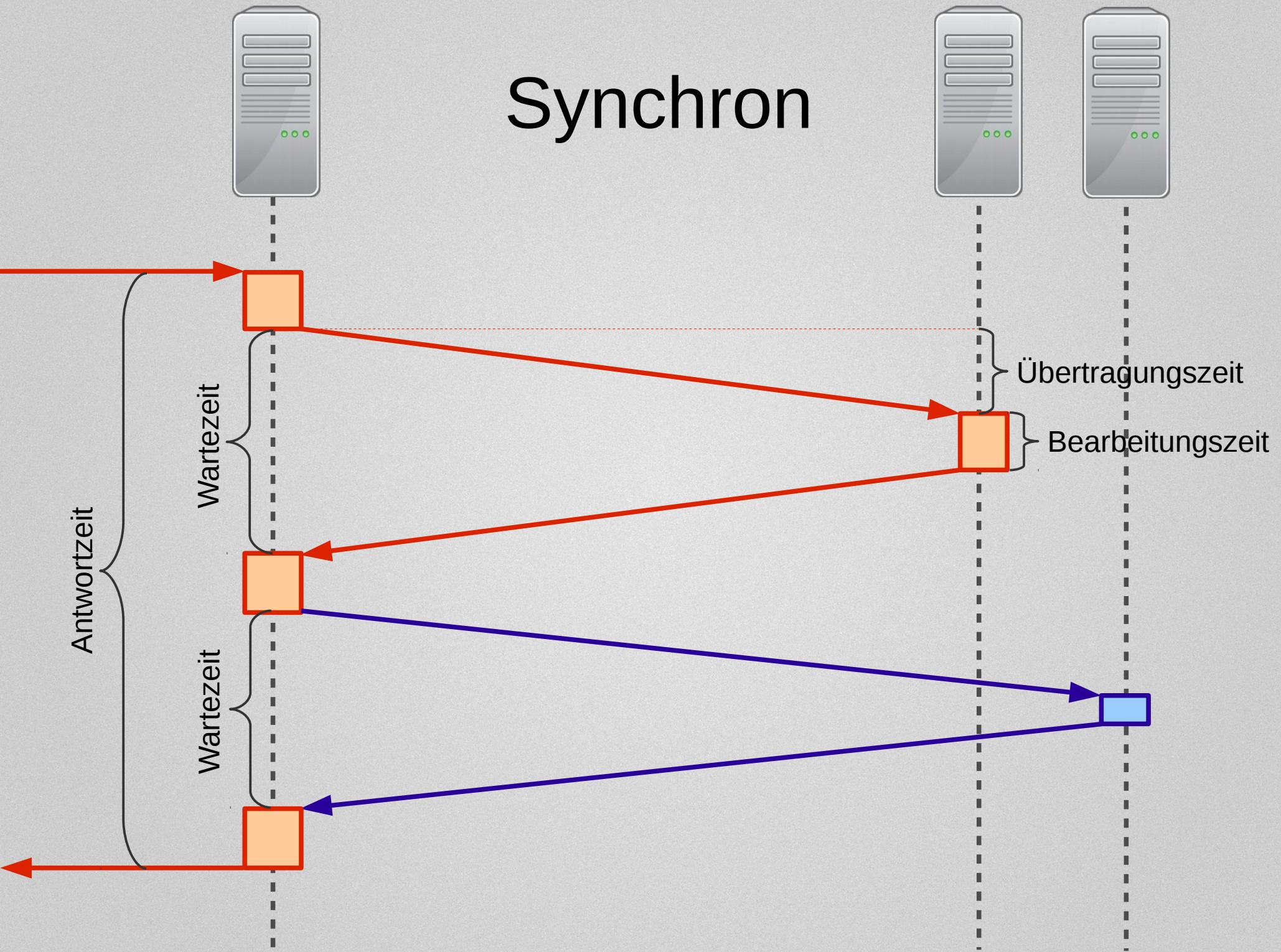
