ScalaBay October Meetup

Agents and Agency in the Internet:

How we get there is where we are going

L.G. Meredith
Managing Partner, Biosimilarity LLC

Outline

- A few verbs
- Their implementation
- A b2b application using them
- A c2c application using them
- Summary
- Questions

A few verbs

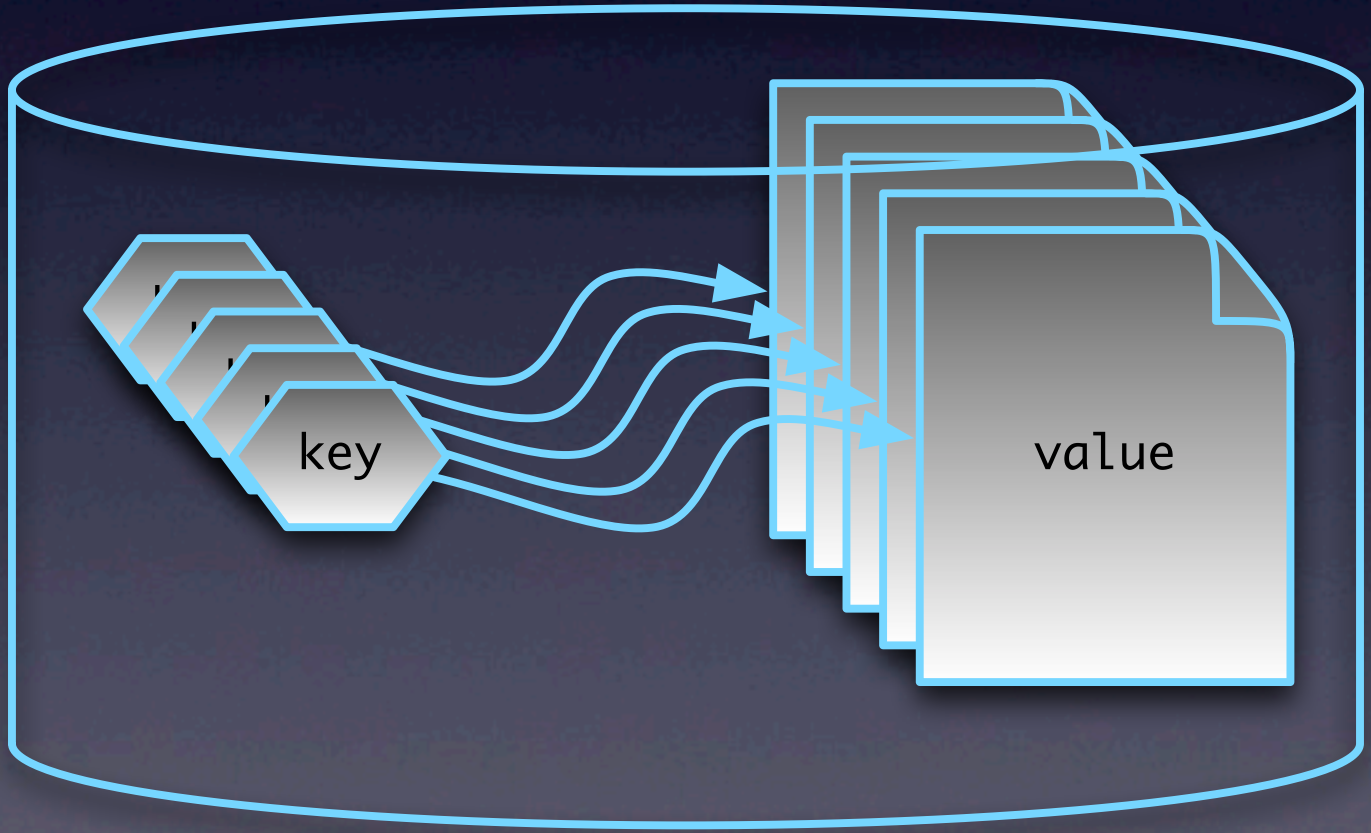
- These verbs are designed to make it easy for distributed applications talk to one another
- They are built around the constructive interpretation of a single logical principle:

