RED HAT SUMMIT

Reactive Programming with Vert.x

Embrace asynchronous to build responsive systems

Clement Escoffier
Principal Software Engineer

redhat.

Reactive

The new gold rush ?

Reactive system, reactive manifesto, reactive extension, reactive programming, reactive Spring…

Scalability, Asynchronous, Back-Pressure, Spreadsheet, Non-Blocking, Actor, Agent…

Reactive ?

Oxford dictionary

1 - Showing a response to a stimulus

1.1 (*Physiology)* Showing an immune response to a specific antigen

1.2 (of a disease or illness) caused by a reaction to something: *‘reactive depression’*

2 - Acting in response to a situation rather than creating or controlling it

Reactive ?

Application to software

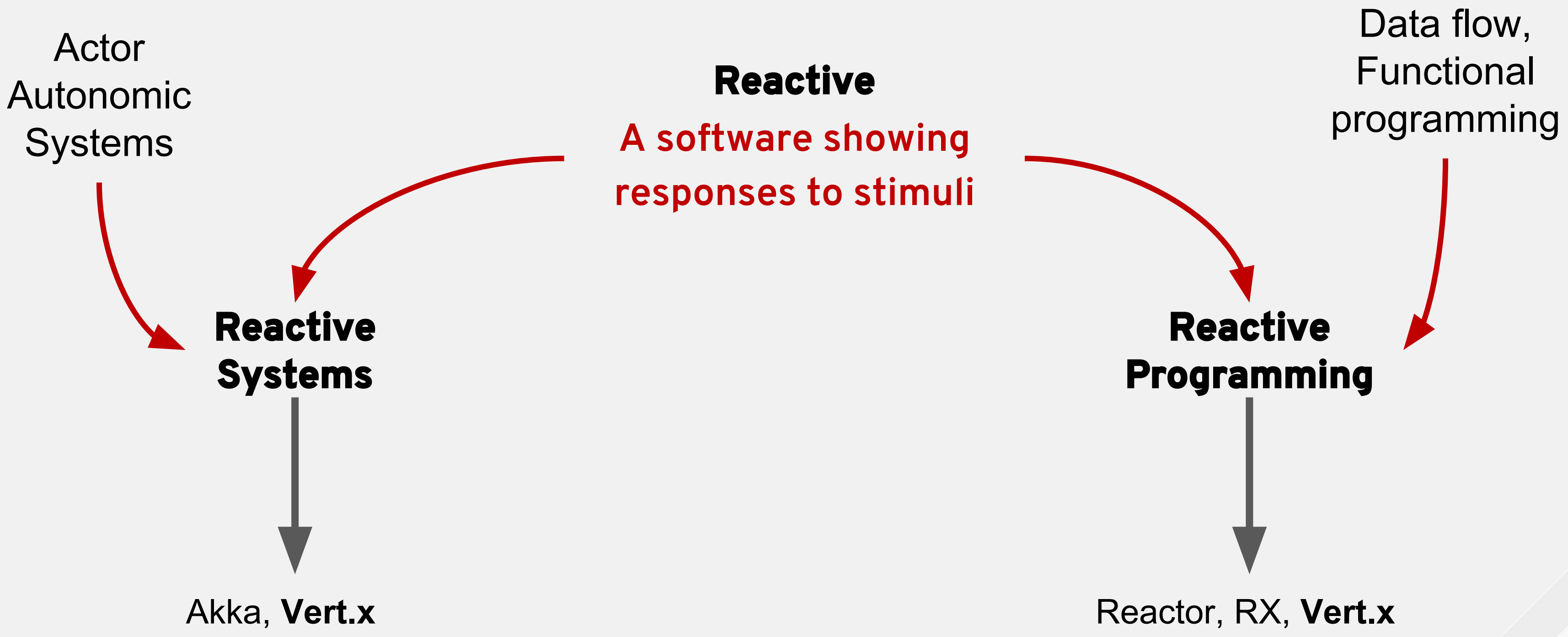
A software showing responses to stimuli

- Events, Messages, Requests, Failures, Measures, Availability…
- The end of the flow of control ?

Is it new?

