redhat.

The Reactive Landscape

http://bit.ly/jfokus-reactive

Clement Escoffier, Vert.x Core Developer, Red Hat

Reactive all the things ???

reactive

ADJECTIVE

1 Showing a response to a stimulus.
'pupils are reactive to light'

1.1 *Physiology* Showing an immune response to a specific antigen.
'lysosomal activity can activate B lymphocytes reactive against self-components'

1.2 (of a disease or illness) caused by a reaction to something.
'reactive arthritis'
'reactive depression'

1.3 Having a tendency to react chemically.
'nitrogen dioxide is a highly reactive gas'

2 Acting in response to a situation rather than creating or controlling it.
'a proactive rather than a reactive approach'

3 *Physics*
Relating to reactance.
'a reactive load'

https://en.oxforddictionaries.com/definition/reactive

reactive

FAKE !

ADJECTIVE

1 Showing a response to a stimulus.
'pupils are reactive to light'

1.1 *Physiology* Showing an immune response to a specific antigen.
'lysosomal activity can activate B lymphocytes reactive against self-components'

1.2 (of a disease or illness) caused by a reaction to something.
'reactive arthritis'
'reactive depression'

1.3 Having a tendency to react chemically.
'nitrogen dioxide is a highly reactive gas'

2 Acting in response to a situation rather than creating or controlling it.
'a proactive rather than a reactive approach'

3 *Software*
A software reacting to stimuli such as messages, requests, metrics, availability....

reactive

FAKE !

ADJECTIVE

1 Showing a response to a stimulus.
'pupils are reactive to light'

1.1 *Physiology* Showing an immune response to a specific antigen.
'lysosomal activity can activate B lymphocytes reactive against self-components'

1.2 (of a disease or illness) caused by a reaction to something.
'reactive arthritis'
'reactive depression'

End of the flow of control ?

1.3 Having a tendency to react chemically.
'nitrogen dioxide is a highly reactive gas'

2 Acting in response to a situation rather than creating or controlling it.
'a proactive rather than a reactive approach'

3 *Software*
A software reacting to stimuli such as messages, requests, metrics, availability....

reactive

FAKE !

ADJECTIVE

1 Showing a response to a stimulus.
'pupils are reactive to light'

1.1 *Physiology* Showing an immune response to a specific antigen.
'lysosomal activity can activate B lymphocytes reactive against self-components'

1.2 (of a disease or illness) caused by a reaction to something.
'reactive arthritis'
'reactive depression'

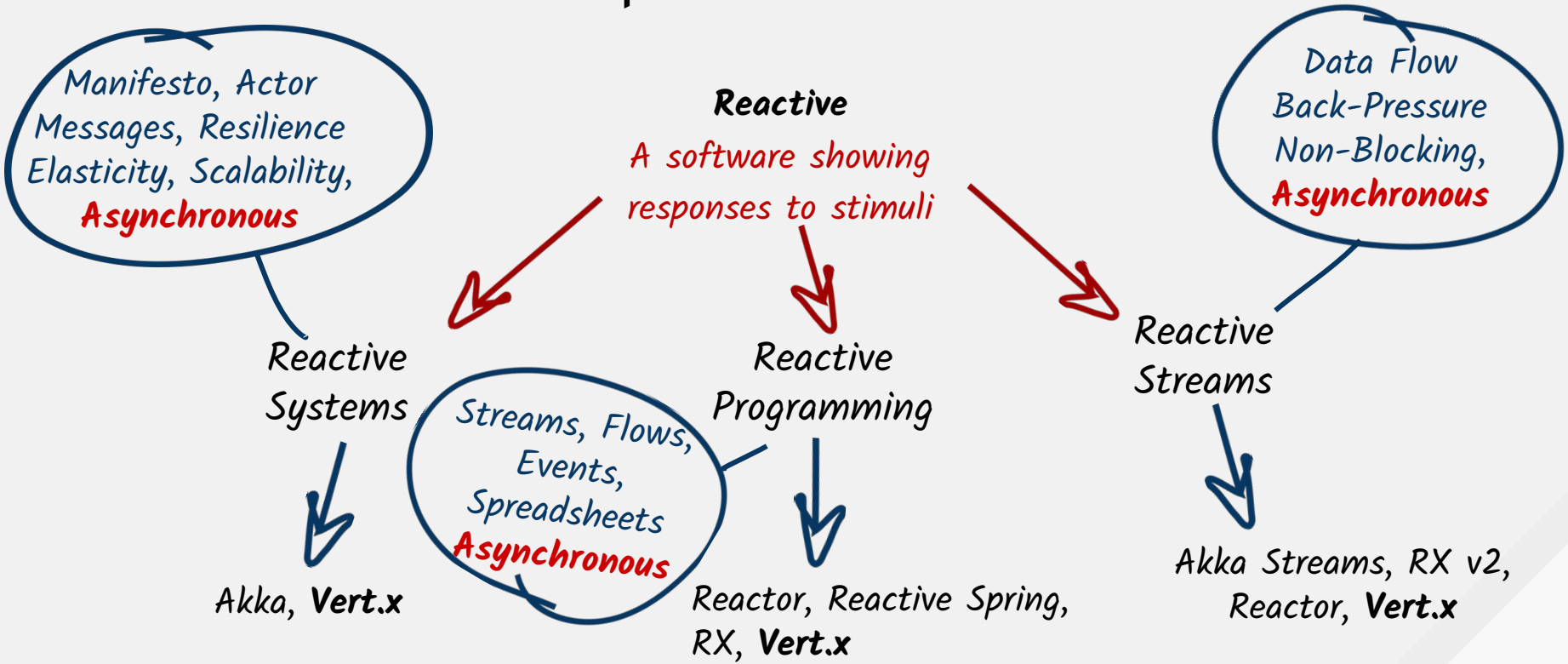
1.3 Having a tendency to react chemically.
'nitrogen dioxide is a highly reactive gas'

End of the flow of control ? or just the comeback of asynchrony ?

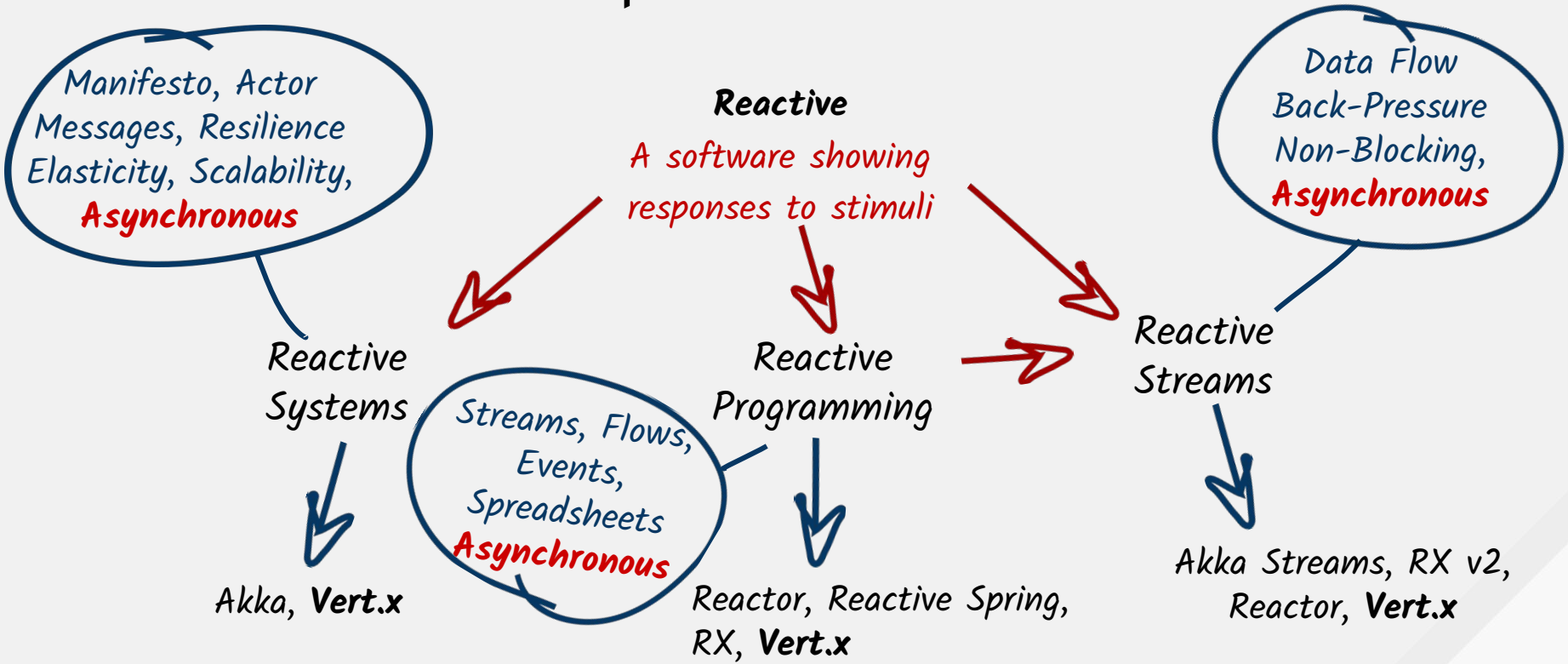
2 Acting in response to a situation rather than creating or controlling it.
'a proactive rather than a reactive approach'

3 *Software*
A software reacting to stimuli such as messages, requests, metrics, availability....