A or ~A

ScalaBay October Meetup

A few verbs

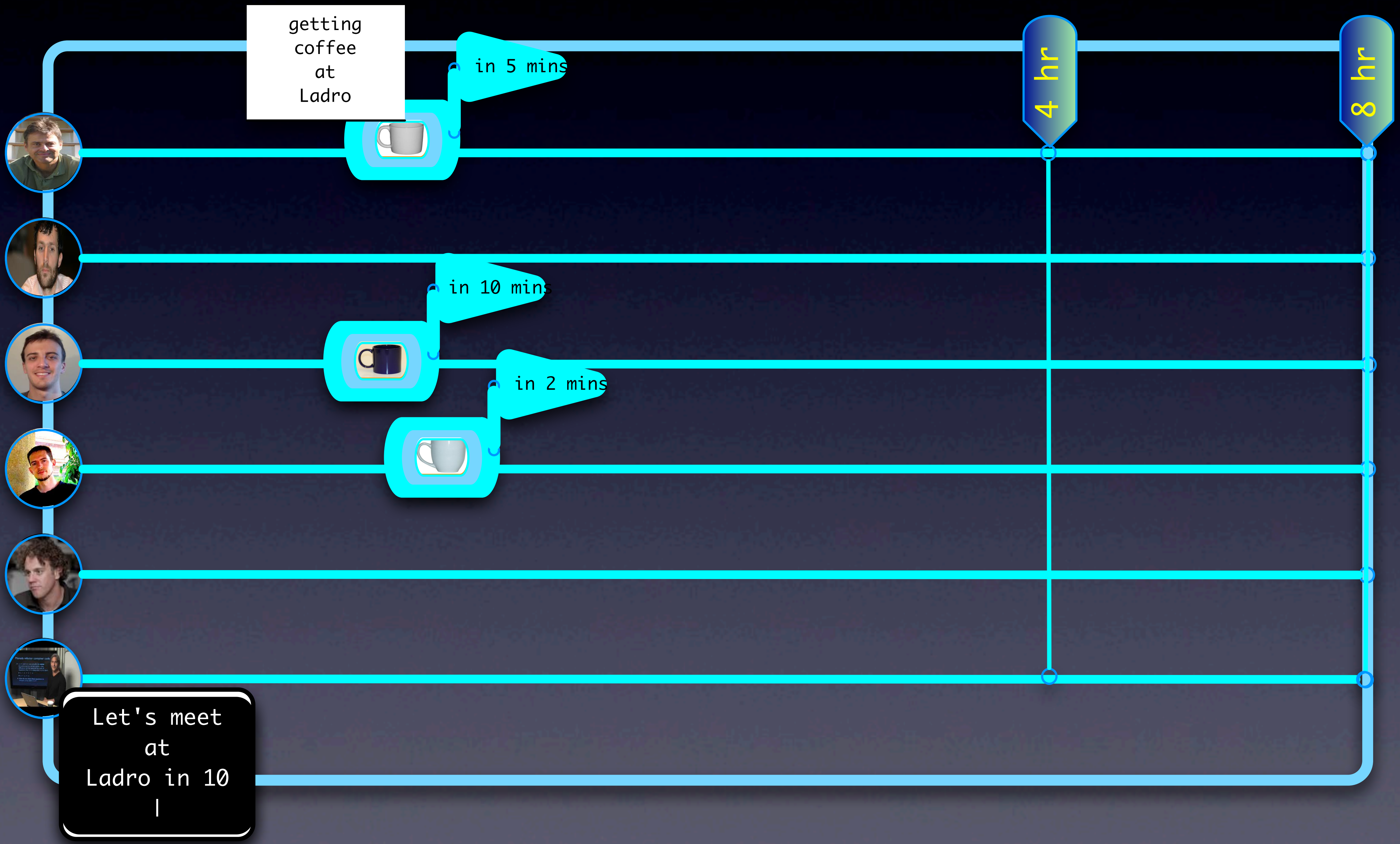
Let’s illustrate with a cartoon.



L.G. Meredith, Managing Partner, Biosimilarity LLC

ScalaBay October Meetup

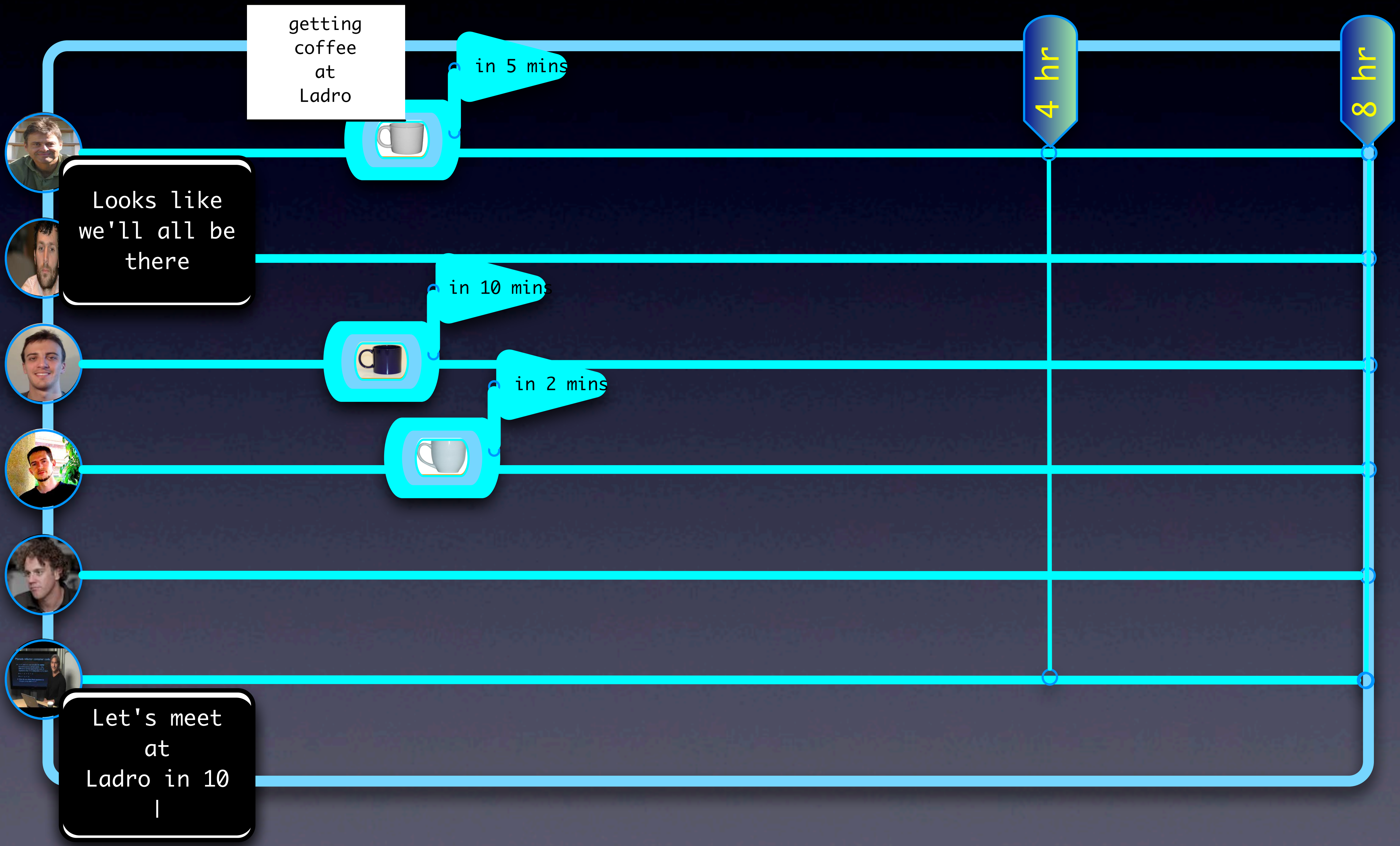
Let’s imagine an app...