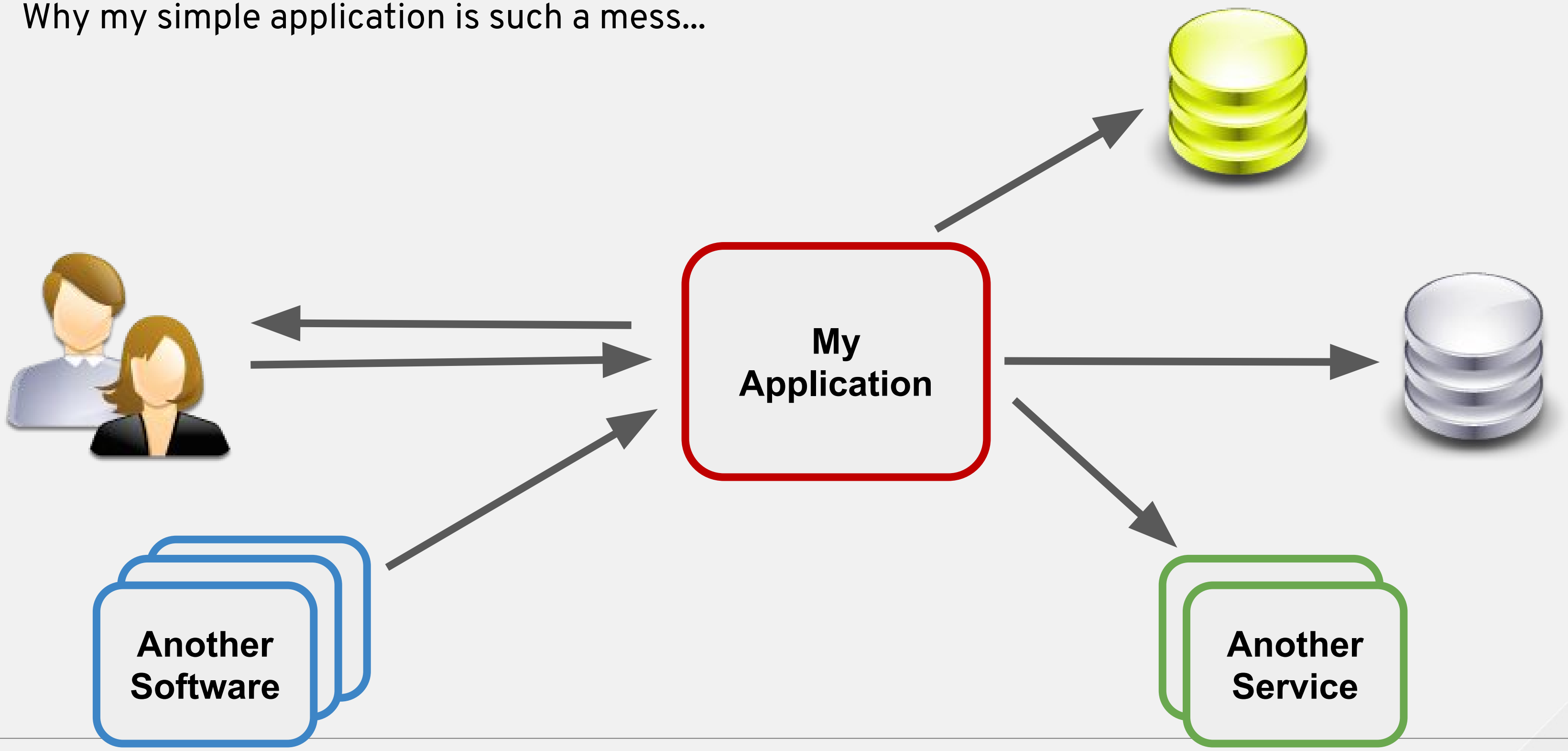
- Actors, Object-oriented programing…
- IOT, Streaming platform, complex event processing, event sourcing…

The two faces of the reactive coin



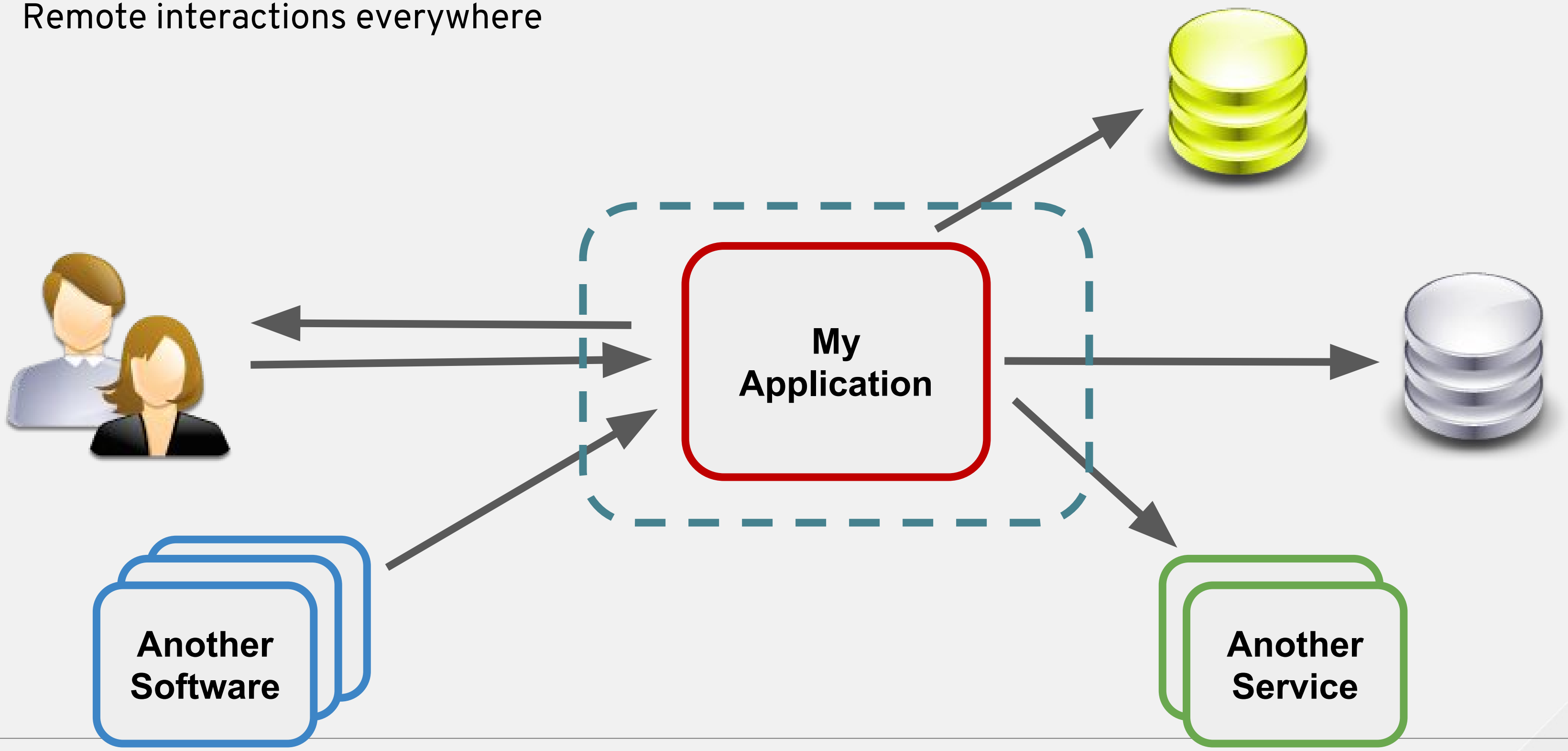
Modern software is not autonomous

Why my simple application is such a mess...