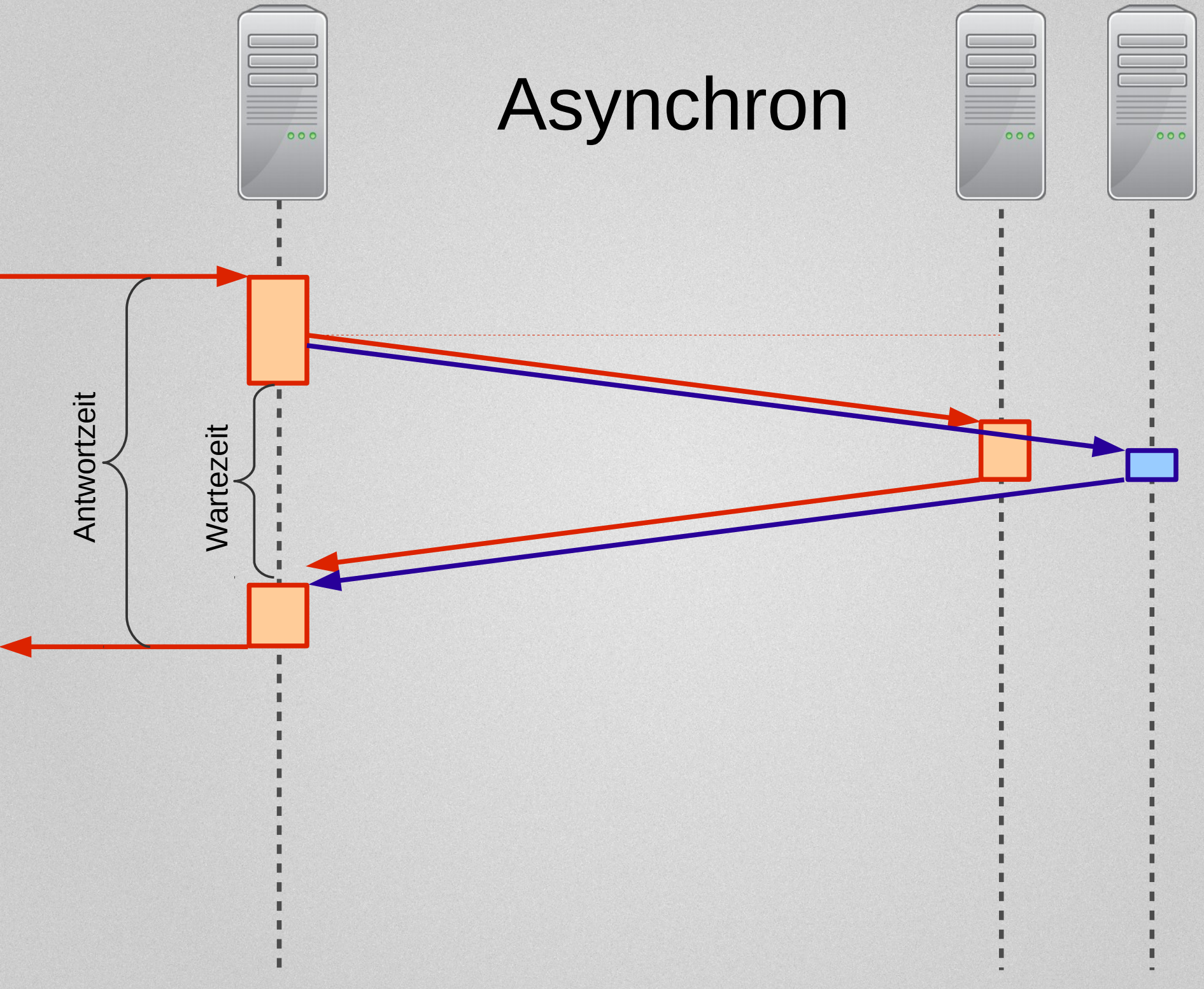
Asynchronen Code testen