The reactive landscape

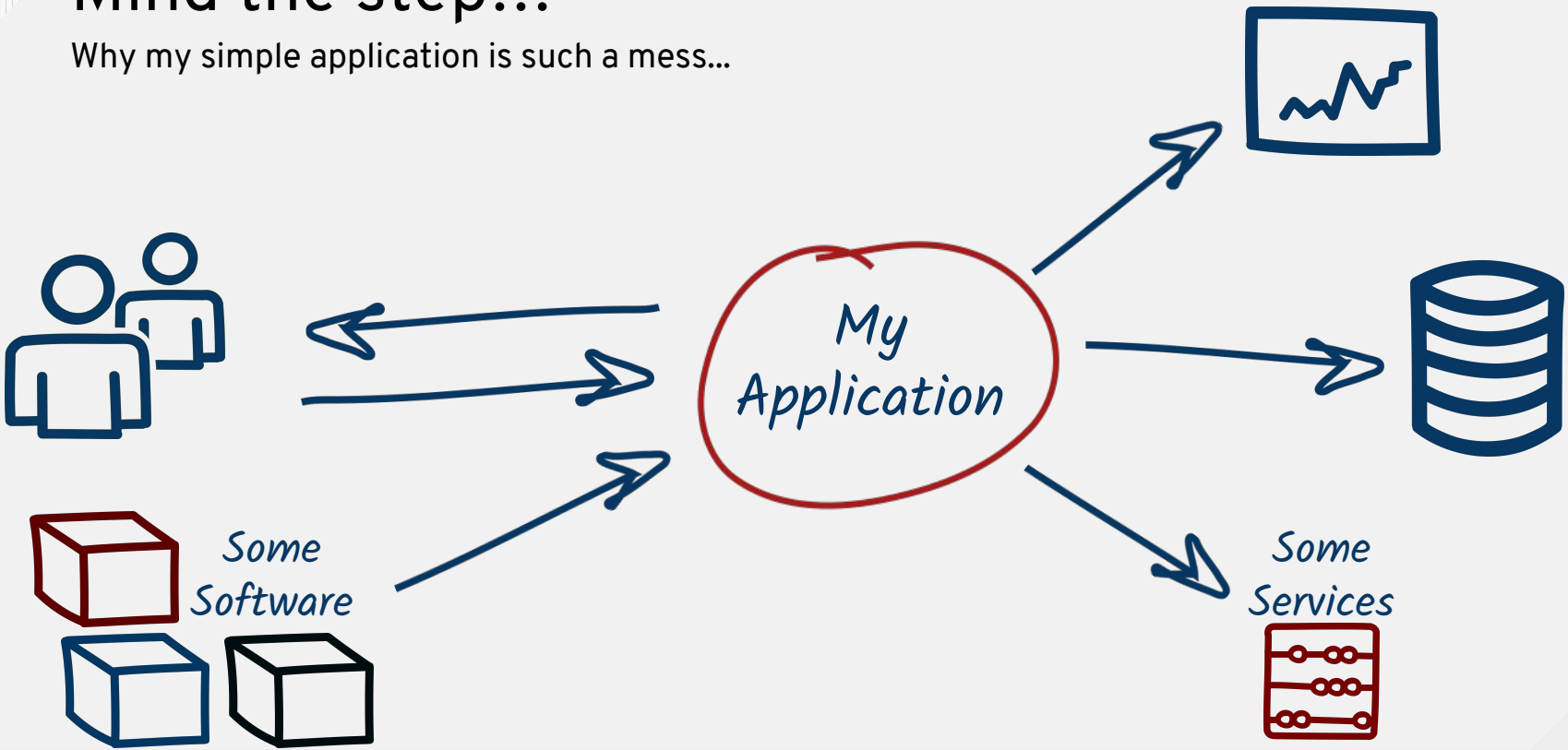


The reactive landscape



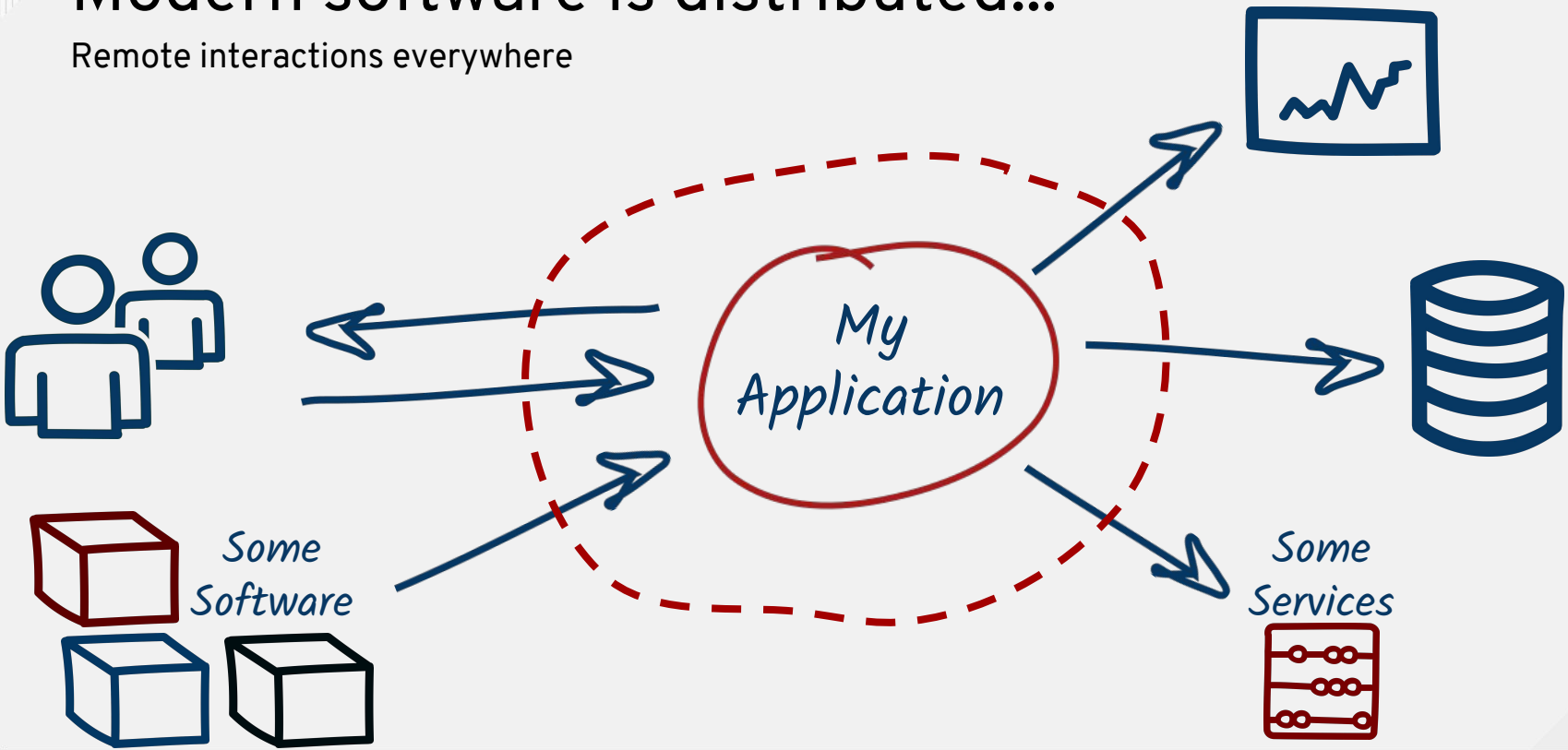
Mind the step...

Why my simple application is such a mess...



