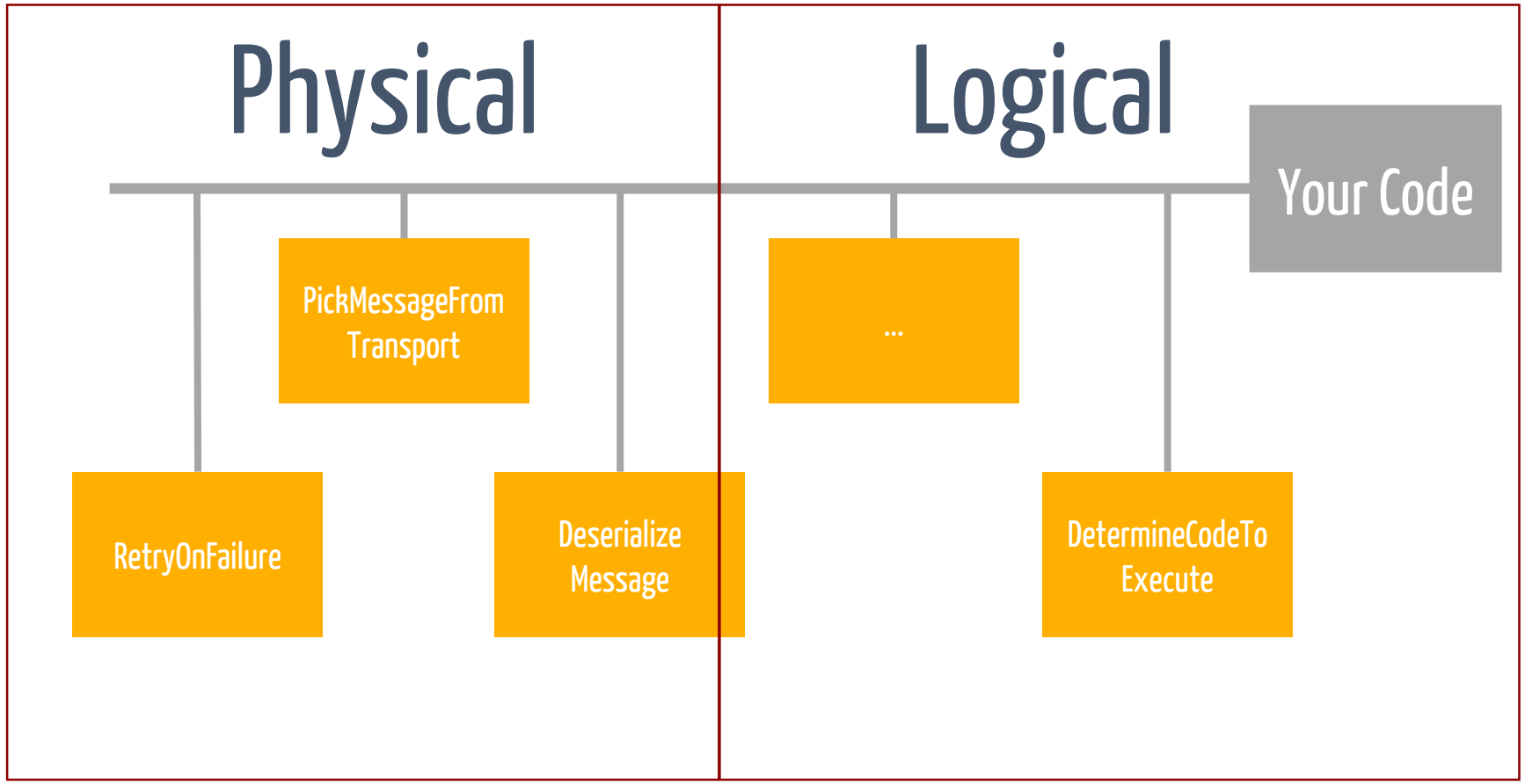

Message
Pump
Queue
Your Code
PickMessageFrom
Transport
DetermineCodeTo
Execute
RetryOnFailure
Deserialize
Message