Quaagilis
Andreas Simon
@ndrssmn

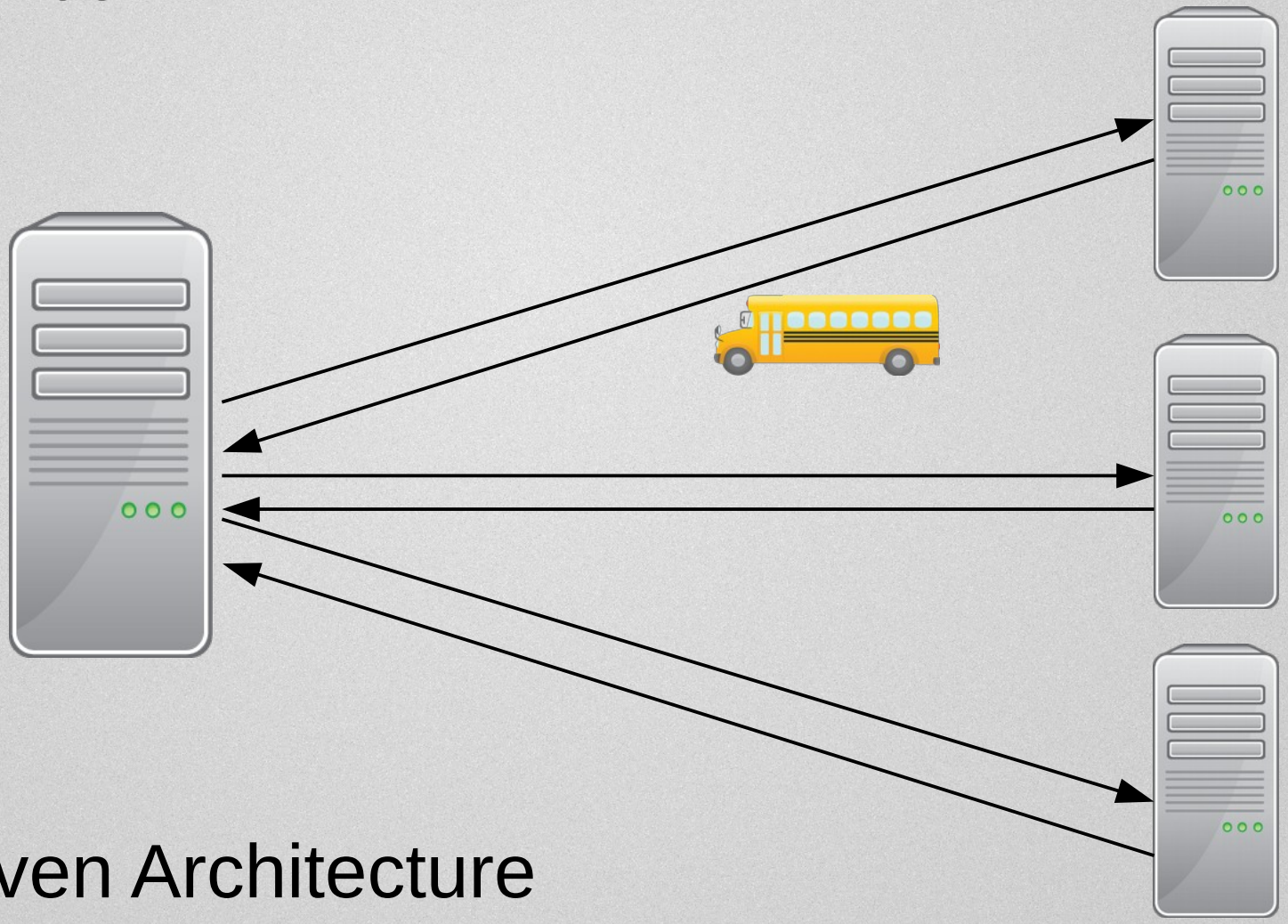
Synchron


Übertragungszeit
Bearbeitungszeit
Wartezeit
Wartezeit
Antwortzeit