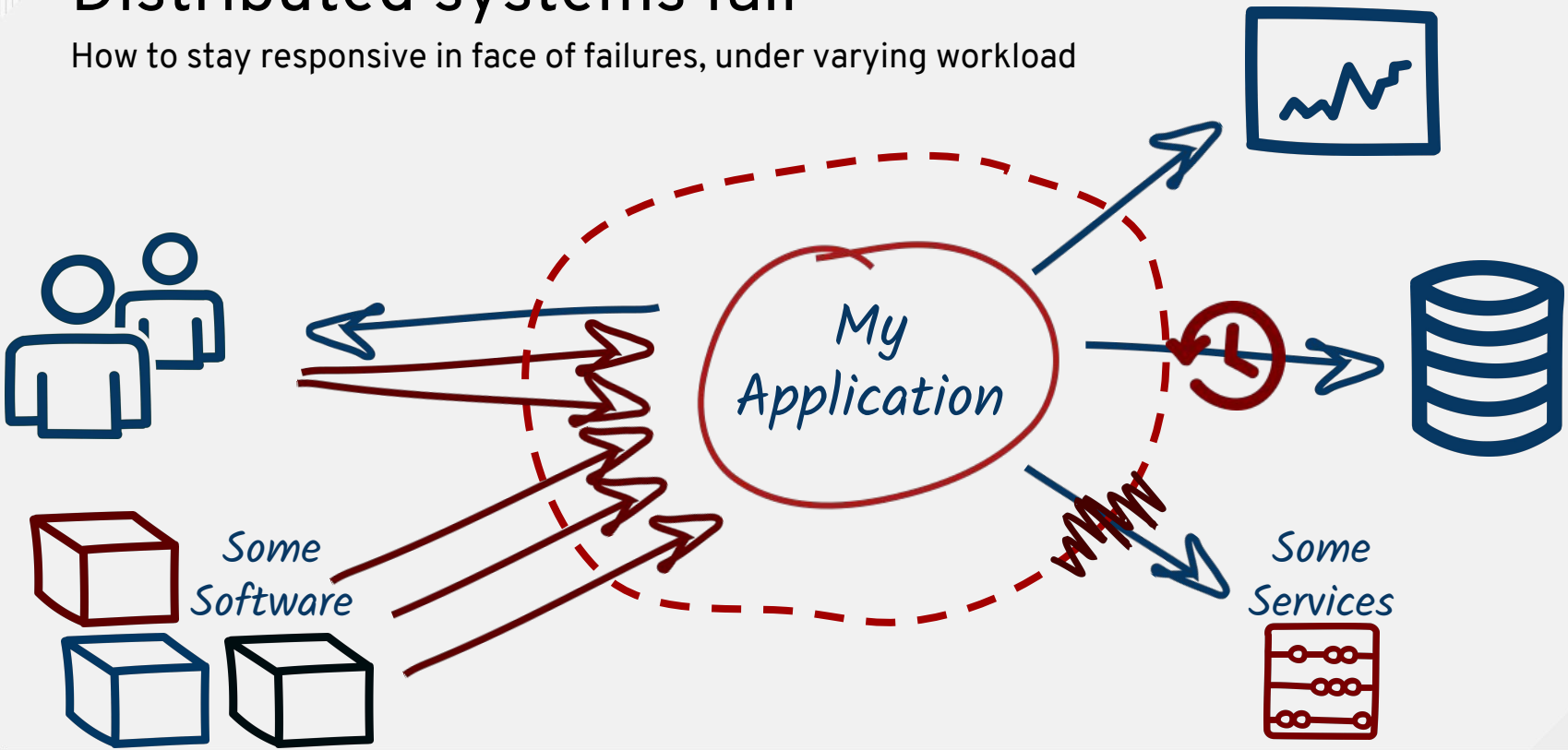
Modern software is distributed...

Remote interactions everywhere



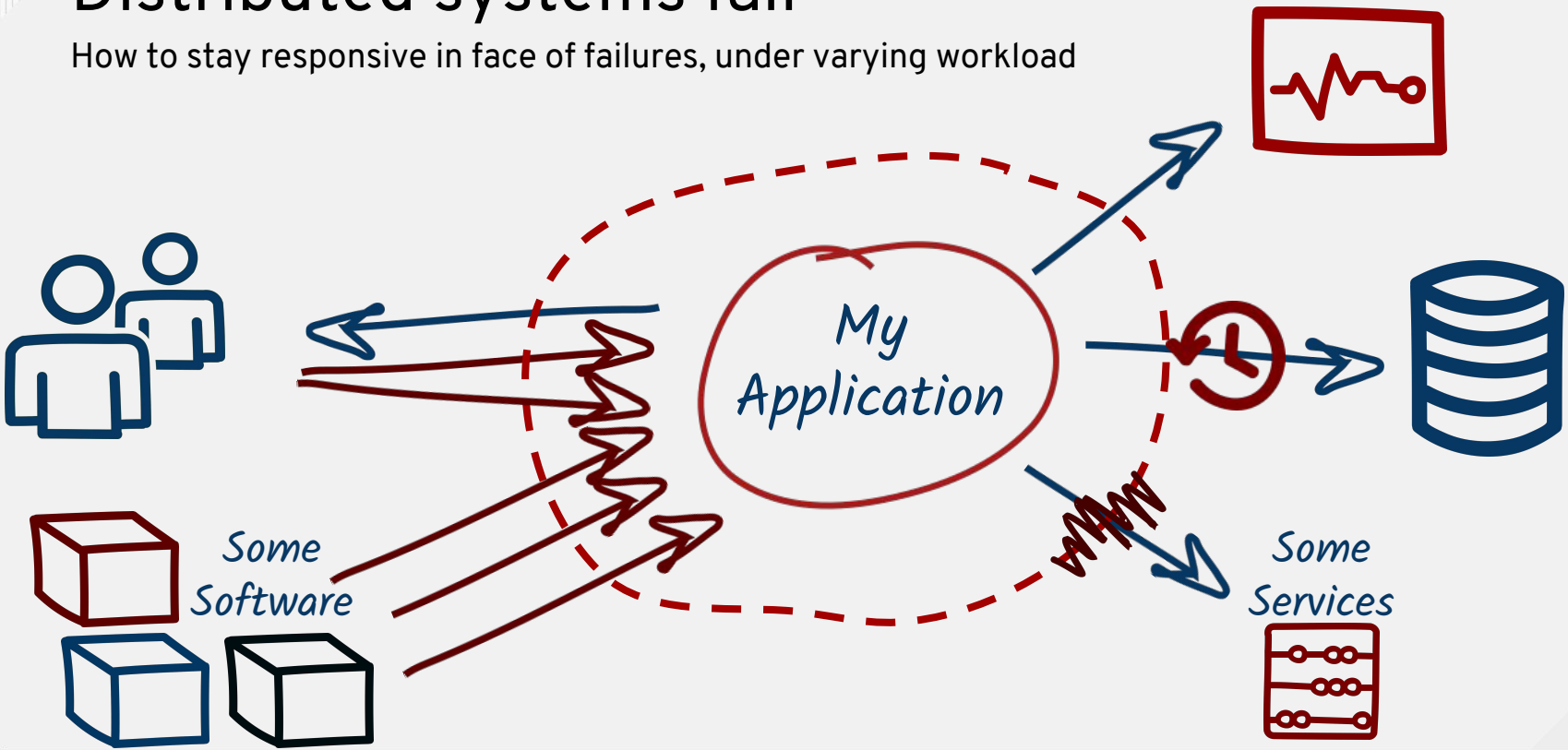
Distributed systems fail

How to stay responsive in face of failures, under varying workload



Distributed systems fail

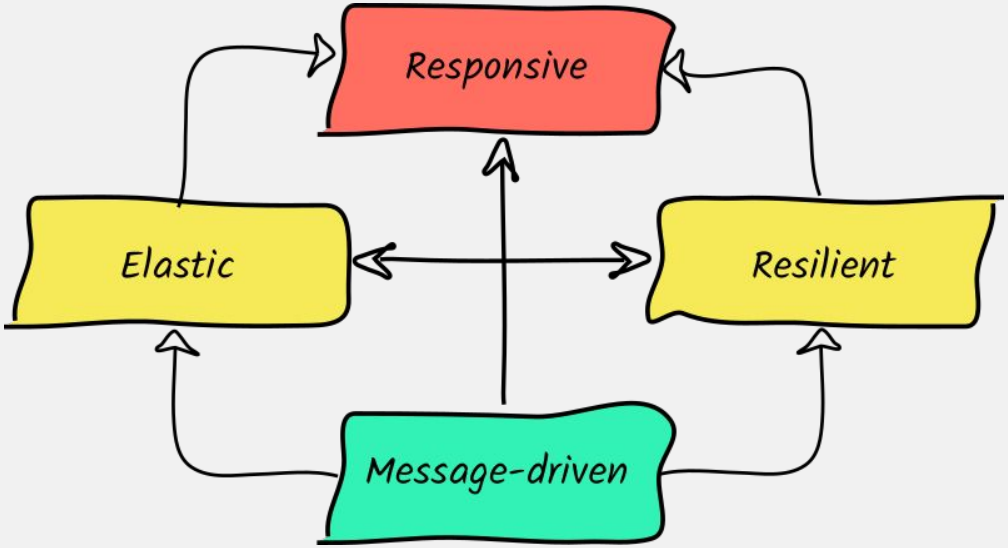
How to stay responsive in face of failures, under varying workload