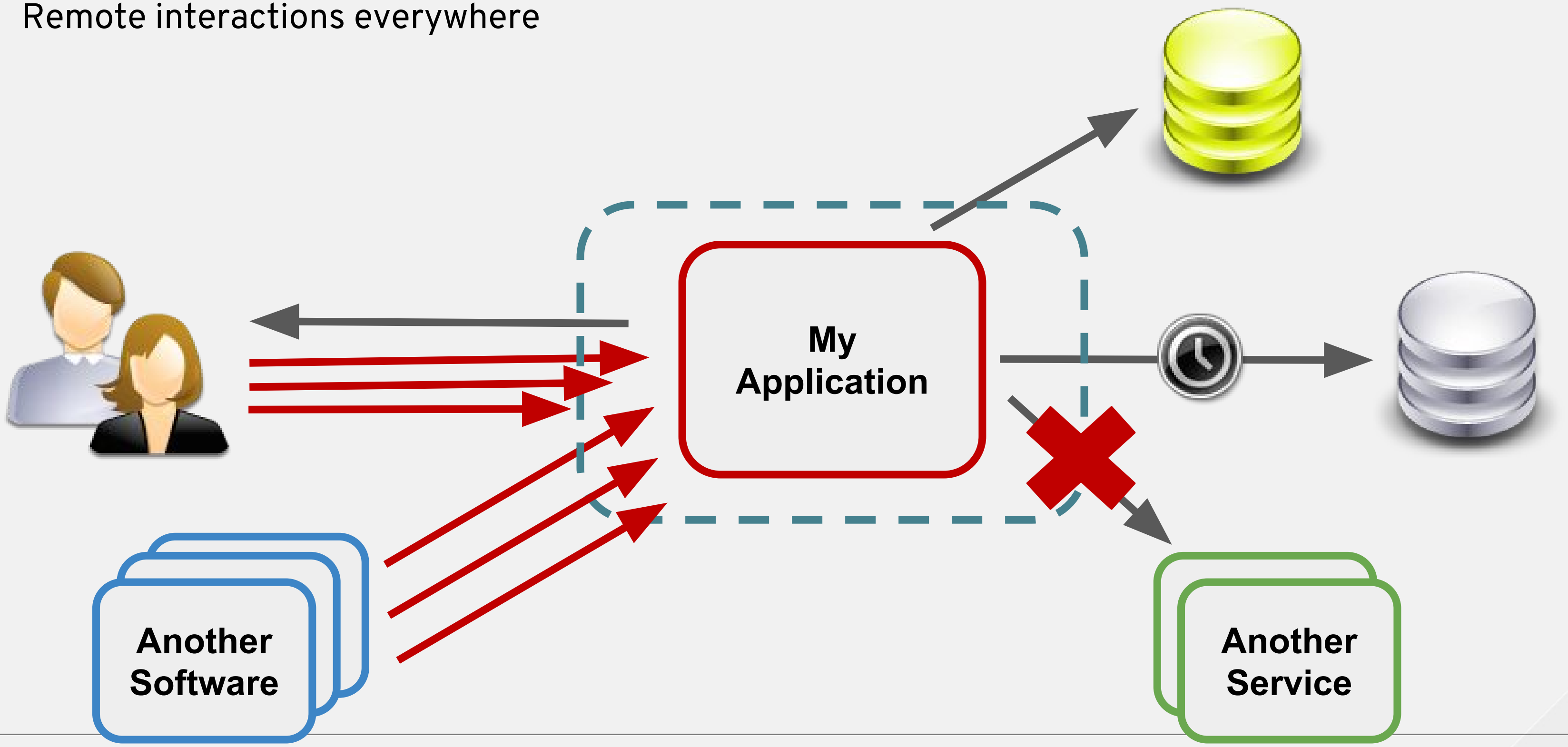
Modern software is not autonomous

Remote interactions everywhere



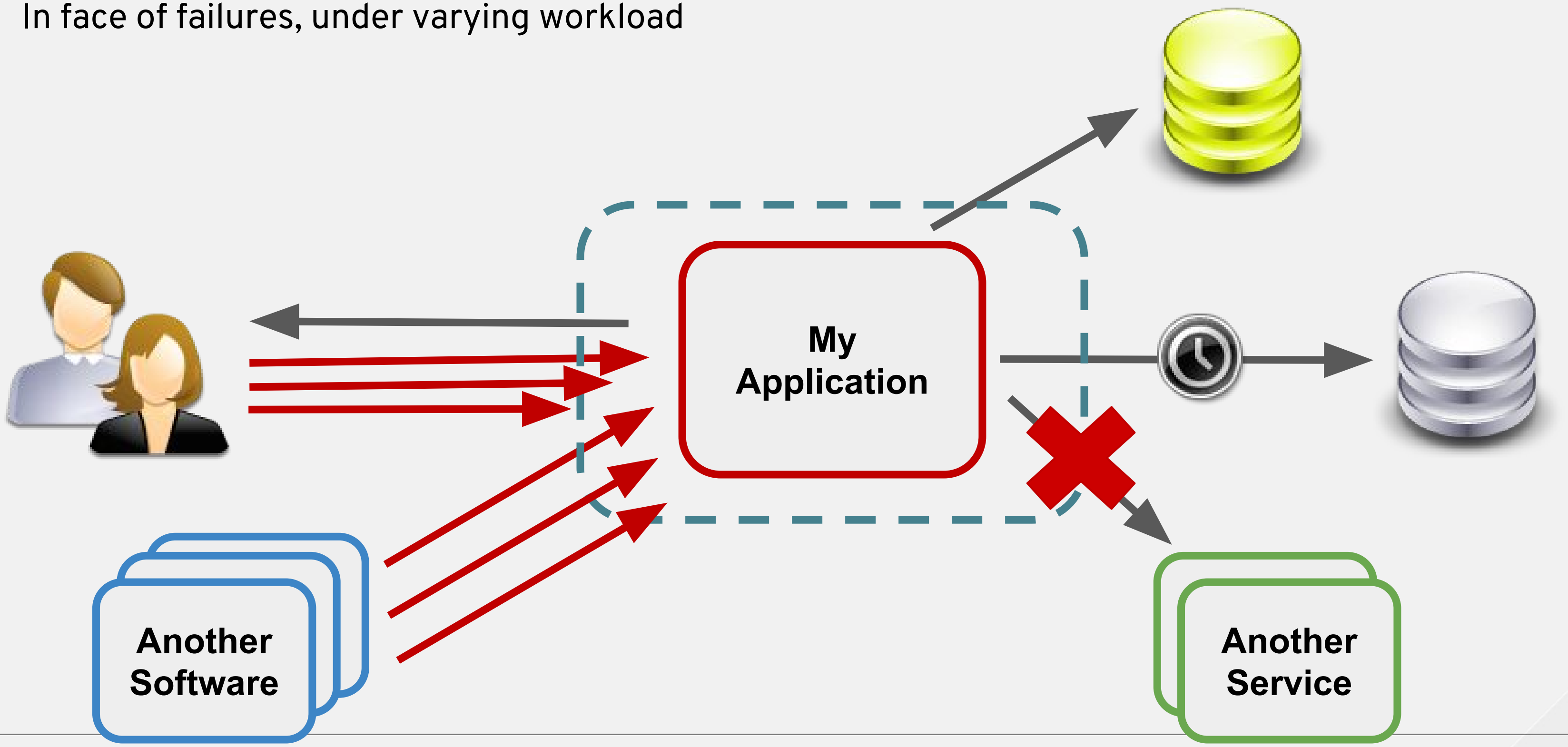
Modern software is not autonomous

Remote interactions everywhere



Need for responsiveness

In face of failures, under varying workload



Reactive Systems => Responsive Systems