Asynchron

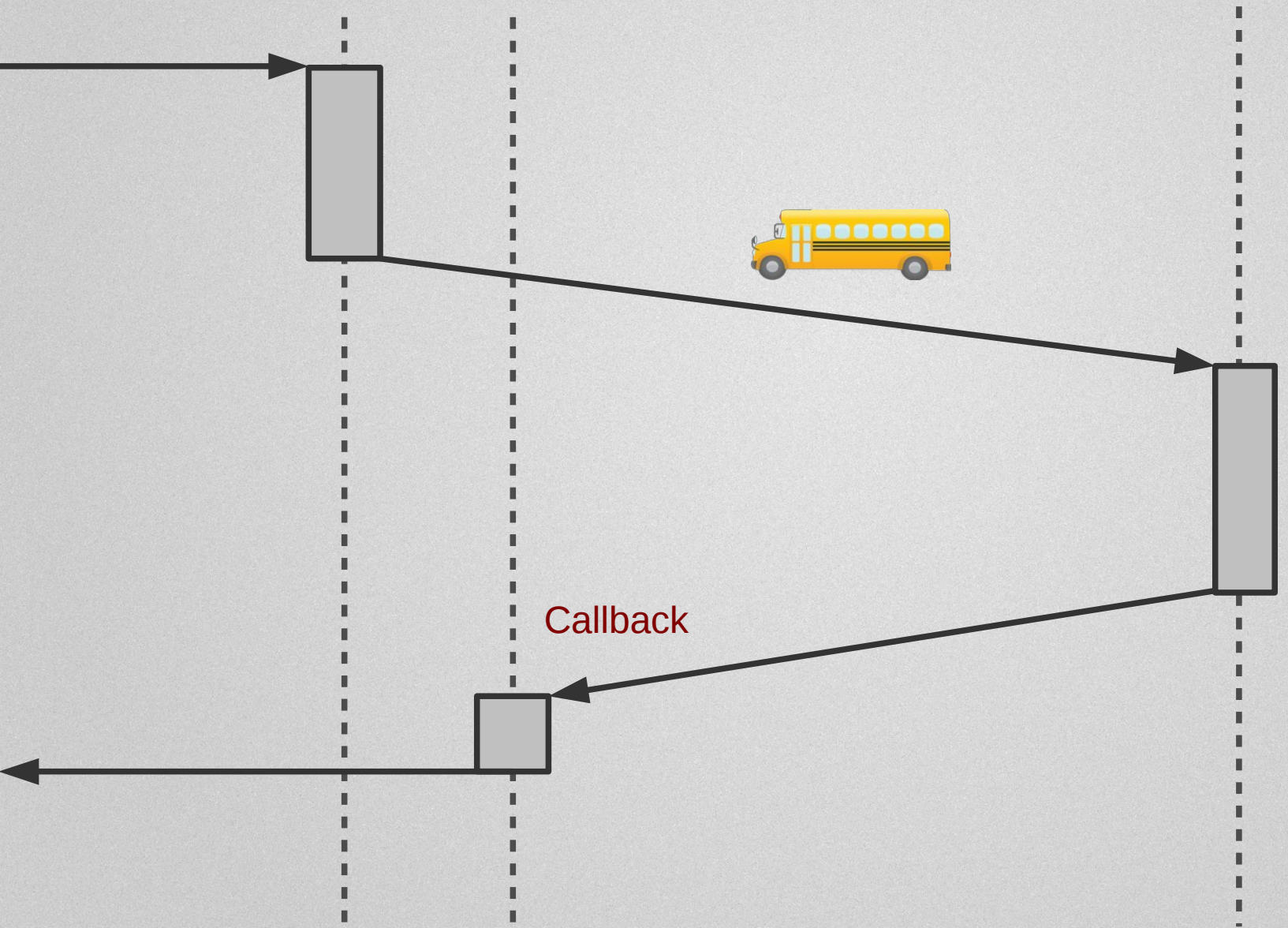


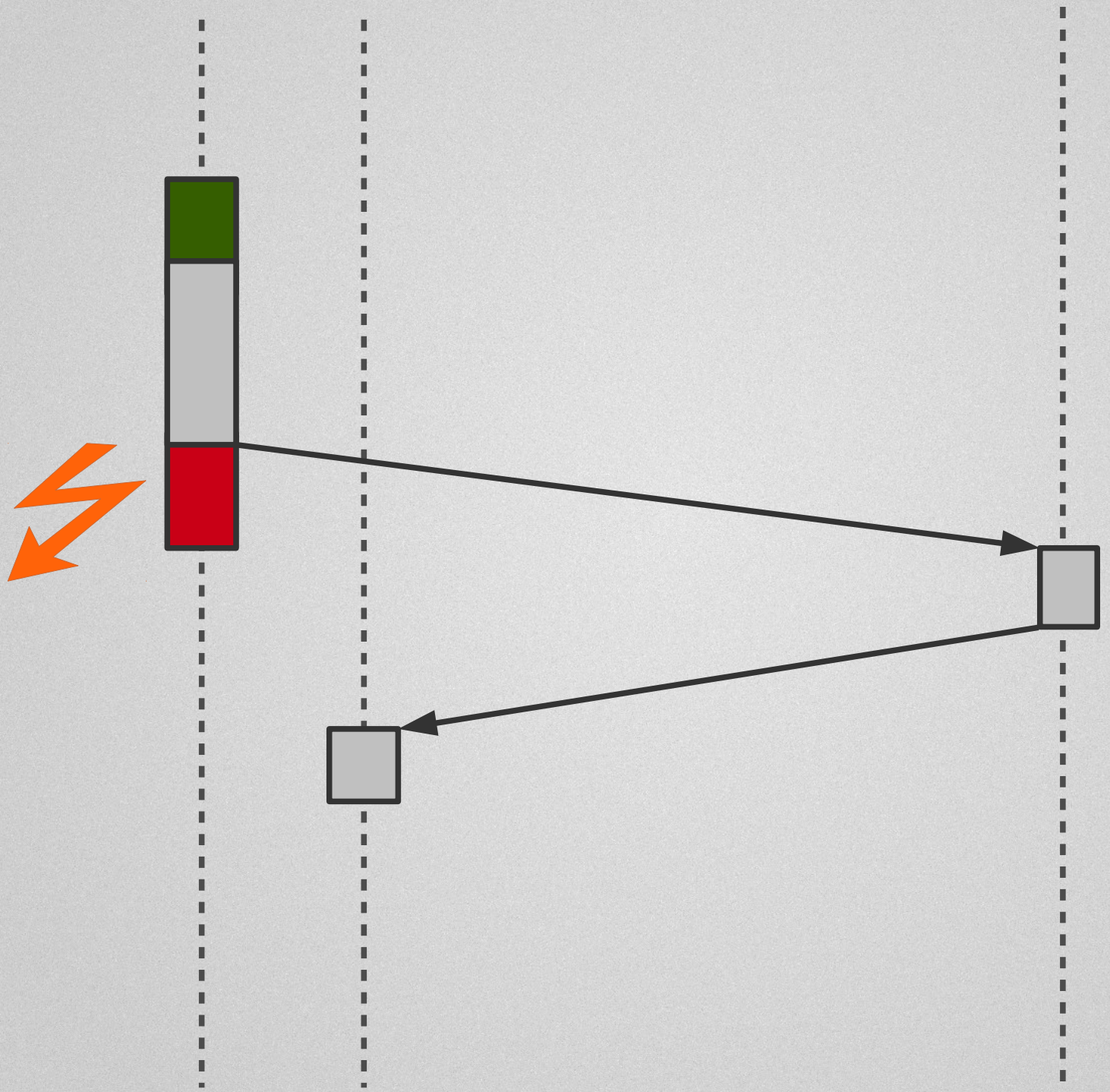
Fehlertoleranz
Verfügbarkeit
Parallelisierung
Performance