Reactive Systems => Responsive Systems

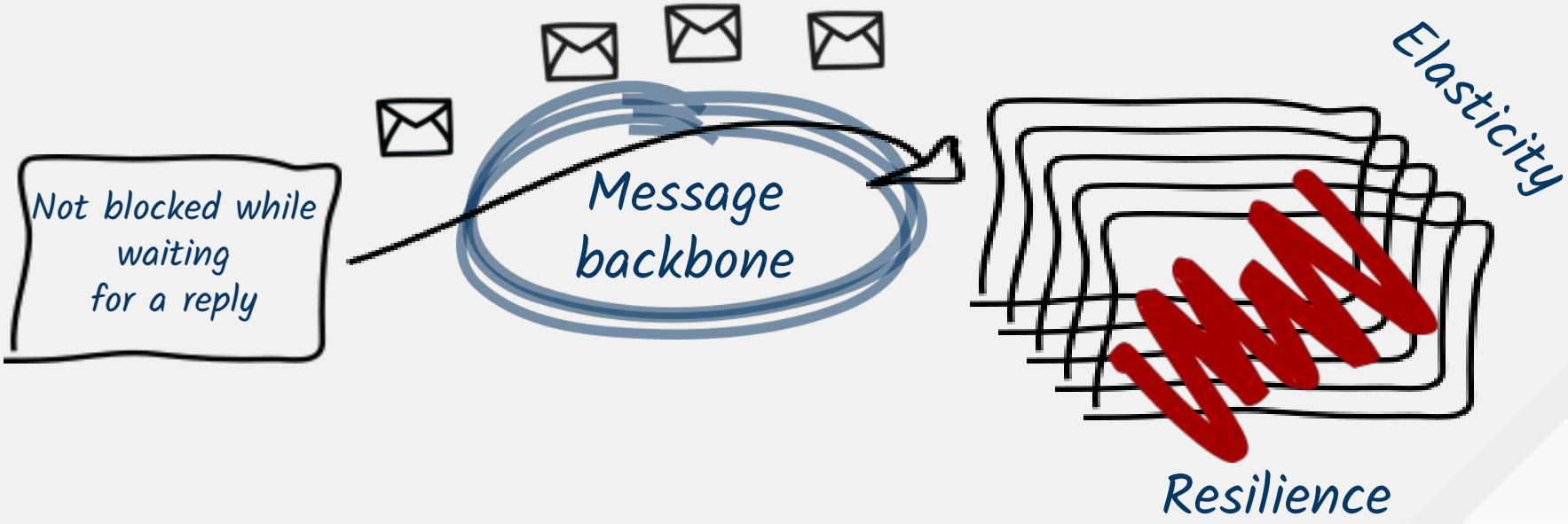
Reactive Manifesto

http://www.reactivemanifesto.org/

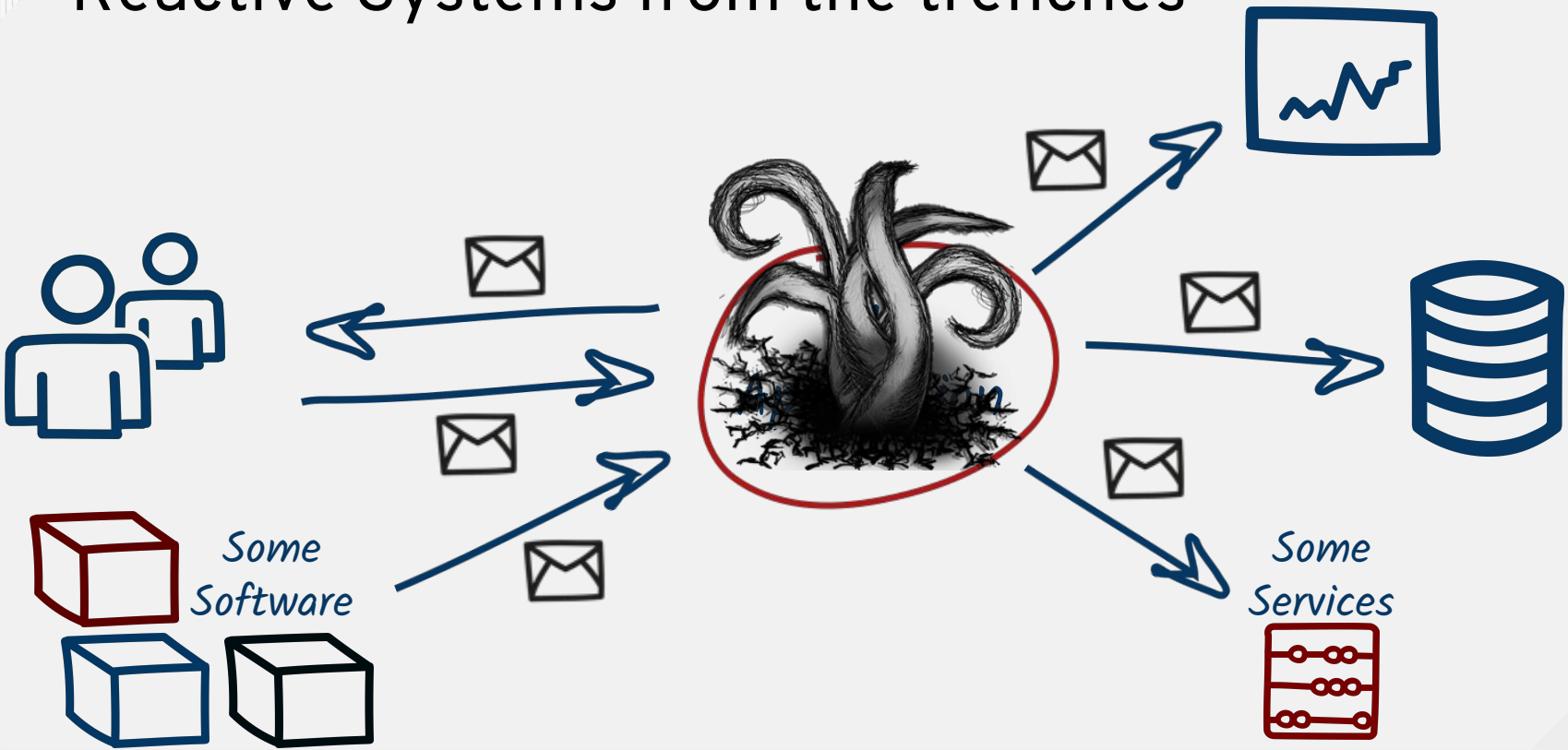


Asynchronous message passing

Asynchronous message passing

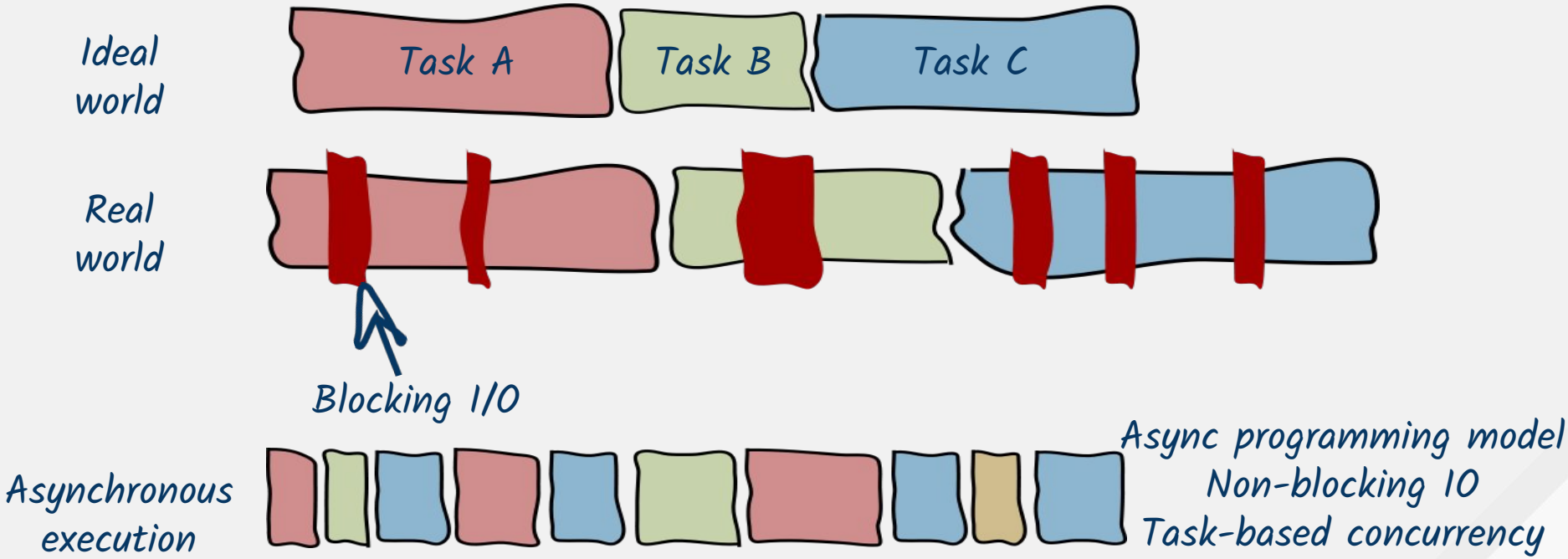


Reactive Systems from the trenches



Asynchronous development model

Asynchronous execution



Asynchronous, welcome to Hollywood

Don't wait, we will call you...