Reactive Manifesto

http://www.reactivemanifesto.org/

Reactive Systems are an architecture style focusing on **responsiveness**

- Asynchronous message passing
- Resilient
- Scalable

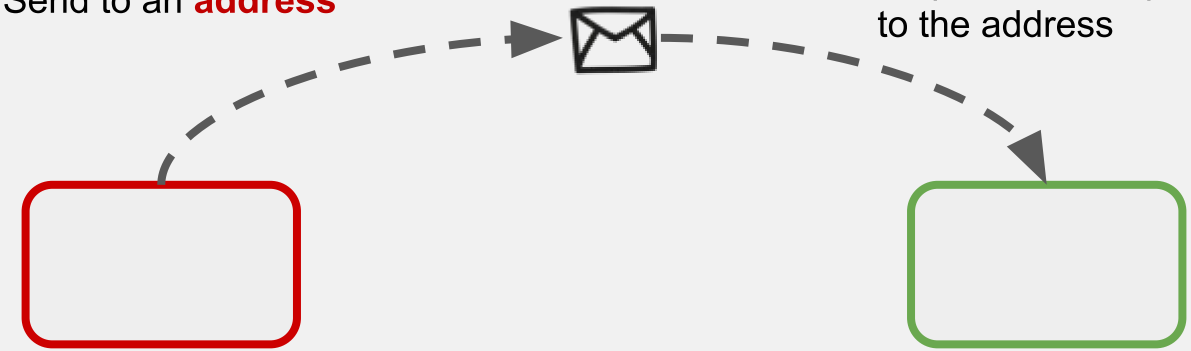
=> Responsiveness

Asynchronous message passing

Components interacts using **messages**