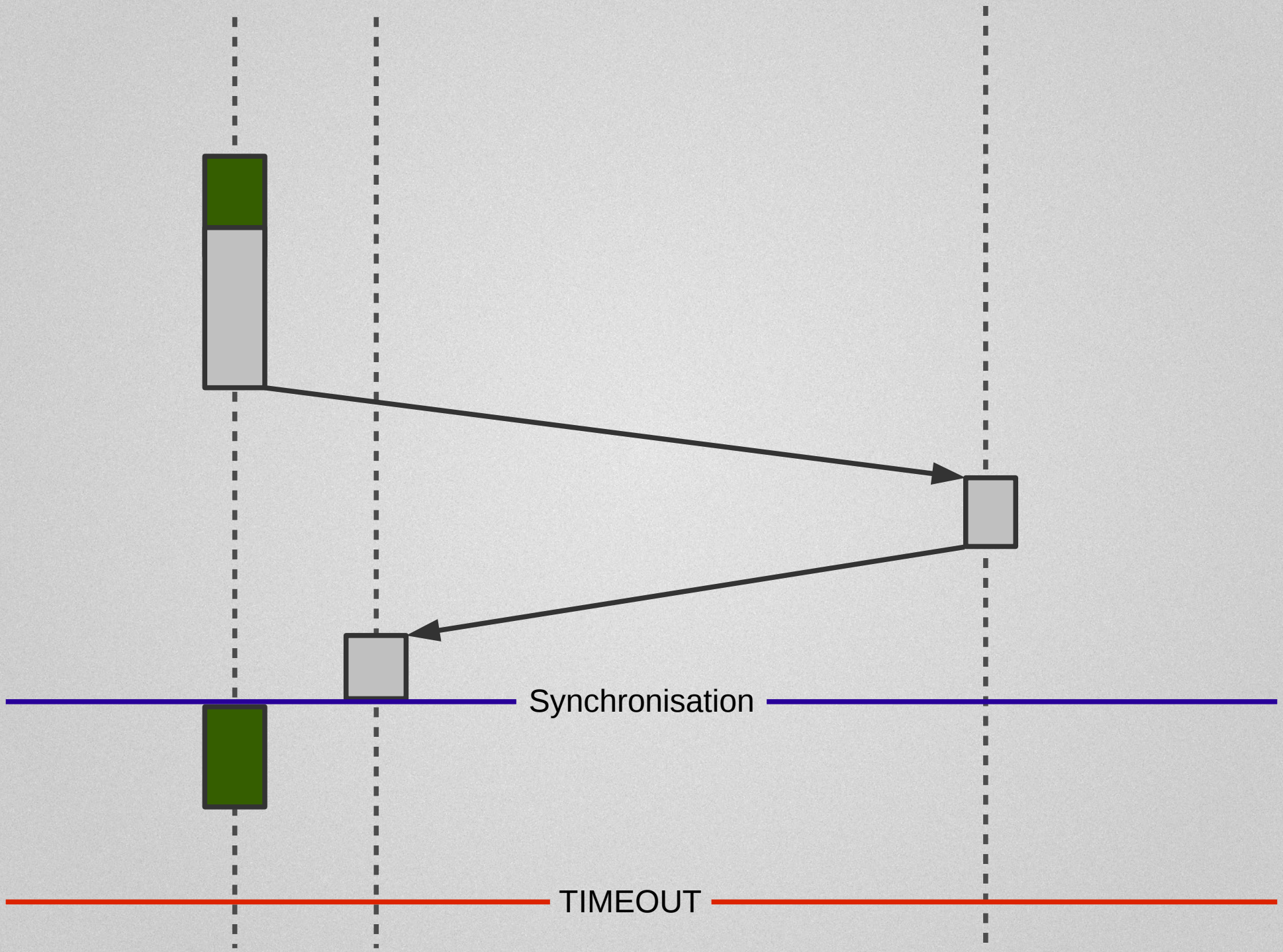


Event-Driven Architecture

Listening






Synchronisation
TIMEOUT

Listening

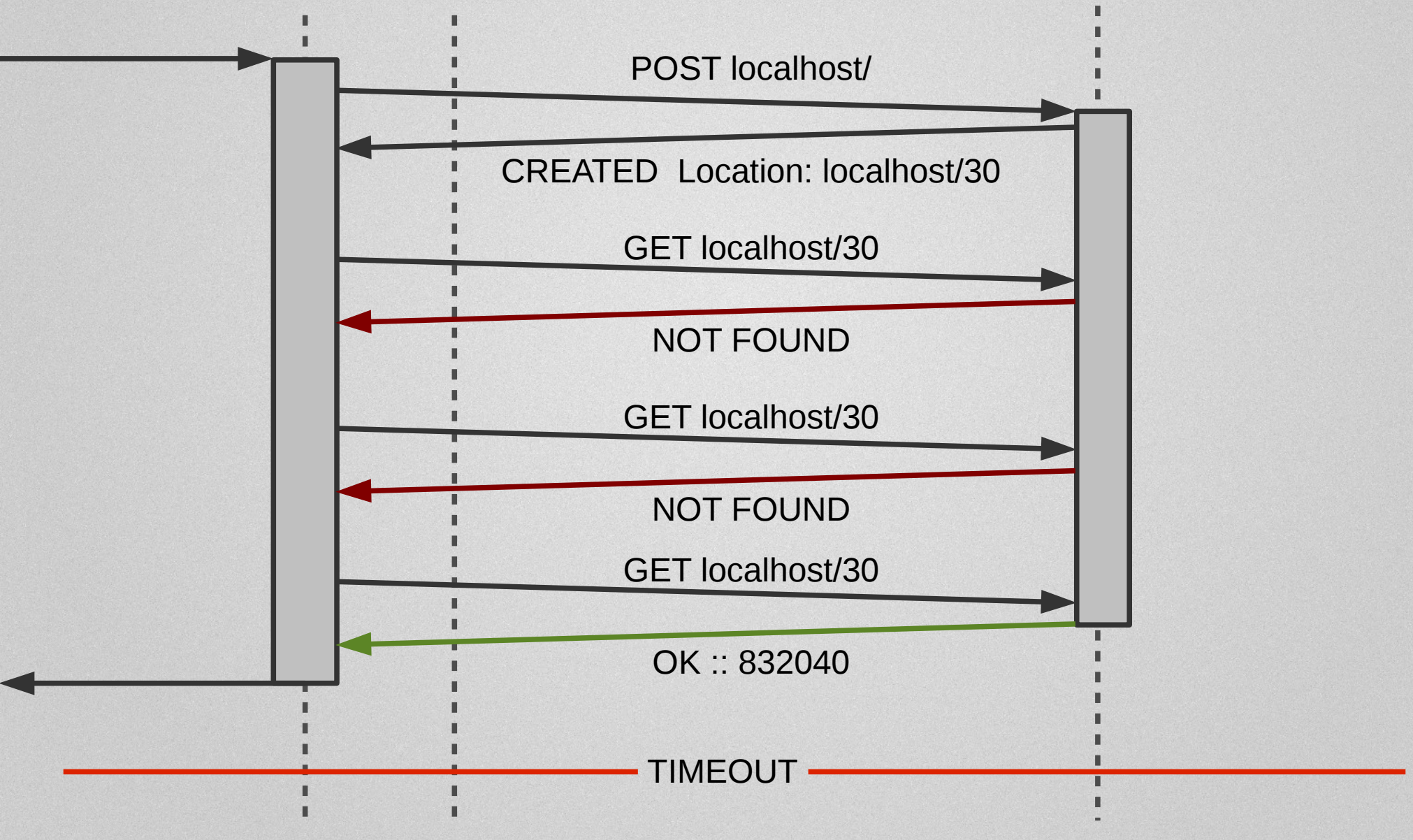
```
@Test public void
should_reply_with_Fibonacci_numbers() throws Exception {
  // Arrange
  NotificationTrace<Integer> trace = new NotificationTrace<>(TIMEOUT);
  String replyQueue = channel.queueDeclare().getQueue();
  FibonacciCalculator.create(connection.createChannel());

  new IntegerConsumer(
    connection.createChannel(),
    trace::append)
  .consumeQueue(replyQueue);

  // Act
  publishNumbers(MIN, MAX, replyQueue);

  // Assert
  trace.containsNotification(equalTo(FIB_MIN));
  trace.containsNotification(equalTo(FIB_MAX));
}
```