Synchronous

```
public int compute(int a, int b) {
    return ...;
}
```

```
int res = compute(1, 2);
```

Asynchronous

```
public void compute(int a, int b,
  Handler<AsyncResult<Integer>> h){
    // ...
    handler.handle(i);
}
```

```
compute(1, 2, res -> {
    // Called with the
    // async result
});
```

Callbacks for notifications and async actions

Web server example

```
vertx.createHttpServer()
    .requestHandler(req ->  // Async reaction
        req.response().end("Reactive greetings")
    )
    .listen(8080, ar -> { // Async operation
        //...
    });
```

Callbacks lead to...

Reality check

```
client.getConnection(conn -> {
   if (conn.failed()) {/* failure handling */}
   else {
       SQLConnection connection = conn.result();
       connection.query("SELECT * from PRODUCTS",
           rs -> {
               if (rs.failed()) {/* failure handling */}
               else {
                   List<JsonArray> lines =    rs.result().getResults();
                   for (JsonArray l : lines) {  System.out.println(new Product(l)); }
                   connection.close(
                      done -> {
                       if (done.failed()) {/* failure handling */}
                   });
               }
       });
   }
});
```

Reactive Programming...

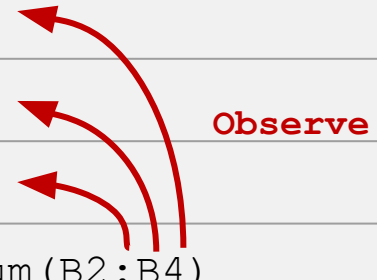
Tame the asynchronous

Are you an “Excel” user?

My Expense Report	
Lunch	15€
Coffee	25€
Drinks	45€
Total	85€

Are you an “Excel” user?

My Expense Report	
Lunch	15€
Coffee	25€
Drinks	45€
Total	=sum(B2:B4)



Streams (a.k.a Observables, Flowables, Publishers)

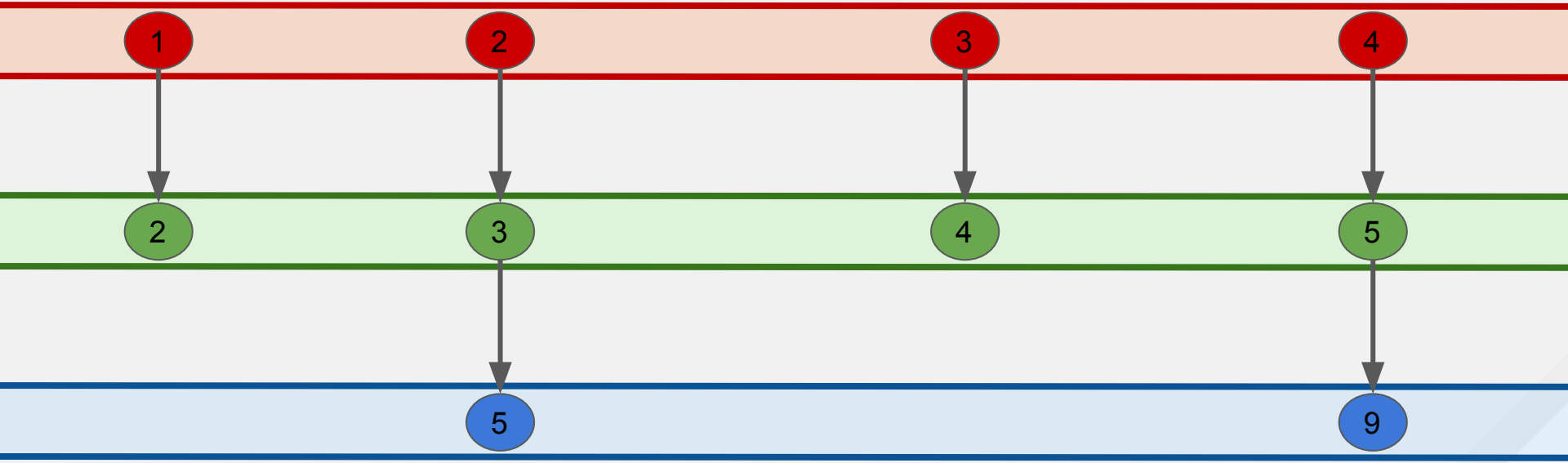
My Expense Report	
Lunch	15€
Coffee	0€
Drinks	0€
Total	15€

My Expense Report	
Lunch	15€
Coffee	25€
Drinks	0€
Total	40€

My Expense Report	
Lunch	15€
Coffee	25€
Drinks	45€
Total	85€

Reactive Programming

Publisher and Subscriber



Reactive Extension - RX Java 2

```
Flowable<Integer> obs1 = Flowable.range(1, 10);

Flowable<Integer> obs2 = obs1.map(i -> i + 1);

Flowable<Integer> obs3 = obs2.window(2)
   .flatMap(MathFlowable::sumInt);

obs3.subscribe(
   i -> System.out.println("Computed " + i)
);
```

Reactive types & Asynchronous

Flowable / Observable - Stream of data, Async reaction

- Bounded or unbounded stream of values
- Data, Error, End of Stream

Singles / Maybe - Async operation

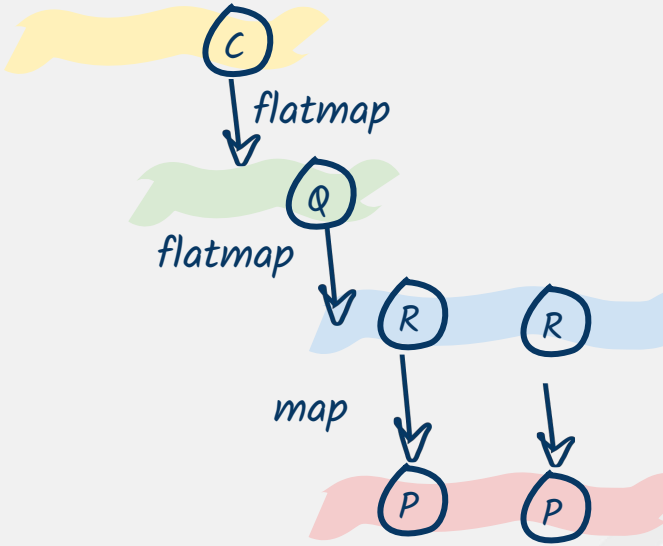
- Stream of one value
- Data, Error, Completion (maybe)

Completables - Async operation (no return)

- Stream without a value
- Completion, Error

Reactive types & Asynchronous

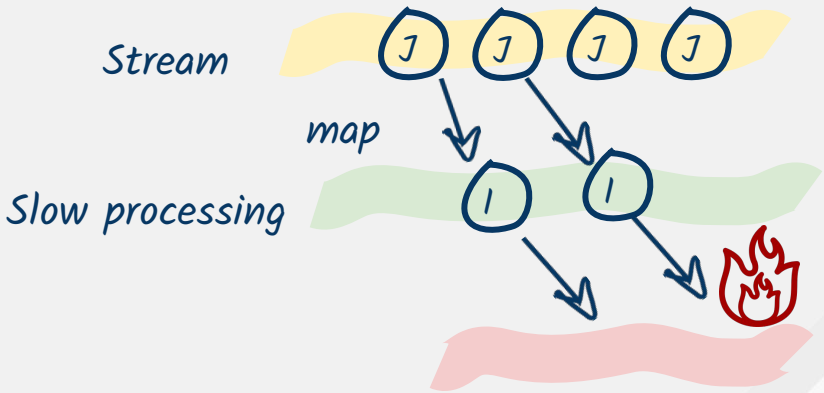
```
client.rxGetConnection()   // Single(async op)
 .flatMapPublisher(conn ->
   conn
     .rxQueryStream("SELECT * from PRODUCTS")
     .flatMapPublisher(SQLRowStream::toFlowable)
     .doAfterTerminate(conn::close)
 ) // Flowable of Rows
 .map(Product::new) // Flowable of Products
 .subscribe(System.out::println);
```



Back pressure & Reactive streams

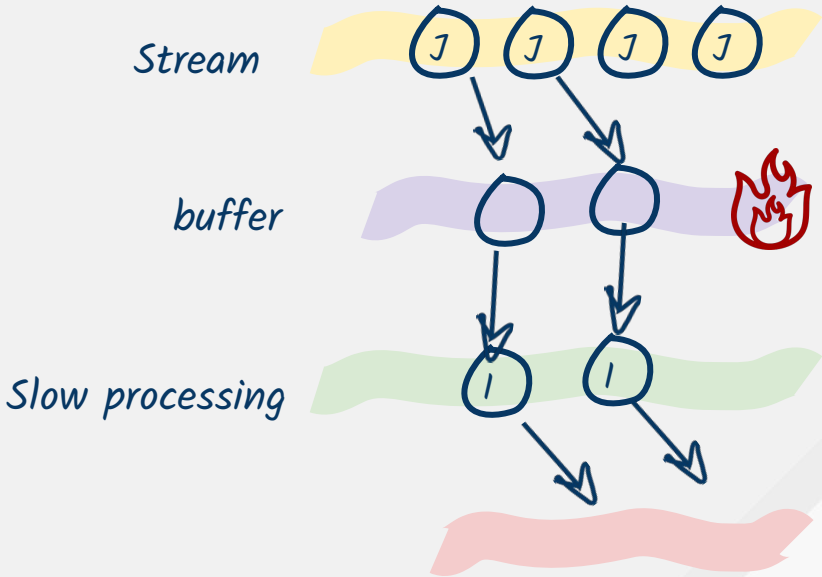
What if … the processing can't keep up

```
Streams.getStream()
   .observeOn(Schedulers.computation())

   .map(i -> {
      Thread.sleep(100); // I'm slow...
      return i.getLong("id");
   })
   .blockingSubscribe(
      x -> System.out.println("Processed:" + x),
      Throwable::printStackTrace,
      () -> System.out.println("Completed !")
   );
```