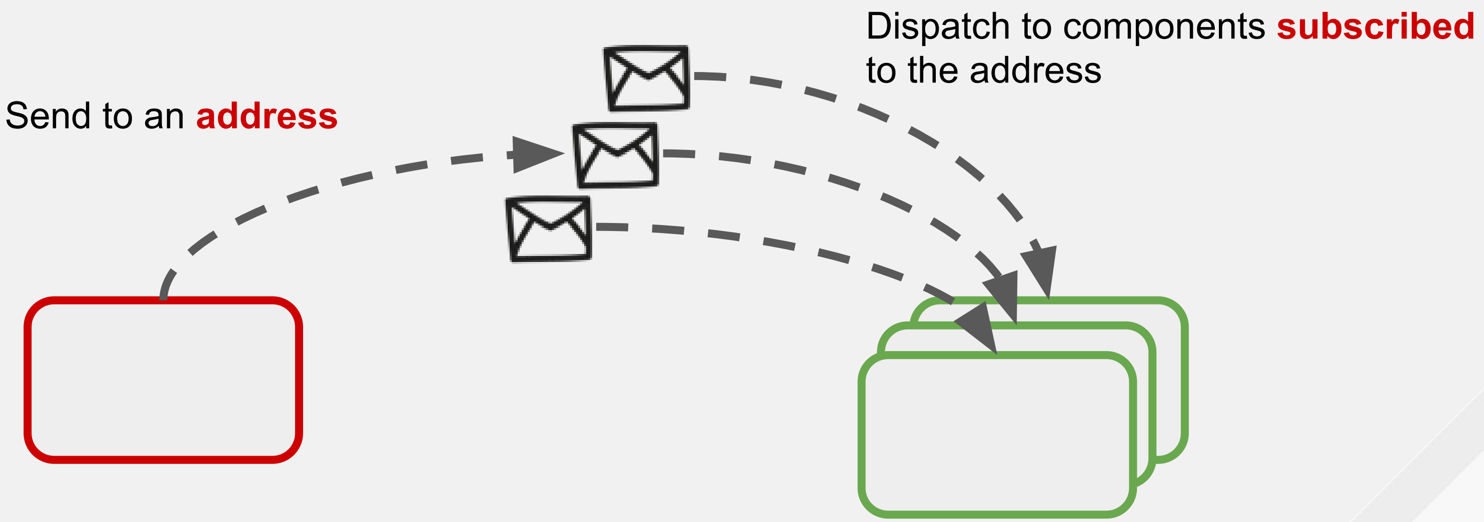
Send to an **address**

Dispatch to components **subscribed** to the address



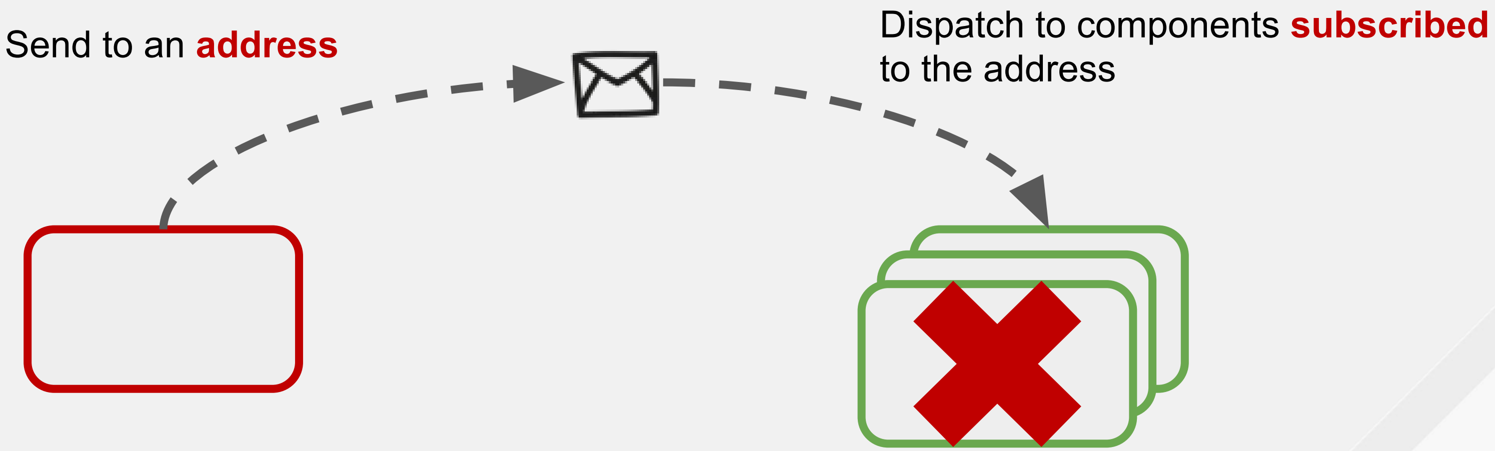
Asynchronous message passing => Elasticity

Messages allows **elasticity**



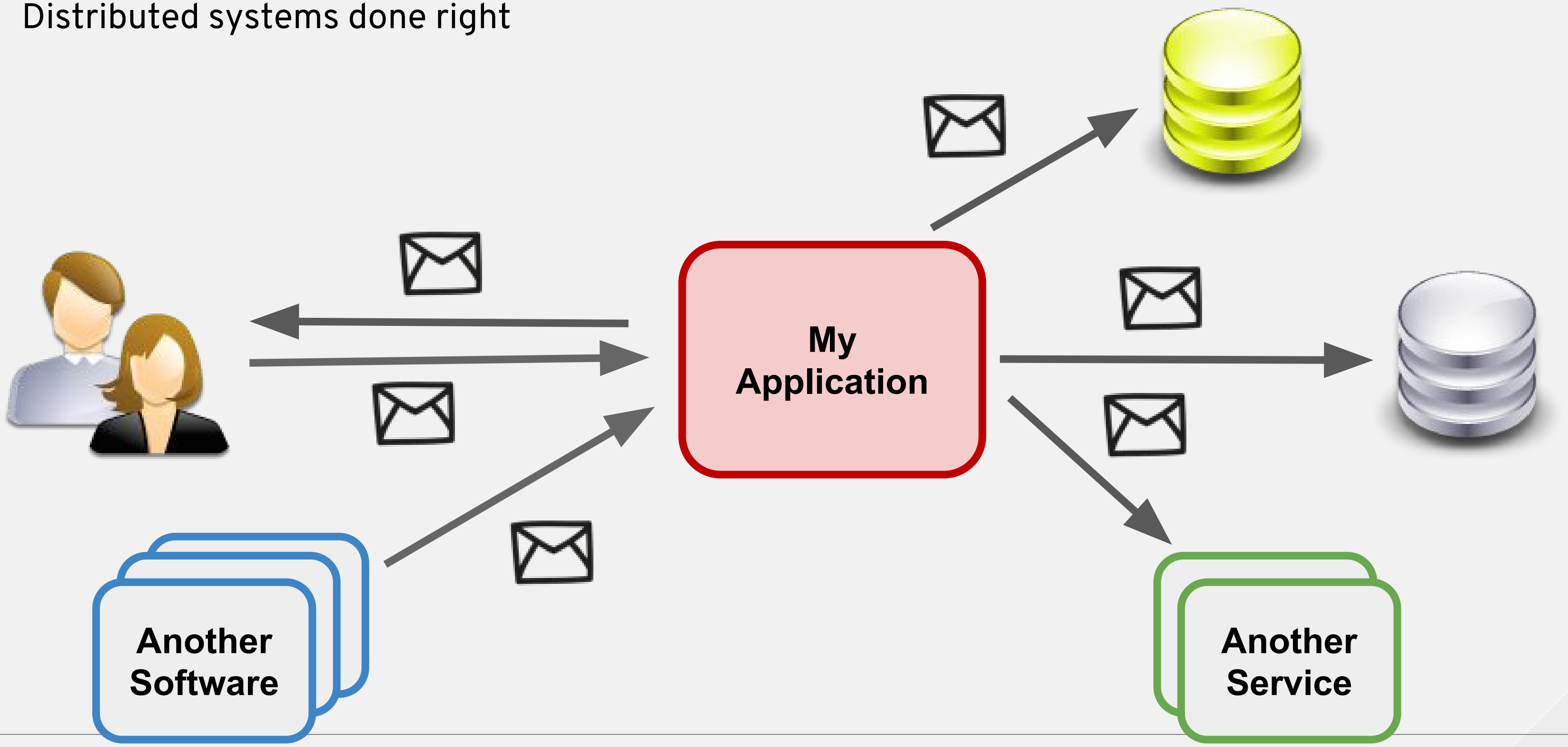
Asynchronous message passing => Resilience