L.G. Meredith, Managing Partner, Biosimilarity LLC

ScalaBay October Meetup

Let’s imagine an app...



L.G. Meredith, Managing Partner, Biosimilarity LLC

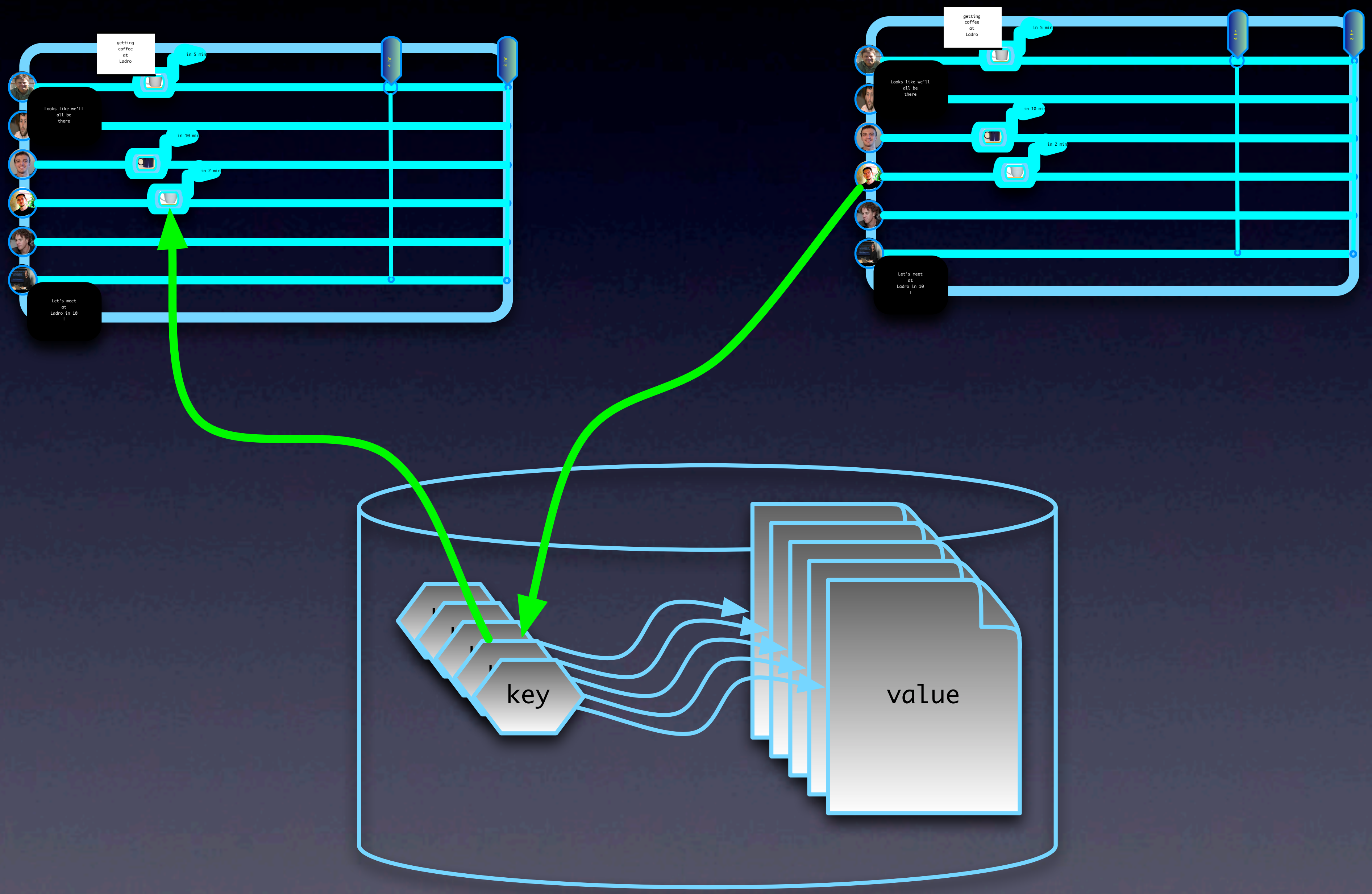
A few verbs

An application like this could benefit from the ability to

- Make a standing query
- Post data to standing queries
- Store data conventionally
- Read data conventionally