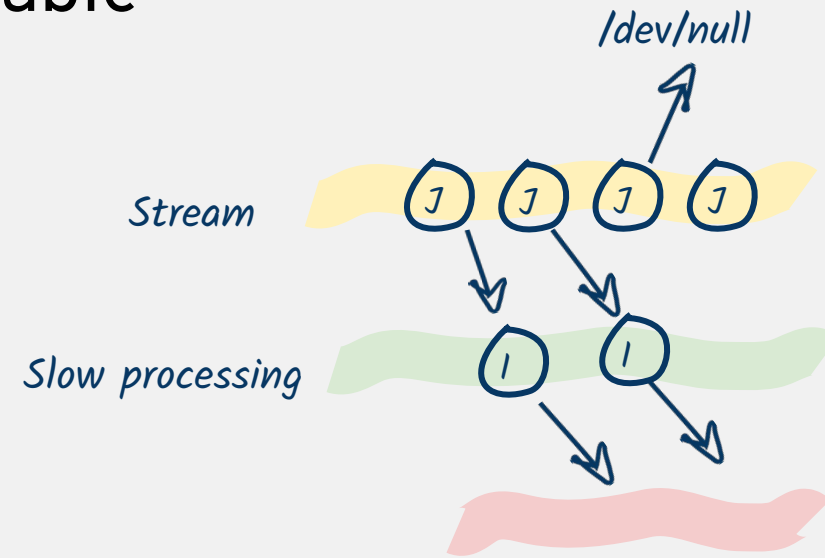
Buffer to handle small bumps

```
Streams.getStream()
  .onBackPressureBuffer(10000)
  .observeOn(Schedulers.computation())

  .map(i -> {
     Thread.sleep(100); // I'm slow...
     return i.getLong("id");
  })
  .blockingSubscribe(
     x -> System.out.println("Processed:" + x),
     Throwable::printStackTrace,
     () -> System.out.println("Completed !")
  );
```



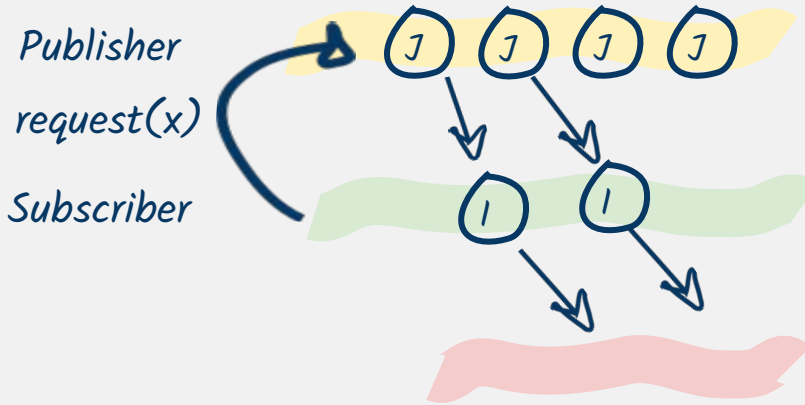
Drop when data loss is acceptable

```
Streams.getStream()
   .onBackPressureDrop()
   .observeOn(Schedulers.computation())

   .map(i -> {
      Thread.sleep(100); // I'm slow...
      return i.getLong("id");
   })
   .blockingSubscribe(
      x -> System.out.println("Processed:" + x),
      Throwable::printStackTrace,
      () -> System.out.println("Completed !")
   );
```



Control Flow and Reactive Streams

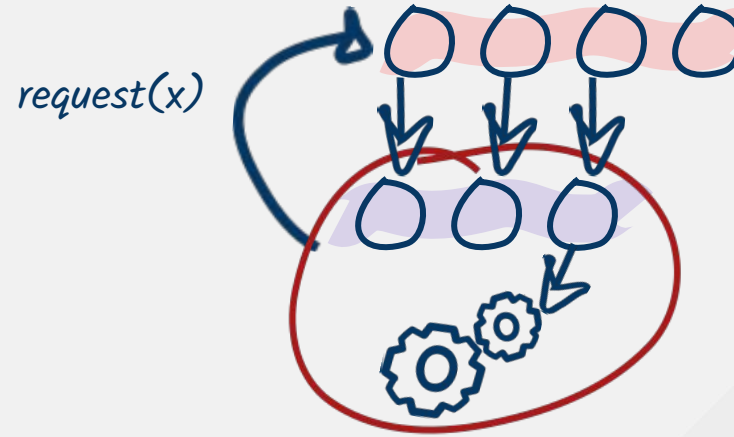
```
Streams.getStreamWithBackPressure()
   .observeOn(Schedulers.computation())

   .map(i -> {
      Thread.sleep(100); // I'm slow...
      return i.getLong("id");
   })
   .blockingSubscribe(
      x -> System.out.println("Processed:" + x),
      Throwable::printStackTrace,
      () -> System.out.println("Completed !")
   );
```



Reactive Stream API - http://www.reactive-streams.org
Java 9 Flow - http://download.java.net/java/jdk9/docs/api/java/util/concurrent/Flow.html

Back to our database access

```
client.rxGetConnection()   // Single(async op)
 .flatMapPublisher(conn ->
   conn
     .rxQueryStream("SELECT * from PRODUCTS")
     .flatMapPublisher(SQLRowStream:: toFlowable)
     .doAfterTerminate(conn::close)
 ) // Flowable of Rows
 .map(Product::new) // Flowable of Products
.subscribe(p -> {
    // do something slow, but it's fine
  });
```