Resilience is not only about failures, it’s also about **self-healing**



So, it’s simple, right ?

Distributed systems done right



Pragmatic reactive systems

And that's what Vert.x offers to you

Development model => Embrace **asynchronous**

Simplified concurrency => **Event-loop** not thread-based

I/O

- **Non-blocking I/O**, if you can't isolate
- HTTP, TCP, Messaging
- RPC

Asynchronous development model

Asynchronous development

Reality check....

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

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

Asynchronous development

Reality check....

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

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

```java
public void compute(int a, int b,
  Handler<Integer> handler) {
     int i = a + b;
     handler.handle(i);
}
```

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

Asynchronous development

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

Reactive programming - let’s rewind….

Do we have Excel users in the room ?

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

redhat.

Reactive programming - let's rewind....

Do we have Excel users in the room ?

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

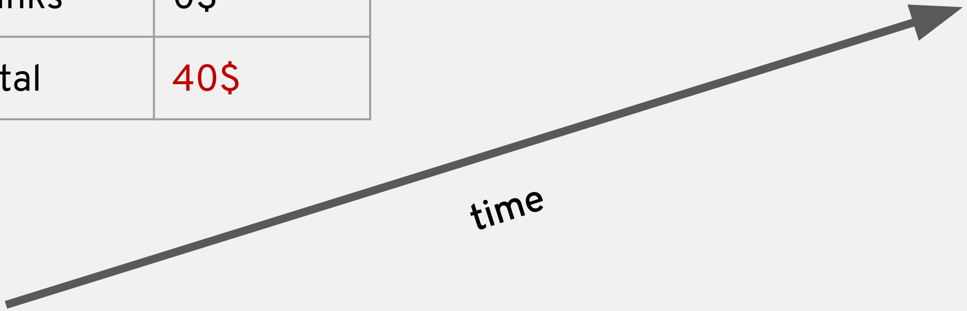
Observe

Streams

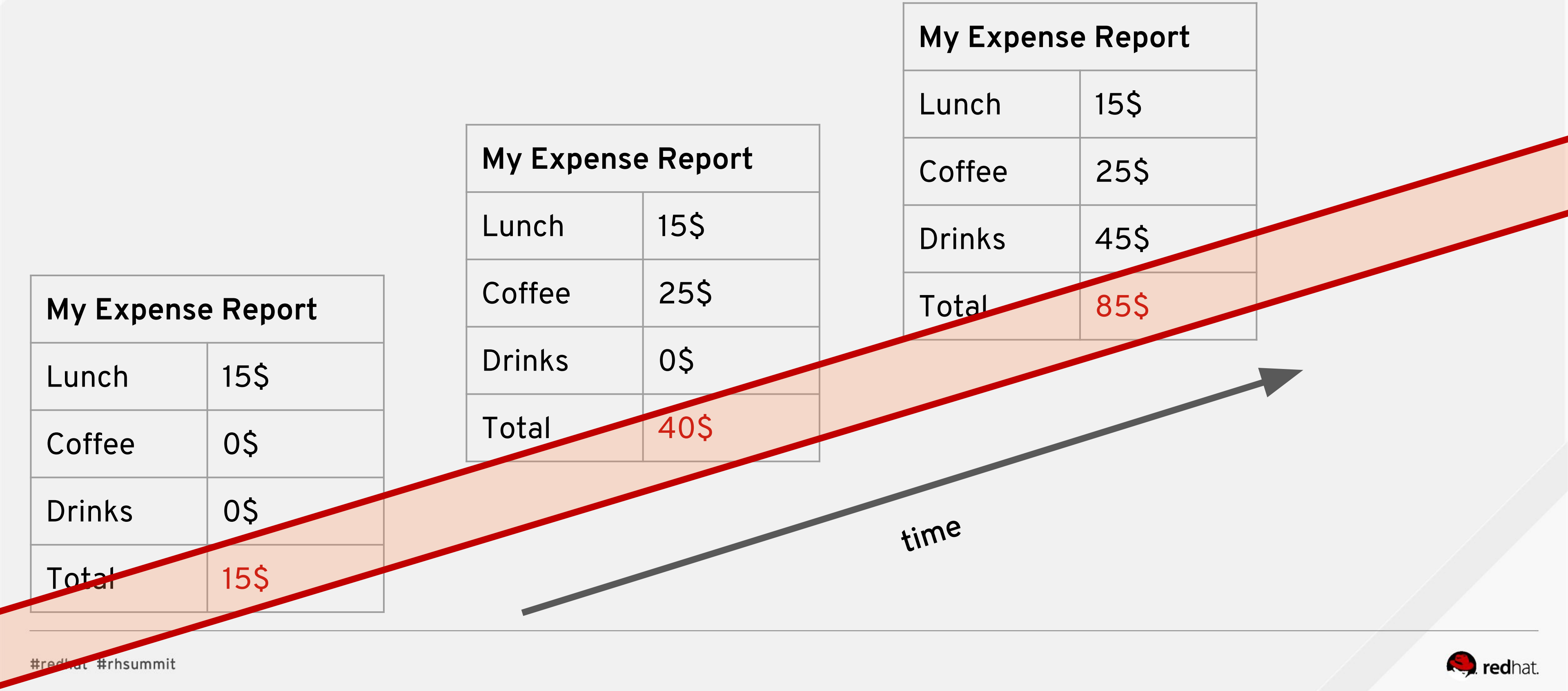
My Expense Report	
Lunch	15$
Coffee	0$
Drinks	0$
Total	15$