Sampling


POST localhost/
CREATED Location: localhost/30
GET localhost/30
NOT FOUND
GET localhost/30
NOT FOUND
GET localhost/30
OK :: 832040
TIMEOUT

Sampling

```
@Test public void
calculates_fib_30() throws Exception {
  // Act
  connection = POST("http://localhost:3000/", "30");
  fibLocation = connection.getHeaderField("Location");

  // Assert
  Probe probe = responseTo(
    fibLocation,
    equalTo(Integer.toString(FIB_30))
  );
  new Poller(TIMEOUT, POLL_DELAY).check(probe);
}
```

```
public class Poller {
  […]
  public void check(Probe probe) {
    […]
    while (!probe.isSatisfied()) {
      […]
      Thread.sleep(pollDelayMillis);
      probe.sample();
    }
  }
}
```

```
public interface Probe {
  void sample();
  boolean isSatisfied();
  void describeAcceptanceCriteriaTo(Description d);
  void describeFailureTo(Description d);
}
```

Test the test

```
@Test public void
is_thread_safe() throws Exception {
  latch = startStressing(STRESSING_THREADS, () -> {
      for (int i = 0; i < ITERATIONS; i++) {
        trace.append("NOT-WANTED");
        Thread.sleep(SLEEPTIME);
      }
      latch.countDown();
  });
  scheduler.schedule(
    () -> trace.append("WANTED"),
    100, TimeUnit.MILLISECONDS
  );

  trace.containsNotification(equalTo("WANTED"));
  latch.await();
  assertThat(
    trace.getAppendCount(),
    is(equalTo((long) STRESSING_THREADS * ITERATIONS + 1))
  );
}
```

Thread-sicher implementieren

```
public class NotificationTrace<T> {

  public void append(T message) {
    synchronized (traceLock) {
      trace.add(message);
      traceLock.notifyAll();
    }
  }

  public void containsNotification(Matcher<? super T> criteria)
    throws AssertionError, InterruptedException {
    Timeout timeout = new Timeout(timeoutMs);

    synchronized (traceLock) {
      stream = new NotificationStream<>(trace, criteria);
      while (!stream.hasMatched()) {
        if (timeout.hasTimedOut()) {
          throw new AssertionError();
        }
        timeout.waitOn(traceLock);
      }
    }
  }
```

Aufräumen

```
@After
public void tearDown() throws InterruptedException {
  executorService.shutdownNow();
  scheduler.shutdownNow();
}
```