ScalaBay October Meetup

A few verbs

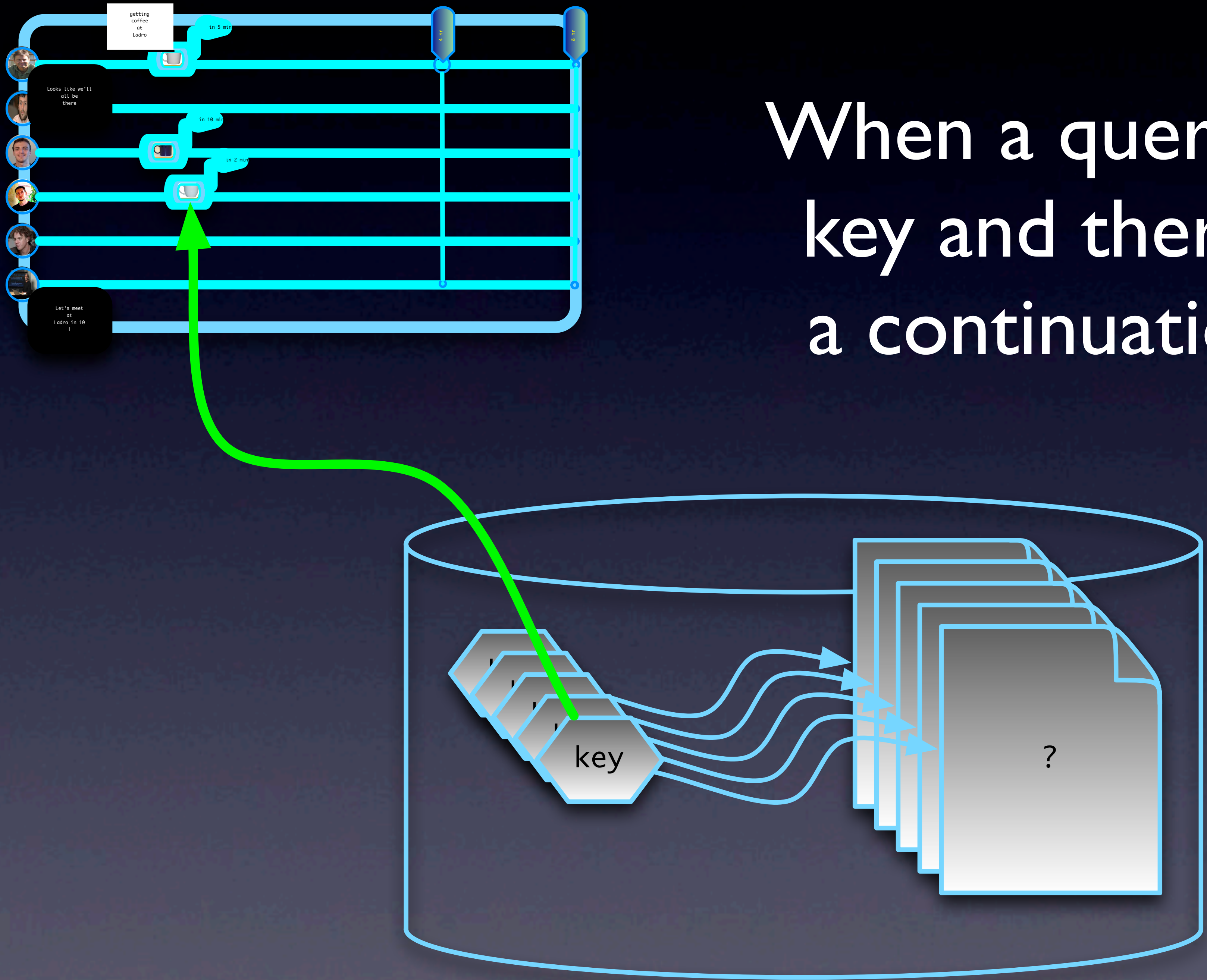

getting coffee at Ladro
in 5 min
Looks like we'll all be there
in 10 min
in 2 min
Let's meet at Ladro in 10
getting coffee at Ladro
in 5 min
Looks like we'll all be there
in 10 min
in 2 min
Let's meet at Ladro in 10
key
value