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

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



Streams



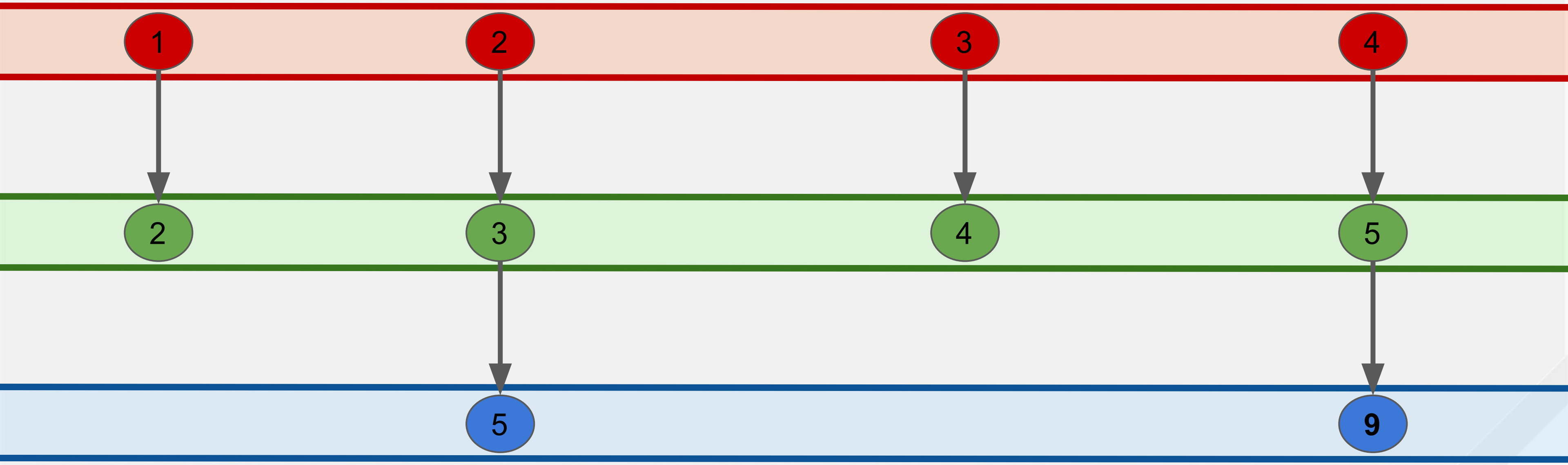
My Expense Report	
Lunch	15$
Coffee	0$
Drinks	0$
Total	15$

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

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

Reactive Programming

Observable and Subscriber



Reactive Extension - RX Java

Click to add subtitle

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

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

Observable<Integer> obs3 = obs2.window(2)
   .flatMap(MathObservable::sumInteger);

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

Reactive types

Observables

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

```
observable.subscribe(
  val -> { /* new value */ },
  error -> { /* failure */ },
  () -> { /* end of data */ }
);
```

Singles

- Stream of one value
- Data, Error

```
single.subscribe(
  val -> { /* the value */ },
  error -> { /* failure */ }
);
```

Completables

- Stream without a value
- Completion, Error

```
completable.subscribe(
  () -> { /* completed */ },
  error -> { /* failure */ }
);
```