Fazit

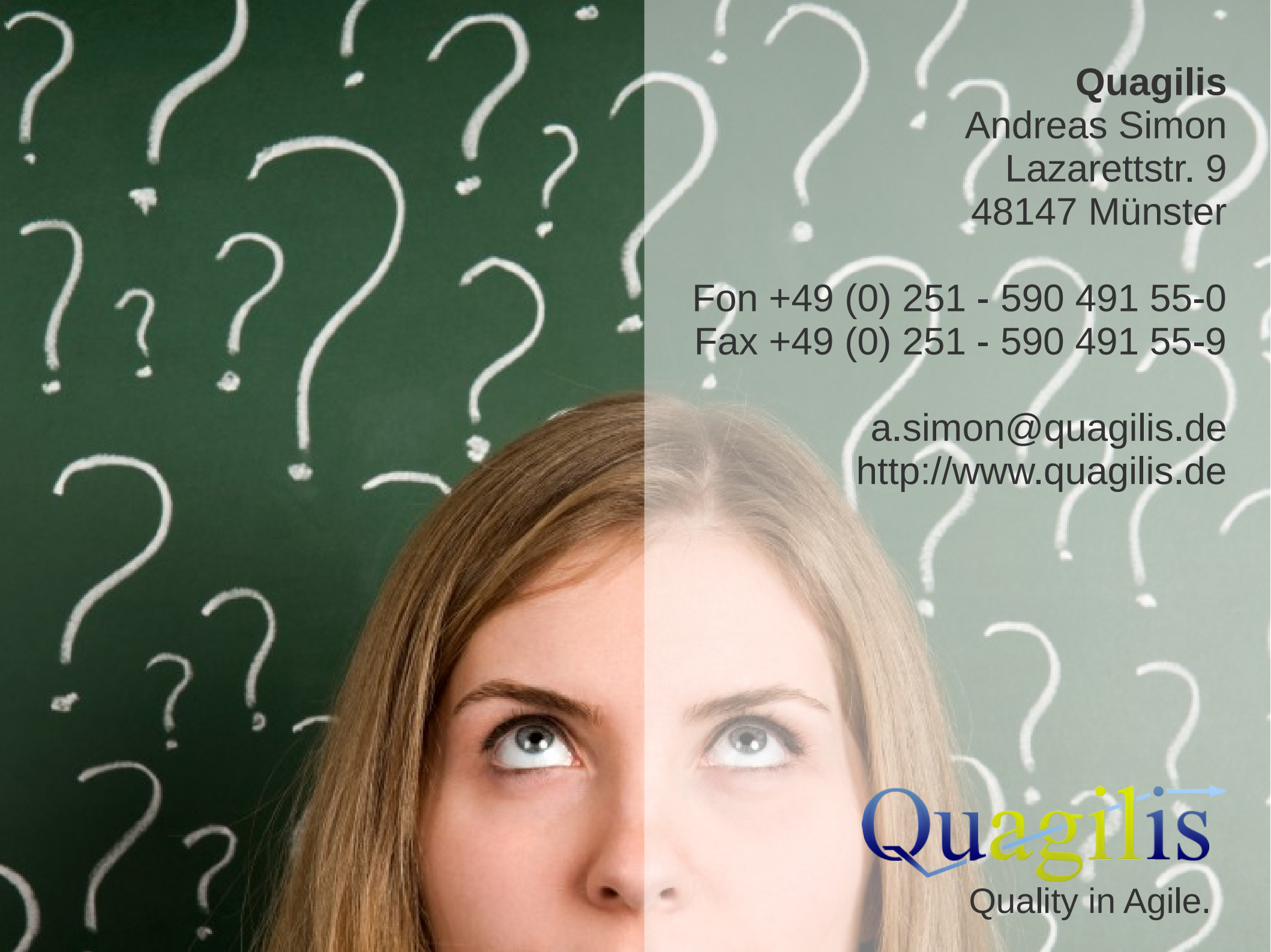
Listening vs. Sampling

Synchronisierungsmechanismen kapseln (und durch Unit-Tests validieren)

Brian Goetz: "Java Concurrency in Practice"

Nat Pryce, Steve Freeman: "Growing Object-Oriented Software"

https://github.com/andreassimon/talk-asynchronen-code-testen



Quagilis
Andreas Simon
Lazarettstr. 9
48147 Münster

Fon +49 (0) 251 - 590 491 55-0
Fax +49 (0) 251 - 590 491 55-9

a.simon@quagilis.de
http://www.quagilis.de

Quagilis

Quality in Agile.