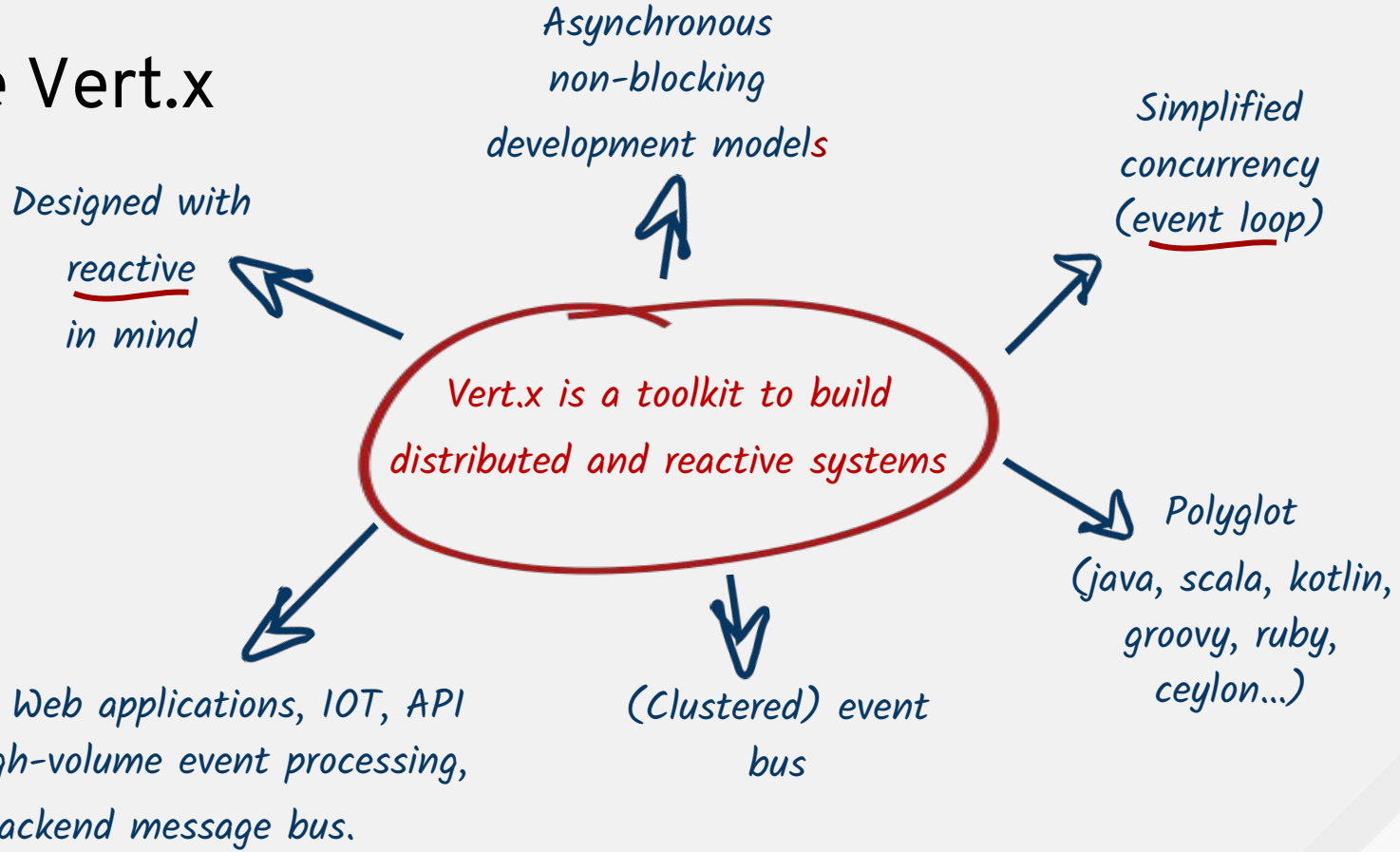


Eclipse Vert.x

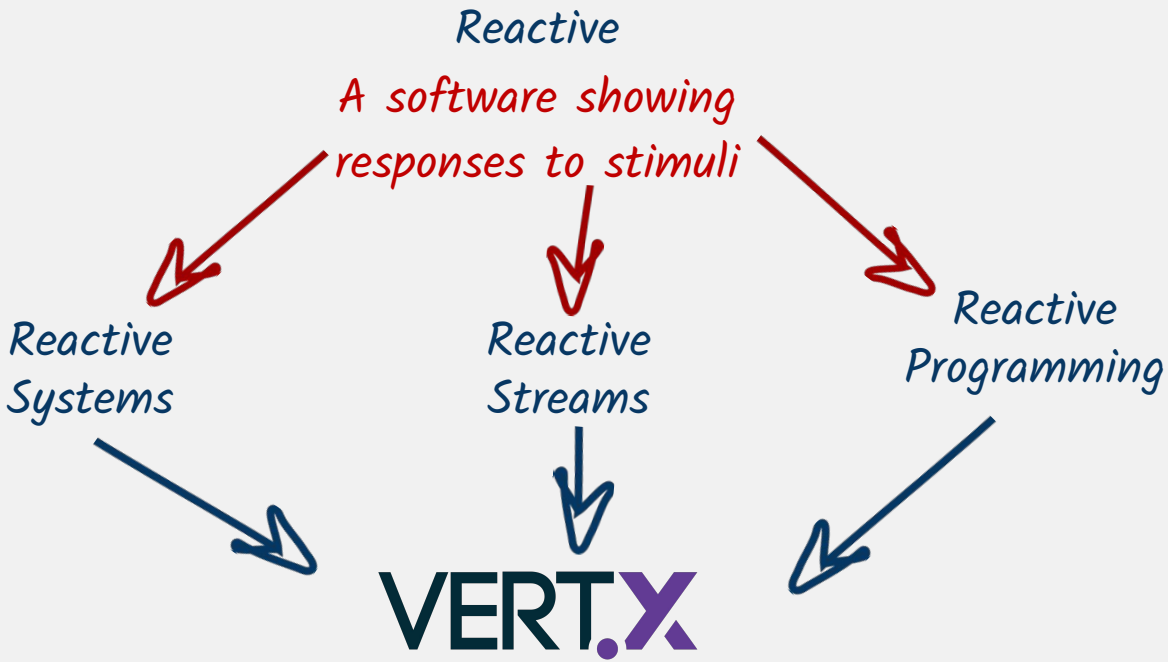
Unleash your reactive superpowers

redhat.

Eclipse Vert.x

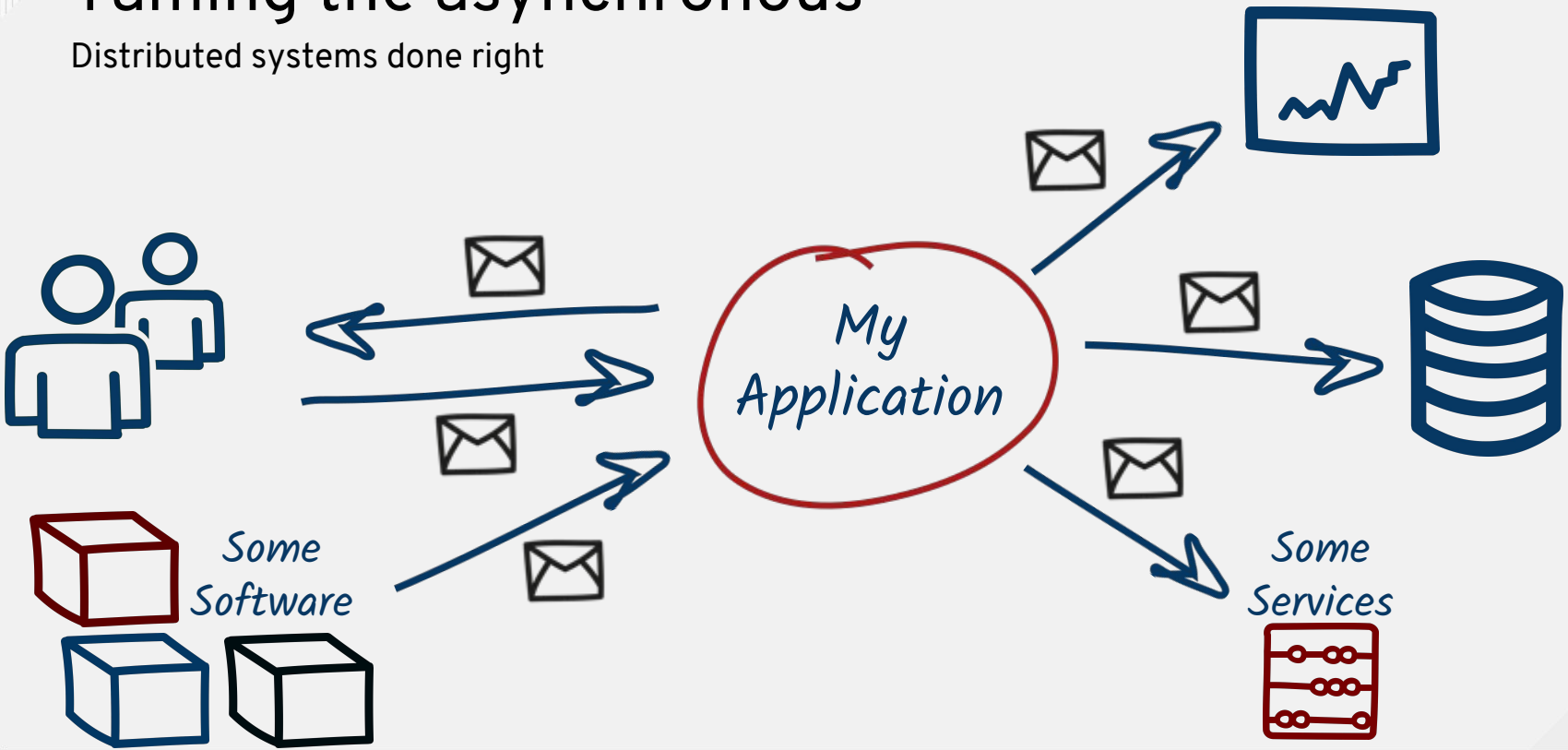


Vert.x - the all-in-one toolkit



Taming the asynchronous

Distributed systems done right



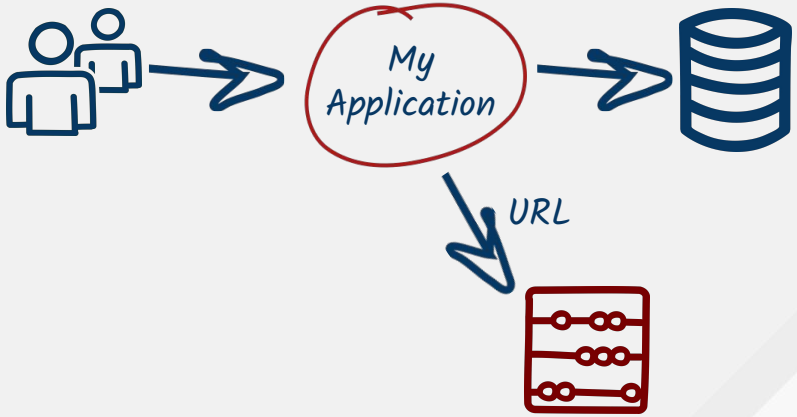
Reactive Web Application

```
private void add(RoutingContext rc) {
  String name = rc.getBodyAsString();
  database.insert(name) // Single (async)
    .subscribe(
      () -> rc.response().setStatusCode(201).end(),
      rc::fail
    );
}

private void list(RoutingContext rc) {
  HttpServerResponse response = rc.response().setChunked(true);
  database.retrieve() // Flowable<Product> (async)
    .subscribe(
      p -> response.write(Json.encode(p) +" \n\n"),
      rc::fail,
      response::end);
}
```