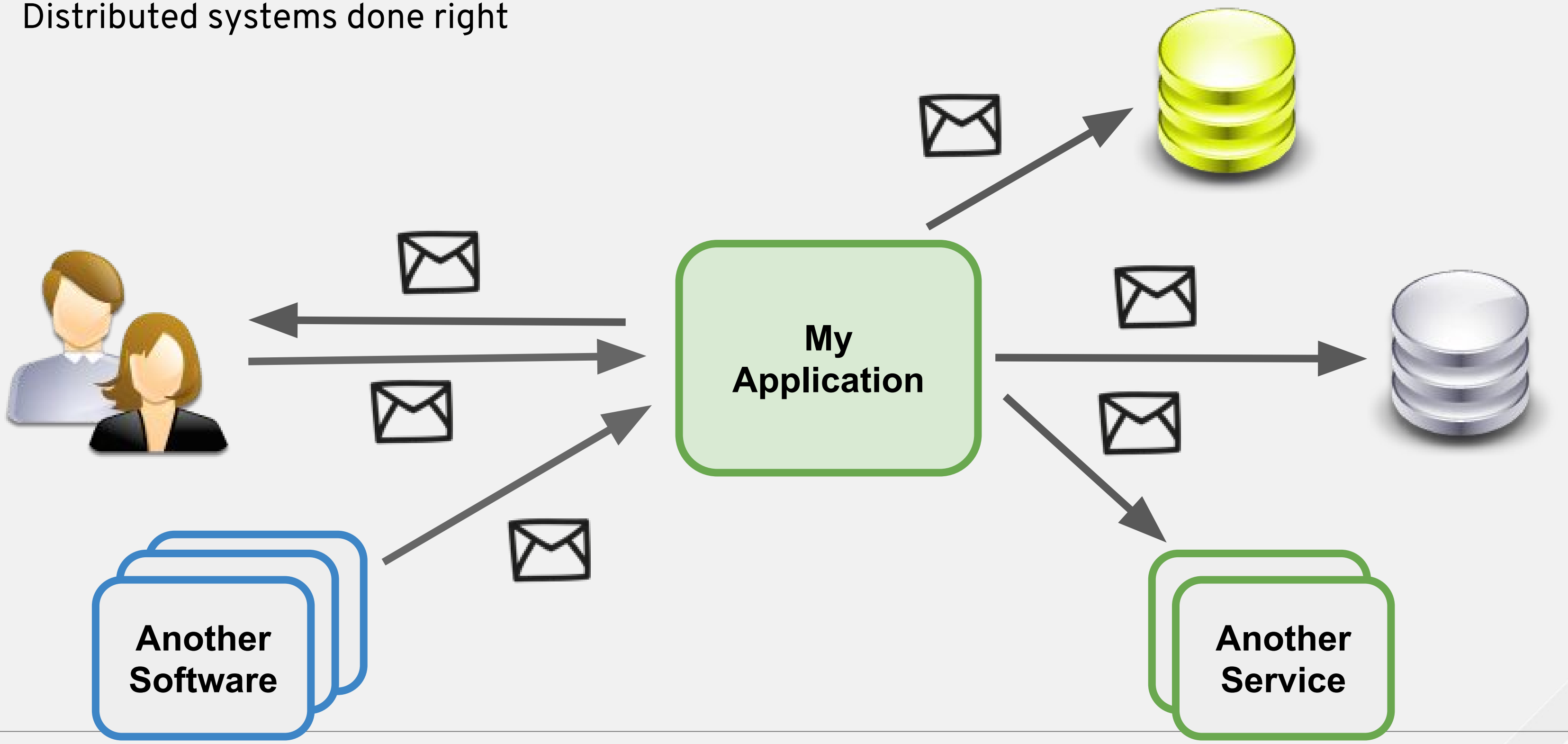
Handling the asynchronous with reactive programming

```
Single<SQLConnection> connection = client.rxGetConnection();
connection
 .flatMapObservable(conn ->
   conn
     .rxQueryStream("SELECT * from PRODUCTS")
     .flatMapObservable(SQLRowStream::toObservable)
     .doAfterTerminate(conn::close)
 )
 .map(Product::new)
 .subscribe(System.out::println);
```

Unleash your superpowers
Vert.x + RX

Back on track

Distributed systems done right



Reactive Web Application

```
public void start() throws Exception {
   Router router = Router.router(vertx);
   router.get("/products").handler(this::list);
   router.route().handler(BodyHandler.create());
   router.post("/products").handler(this::add);
   vertx.createHttpServer().requestHandler(router::accept).listen(8080);
}
private void add(RoutingContext rc) {
   storage.add(rc.getBodyAsString()).subscribe(
           () -> rc.response().setStatusCode(201).end(), rc::fail);
}
private void list(RoutingContext rc) {
   HttpServerResponse response = rc.response().setChunked(true);
   storage.retrieve().subscribe(
           product -> response.write(product + "\n"),
           rc::fail,  response::end);
}
```

Orchestrating remote interactions

Sequential composition

```
WebClient client = ...

client.post("/products")
   .rxSendBuffer(Buffer.buffer("wine"))
   .flatMap(r -> client.get("/products").rxSend())
   .map(HttpResponse::bodyAsString)
   .subscribe(
       System.out::println,
       Throwable::printStackTrace
   );
```

Orchestrating remote interactions

Parallel composition

```
WebClient client =  ...
String[] toDelete = {"cheese", "wine"};

Observable.from(toDelete)
   .flatMapSingle(product ->
       client.delete("/products/" + product).rxSend())
   .toCompletable()
   .andThen(client.get("/products").rxSend())
   .map(HttpResponse::bodyAsString)
   .subscribe(
       System.out::println,
       Throwable::printStackTrace
   );
```

Resilience

Timeout, Retry....

```
client
   .get("/products")
   .rxSend()
   .subscribeOn(RxHelper.scheduler(context))
   .timeout(5, TimeUnit.SECONDS)
   .retry(1)
   .map(HttpResponse::bodyAsString)
   .onErrorReturn(t -> "unable to retrieve the list")
   .subscribe(
       System.out::println
   );
```

Vert.x + RX

RX-ified API

- *rx* methods are returning Single
- ReadStream provides a toObservable method
- Use RX operator to combine, chain, orchestrate asynchronous operations
- Use RX reactive types to be notified on messages (*Observable*)

Follows Vert.x execution model

- Single-threaded, Event loop
- Provide a RX scheduler

What you can do with it

- Messaging (event bus), HTTP 1 & 2 client and server, TCP client and server, File system
- Async data access (JDBC, MongoDB, Redis...)

The path to better systems

All you need is (reactive) love

Reactive Systems + **Reactive Programming** = ❤

RED HAT SUMMIT

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos

redhat.