Pattern
Build It
WrapUp

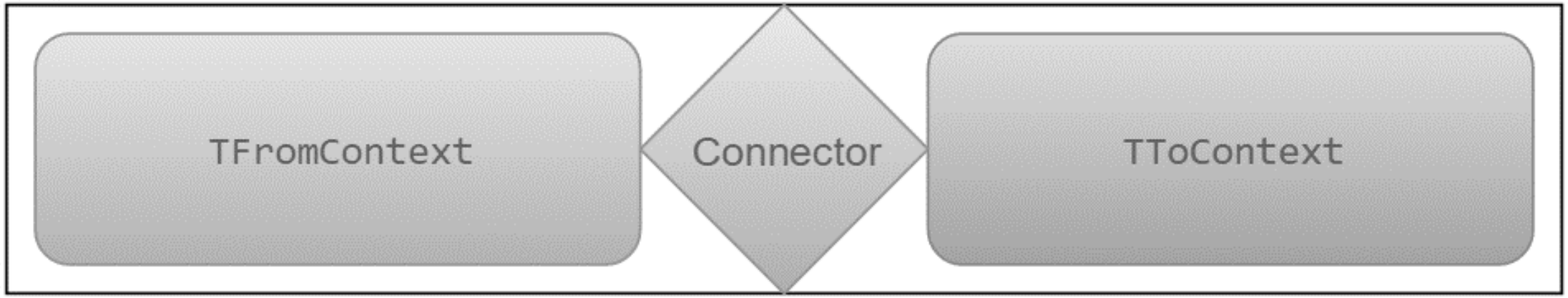
await Demo



Where to place links?


Physical
Logical
Your Code
PickMessageFrom Transport
...
RetryOnFailure
Deserialize Message
DetermineCodeTo Execute