L.G. Meredith, Managing Partner, Biosimilarity LLC

ScalaBay October Meetup

A few verbs

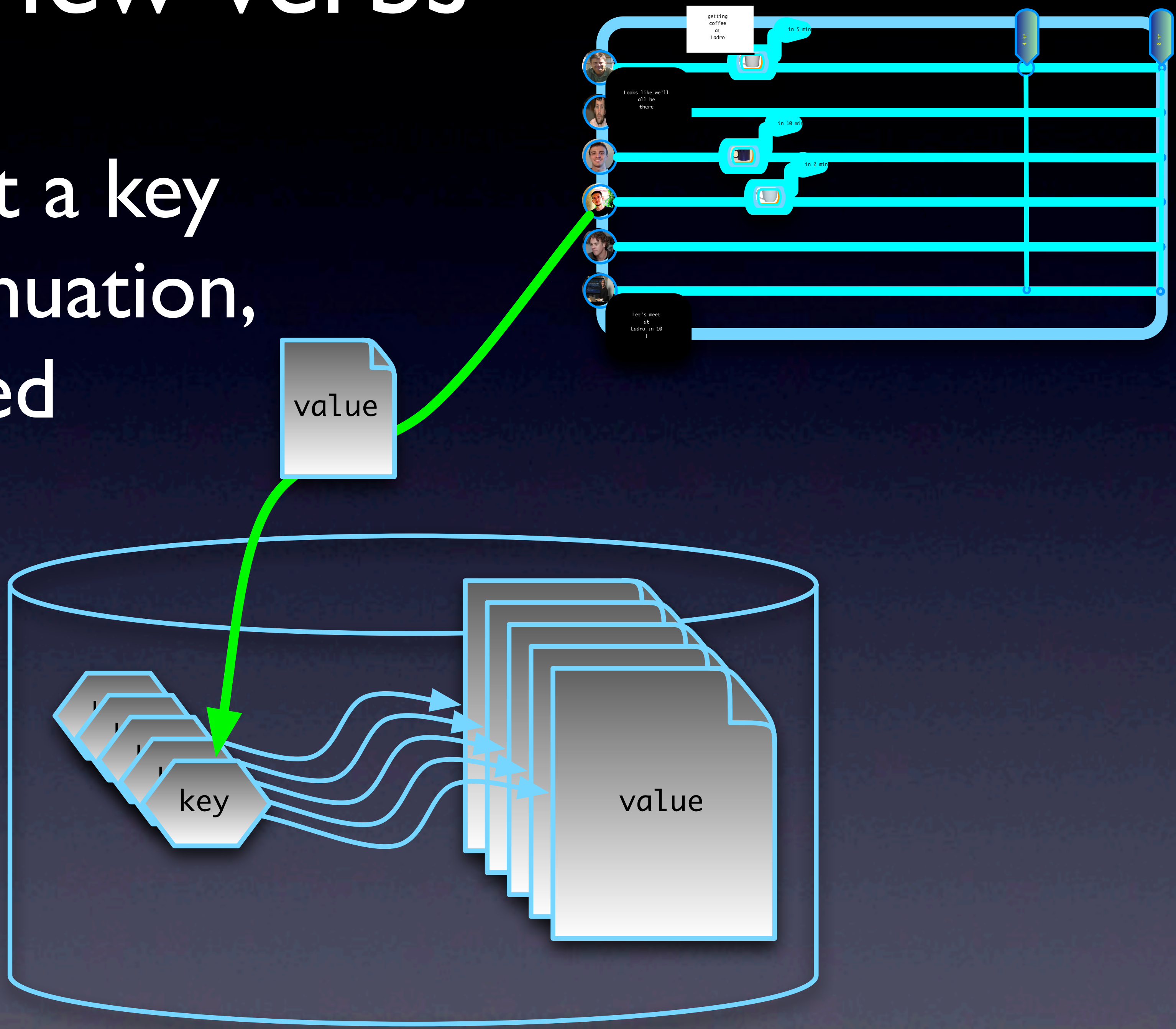
When a query arrives at a key and there is no data a continuation is stored



L.G. Meredith, Managing Partner, Biosimilarity LLC

A few verbs

When data arrives at a key
and there is no continuation,
the data is stored



getting coffee at Ladro

Looks like we'll all be there

Let's meet at Ladro in 10

A few verbs

We can make some decisions about the nature of the interaction with respect to the stored entity (continuation or data)

- The action can consume the stored entity
- The action can make a copy the stored entity and allow interaction on the copy

A few verbs

	ephemeral - data ephemeral - k	persistent - data ephemeral - k	ephemeral - data peristent - k	persistent - data persistent - k
producer	put	store	publish	publish with history
consumer	get	read	subscribe	subscribe

A few verbs

	ephemeral - data ephemeral - k	persistent - data ephemeral - k	ephemeral - data peristent - k	persistent - data persistent - k
producer	put	store	publish	publish with history
consumer	get	read	subscribe	subscribe

Traditional DB operations

A few verbs

	ephemeral - data ephemeral - k	persistent - data ephemeral - k	ephemeral - data peristent - k	persistent - data persistent - k
producer	put	store	publish	publish with history
consumer	get	read	subscribe	subscribe

Traditional messaging operations

L.G. Meredith, Managing Partner, Biosimilarity LLC

A few verbs

	ephemeral - data ephemeral - k	persistent - data ephemeral - k	ephemeral - data peristent - k	persistent - data persistent - k
producer	put	store	publish	publish with history
consumer	get	read	subscribe	subscribe