Orchestrating remote interactions

Sequential composition

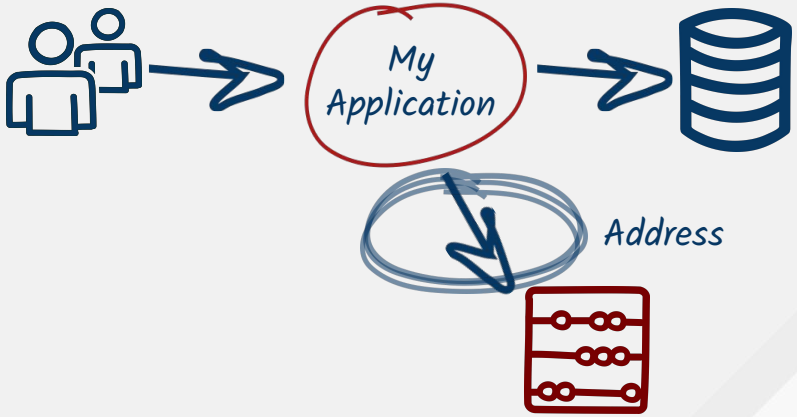
```
WebClient pricer = ...
HttpServerResponse response = rc.response().setChunked(true);
database.retrieve()
  // For each row call...
  .flatMapSingle(p ->
     webClient
        .get("/prices/" + p.getName())
        .rxSend()
        .map(HttpResponse::bodyAsJsonObject)
        .map(json -> p.setPrice(json.getDouble("price")))
  )
  .subscribe(
     p -> response.write(Json.encode(p) + " \n\n"),
     rc::fail,
     response::end);
```



Orchestrating remote interactions

Event Bus

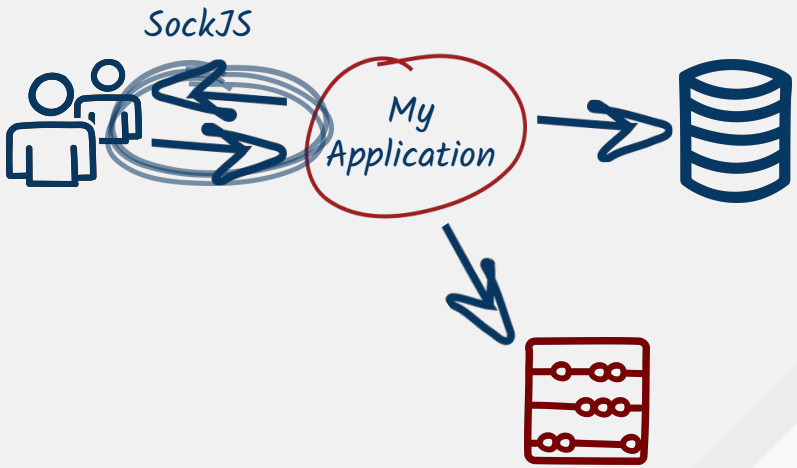
```
WebClient pricer = ...
HttpServerResponse response = rc.response().setChunked(true);
database.retrieve()
  // For each row call...
  .flatMapSingle(p ->
    vertx.eventBus()
        .<JsonObject>rxSend("pricer", p.getName())
        .map(Message::body)
        .map(json -> p.setPrice(json.getDouble("price")))
  )
  .subscribe(
      p -> response.write(Json.encode(p) + " \n\n"),
      rc::fail,
      response::end);
```



Push data using **event bus** bridges

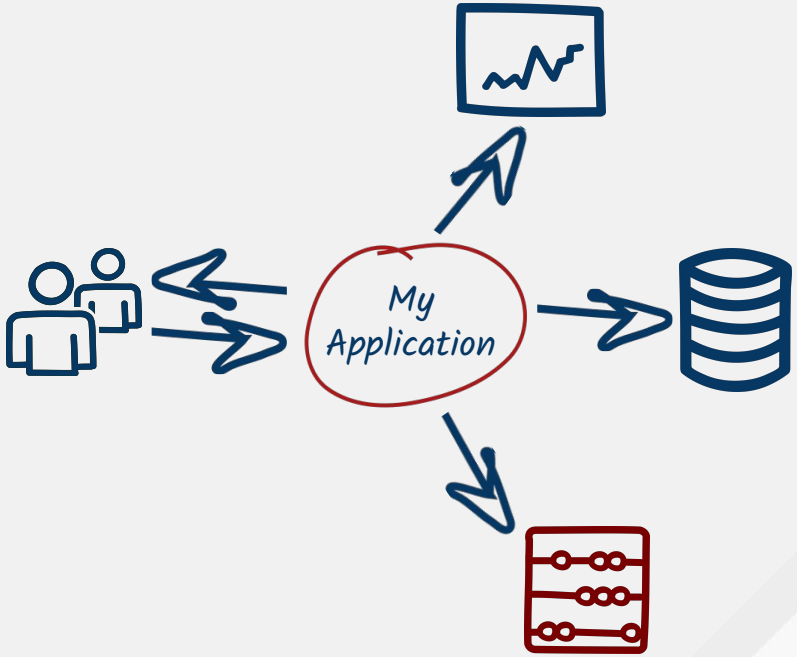
Web Socket, SSE...

```
String name = rc.getBodyAsString().trim();
database.insert(name)
   .flatMap(...)
   .subscribe(
      p -> {
         String json = Json.encode(p);
         rc.response().setStatusCode(201).end(json);
         vertx.eventBus().publish("products", json);
      },
      rc::fail);
```



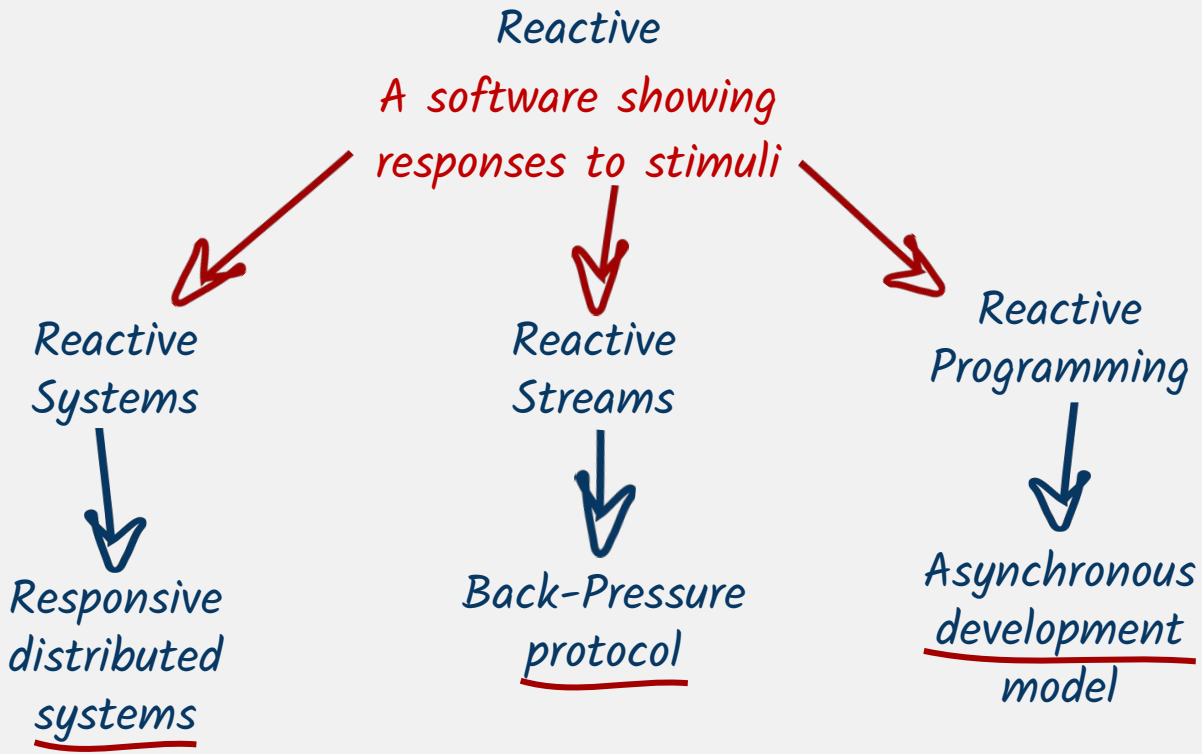
Executing several operations concurrently

```
database.insert(name)
  .flatMap(p -> {
    Single<Product> price = getPriceForProduct(p);
    Single<Integer> audit = sendActionToAudit(p);
    return Single.zip(price, audit, (pr, a) -> pr);
  })
  .subscribe(
    p -> {
      String json = Json.encode(p);
      rc.response().setStatusCode(201).end(json);
      vertx.eventBus().publish("products", json);
    },
    rc::fail);
```



Reactive => Build better systems

Welcome to an asynchronous world



Building Reactive Microservices in Java

Use Eclipse Vert.x to build reactive microservices:

- Explore the elements of reactive microservices and learn how Vert.x works.
- Build and consume a single microservice to understand how messaging improves its reactiveness.
- Create an entire microservices system, using stability and resilience patterns to manage failures.

Download the book from: http://bit.ly/vertx-reactive

clement@apache.org