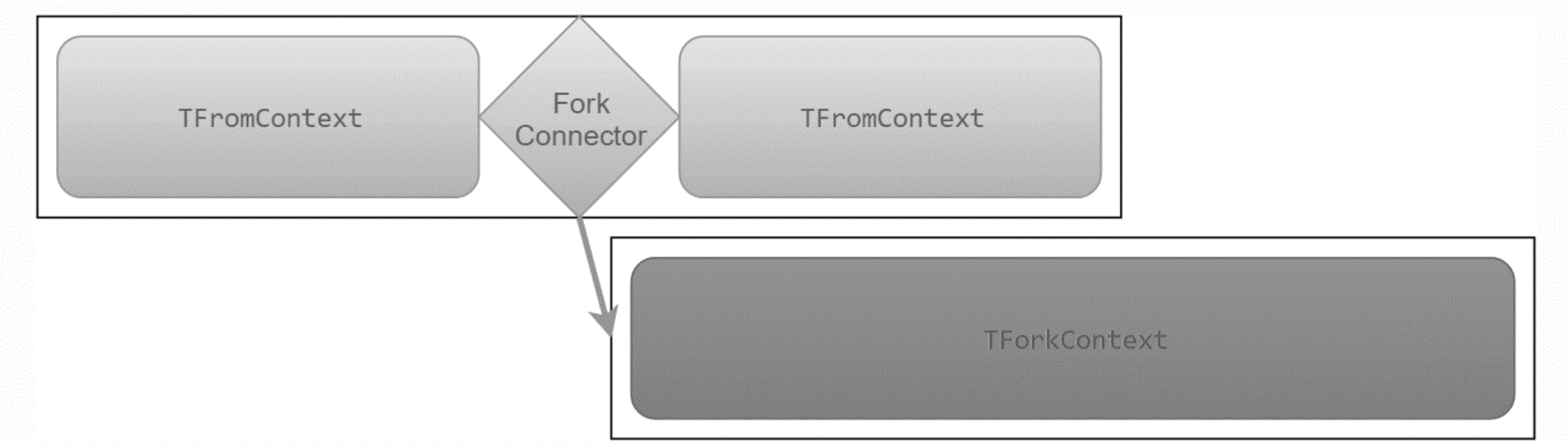
Stages



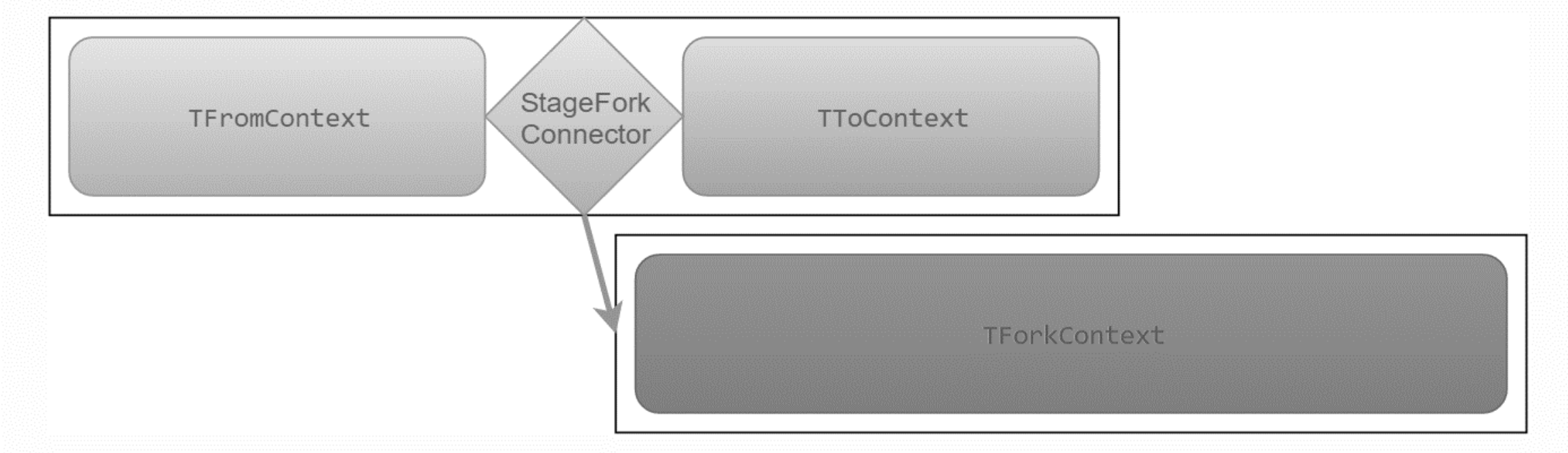
Stage Connector

await Demo

missing ConfigureAwait(false) ;)