We use these for item-level locking in a distributed setting

A few verbs

How is this related to the constructive interpretation of the logical principle

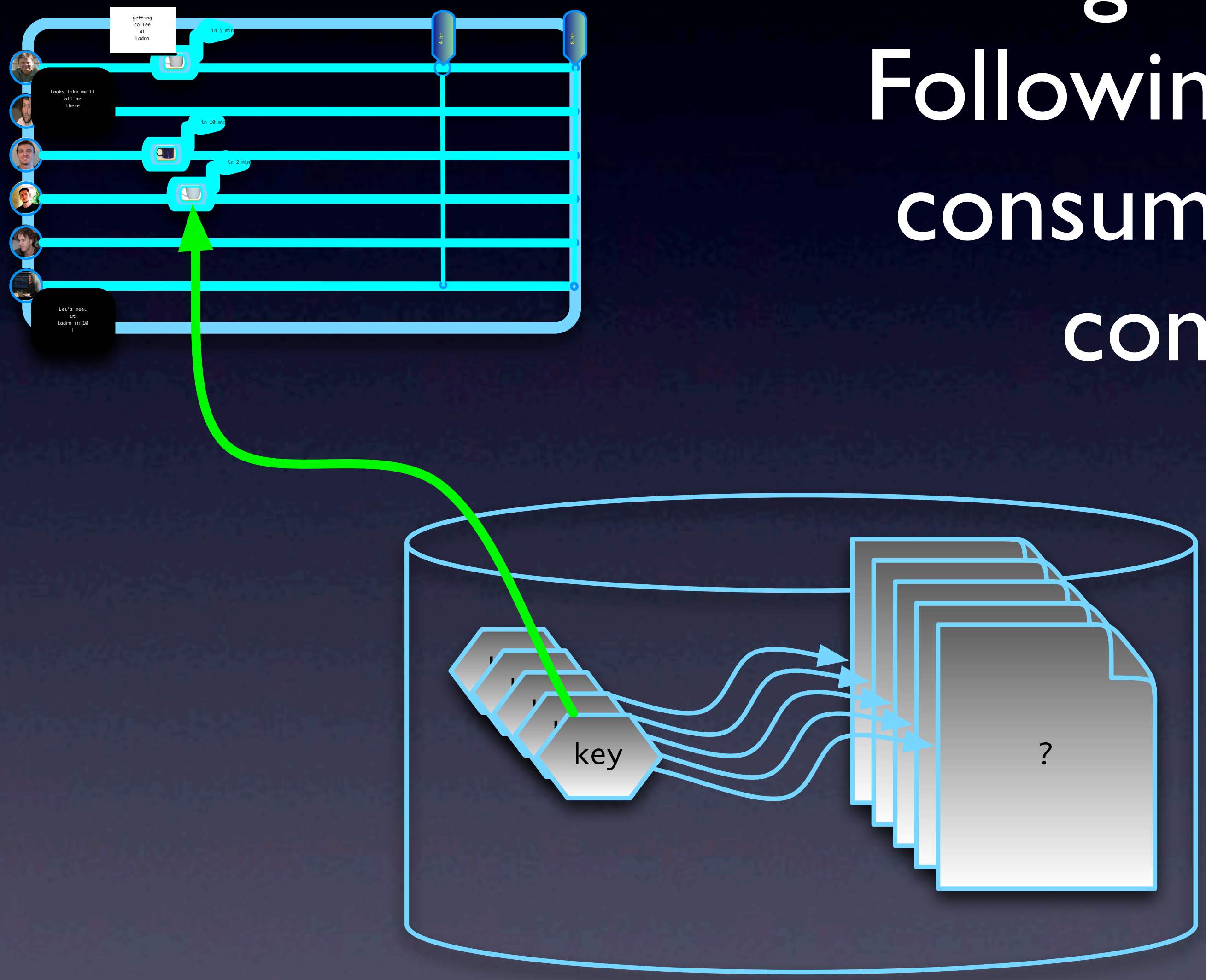
A or ~A

A few verbs

Now, how is this related to the π-calculus?

A few verbs

Let's design a DSL for these verbs. Following LINQ we'll have consumption map to for-comprehensions.