Fork Connector

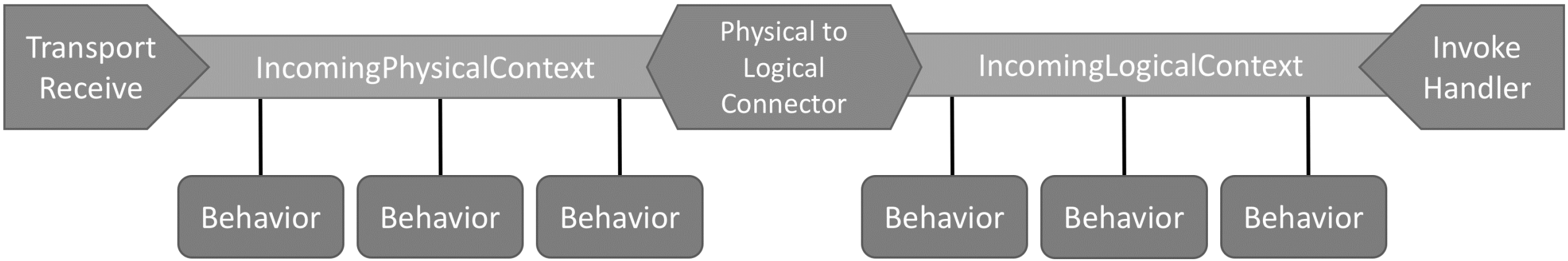


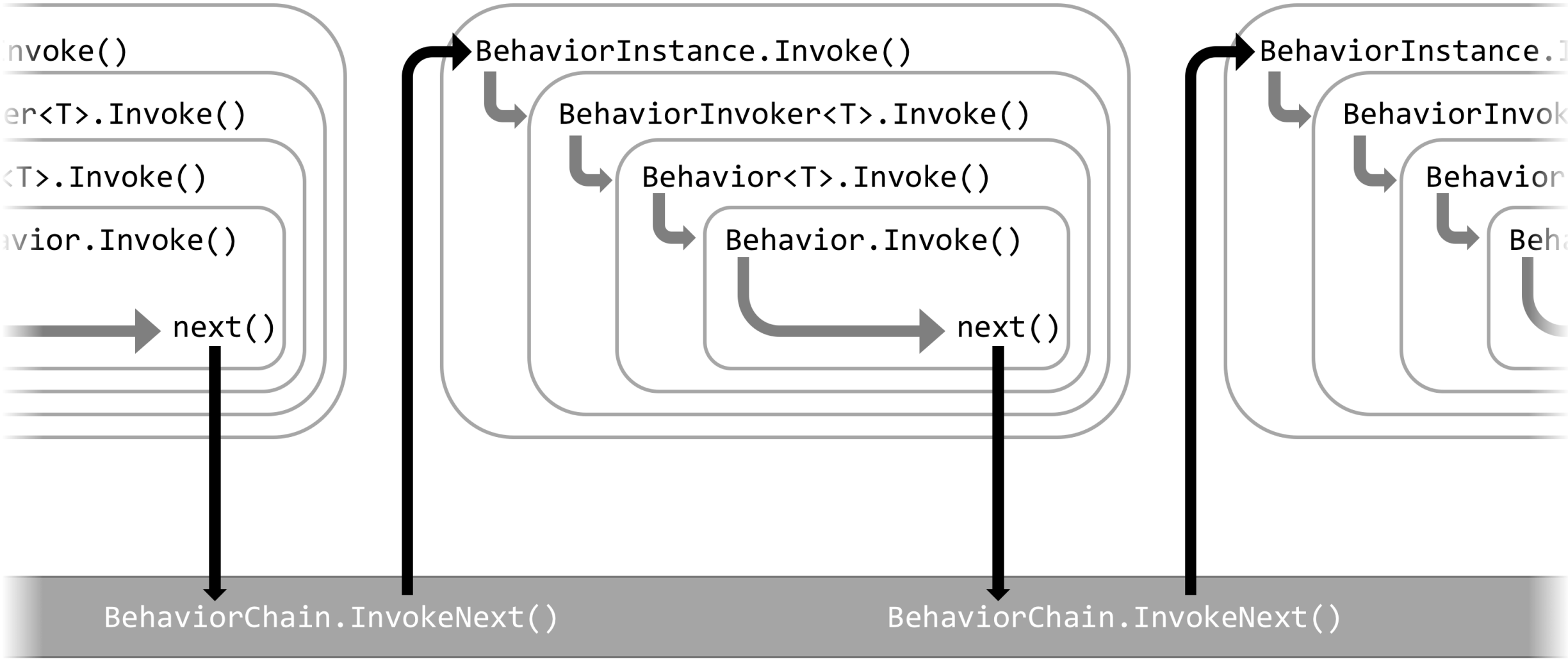
Stage Fork Connector

Tree of Responsibility

Keep calm and let your head explode

The need for speed


Transport Receive
IncomingPhysicalContext
Physical to Logical Connector
IncomingLogicalContext
Invoke Handler
Behavior
Behavior
Behavior
Behavior
Behavior
Behavior


nvoke()
er<T>.Invoke()
<T>.Invoke()
vior.Invoke()
next()
BehaviorChain.InvokeNext()
BehaviorInstance.Invoke()
BehaviorInvoker<T>.Invoke()
Behavior<T>.Invoke()
Behavior.Invoke()
next()
BehaviorChain.InvokeNext()
BehaviorInstance.I
BehaviorInvok
Behavior
Beha

What if we write this code by hand?

```
behaviorOne.Invoke(context0, context1 => {
    physicalToLogicalConnector.Invoke(context1, context2 => {
        behaviorTwo.Invoke(context2, context3 => {
            invokeHandlersBehavior.Invoke(context3, context4 => {
                // End of pipeline
            });
        });
    });
});
```

Expression Trees

TBD


Pattern
Build It
WrapUp

NSB v6

Is Async all the way

Uses the chain of responsibility pattern heavily

particular.net/blog/async-await-its-time

docs.particular.net/nservicebus/pipeline/customizing-v6

Recap

reminder

Chain of Responsibility or Russian Dolls is a flextensible pattern ideally suited to build robust IO bound pipelines

The pattern is used in many OSS projects

Know it, learn it, love it *

Particular
Software

Thanks

Slides, Links...

github.com/danielmarbach/async-dolls

Particular Software

await Q & A

Particular Software

Software Engineer
Enthusiastic Software Engineer
Microsoft MVP

@danielmarbach
particular.net/blog
planetgeek.ch

Particular
Software

Thanks