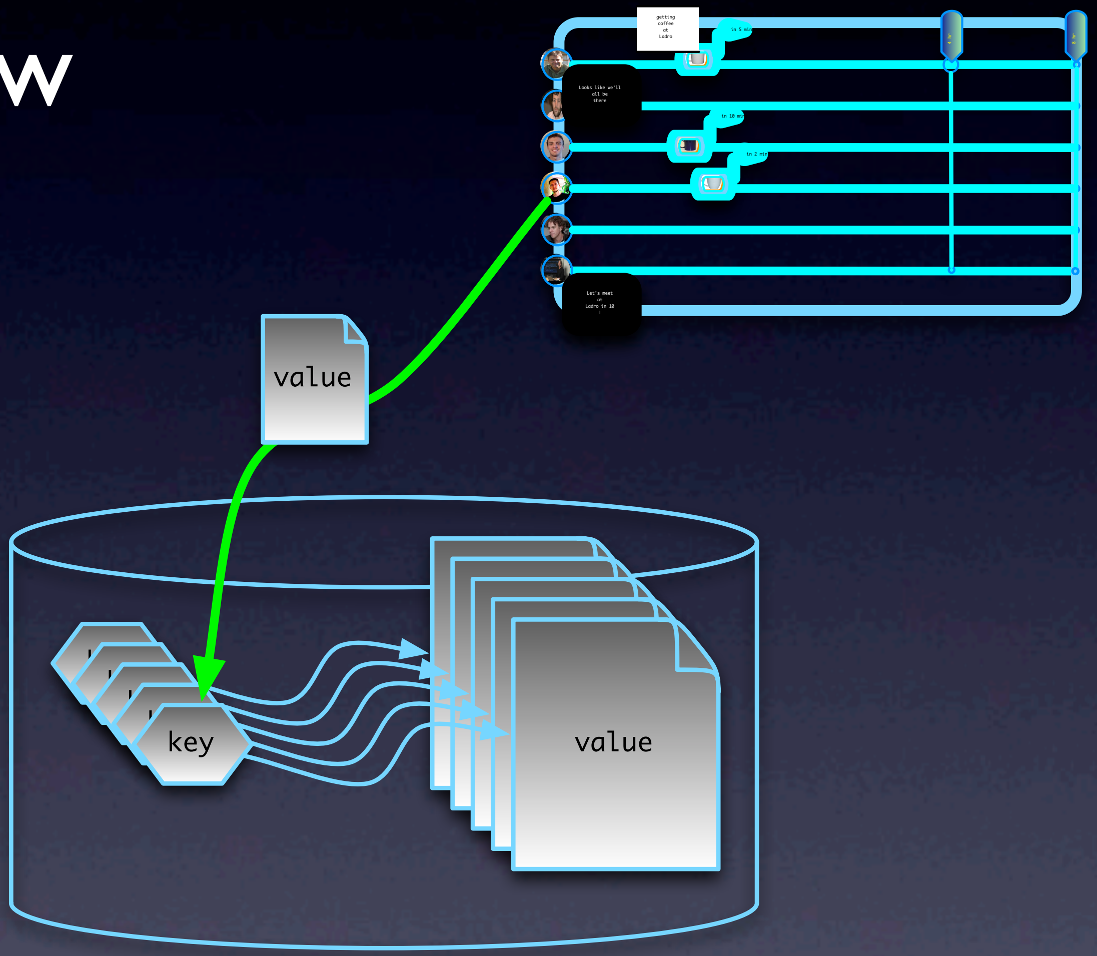


```
for(
 e <- collection.<verb>( key )
 if cond
) {
  handle( e )
}
```

A few verbs

And we'll have production follow messaging style

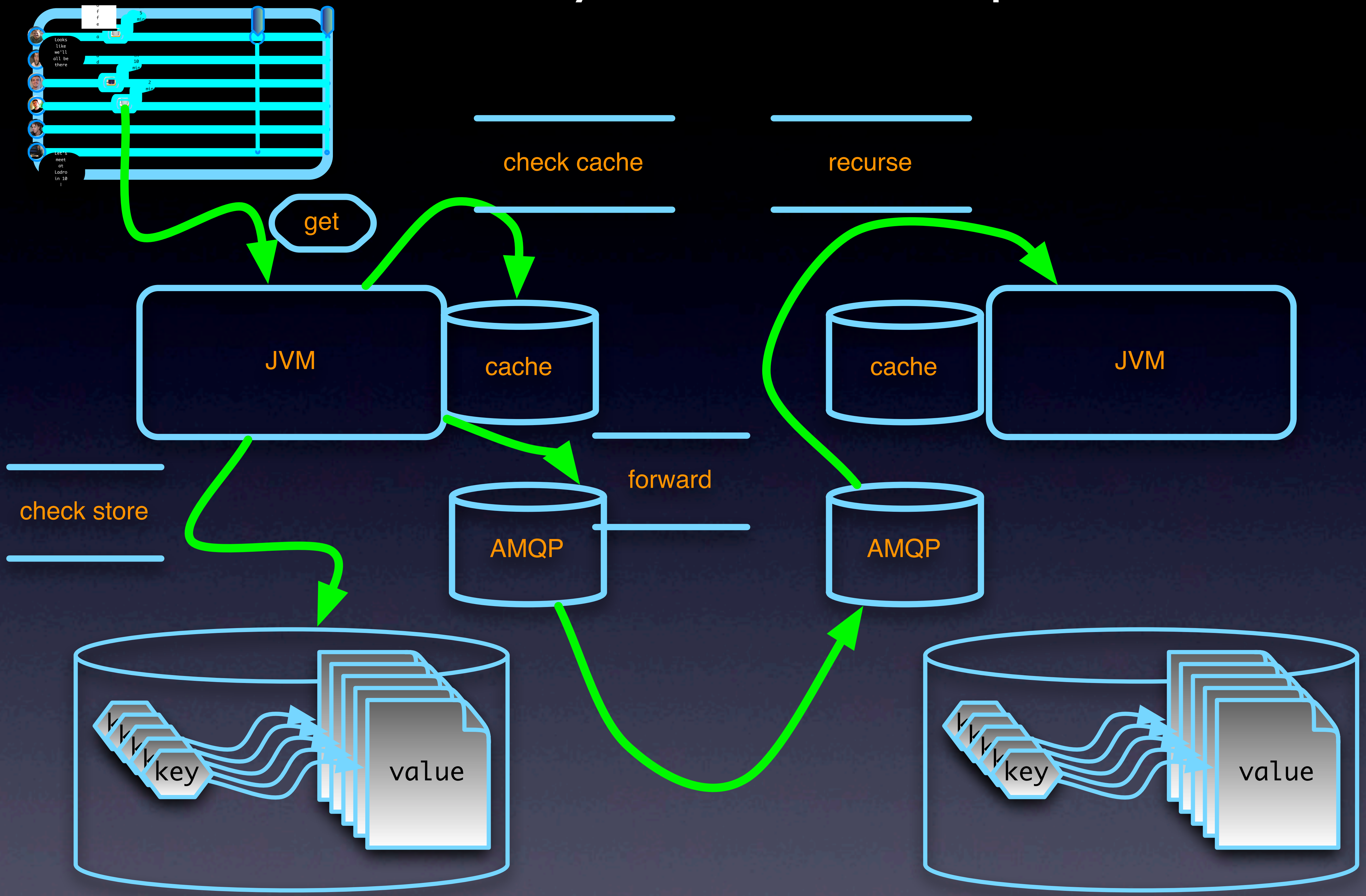
```
collection.<verb>( key, value )
```



A few verbs

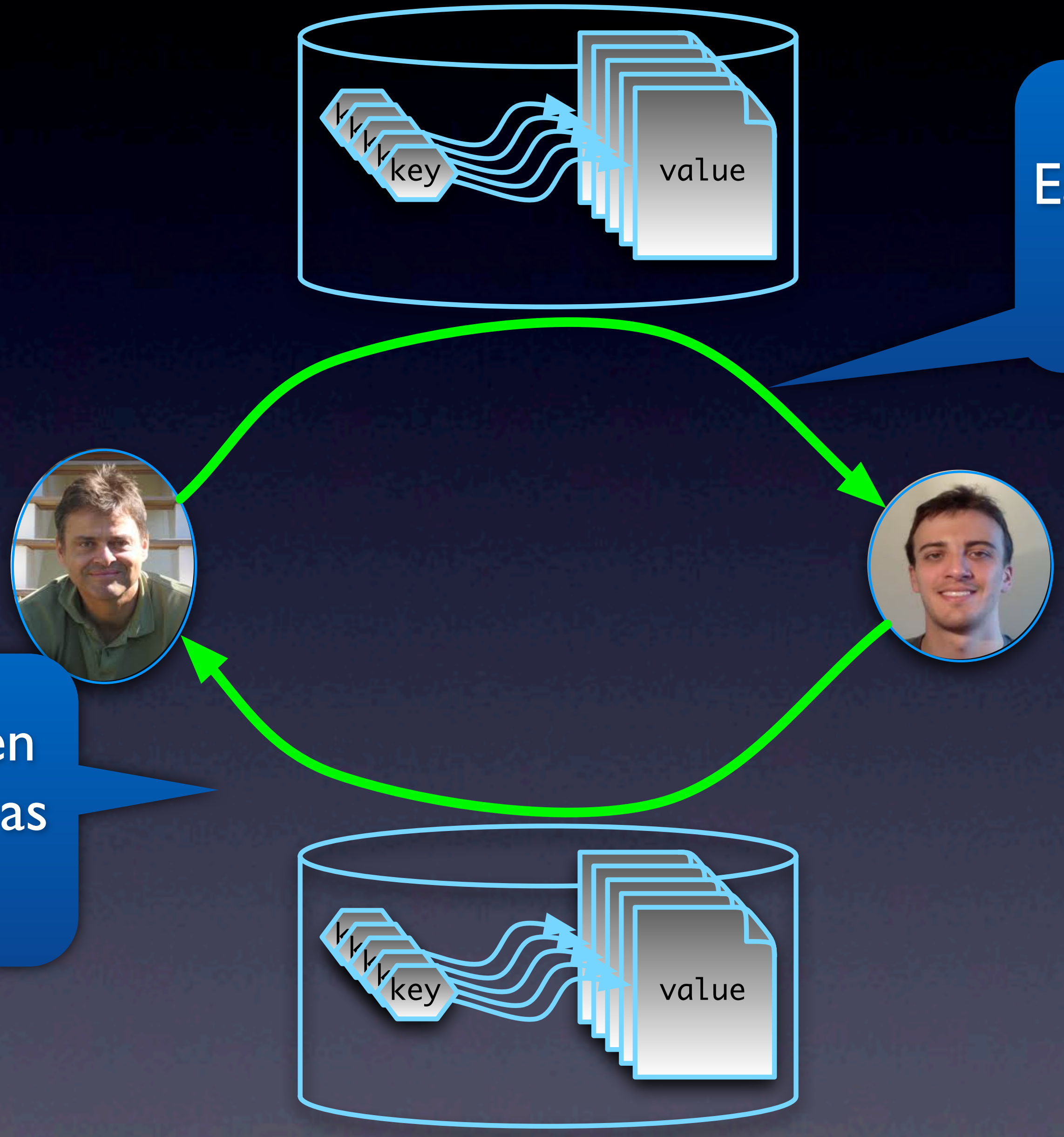
If we squint a little we can just make out the π-calculus.

ScalaBay October Meetup



L.G. Meredith, Managing Partner, Biosimilarity LLC

How does this relate to agents in the Internet?

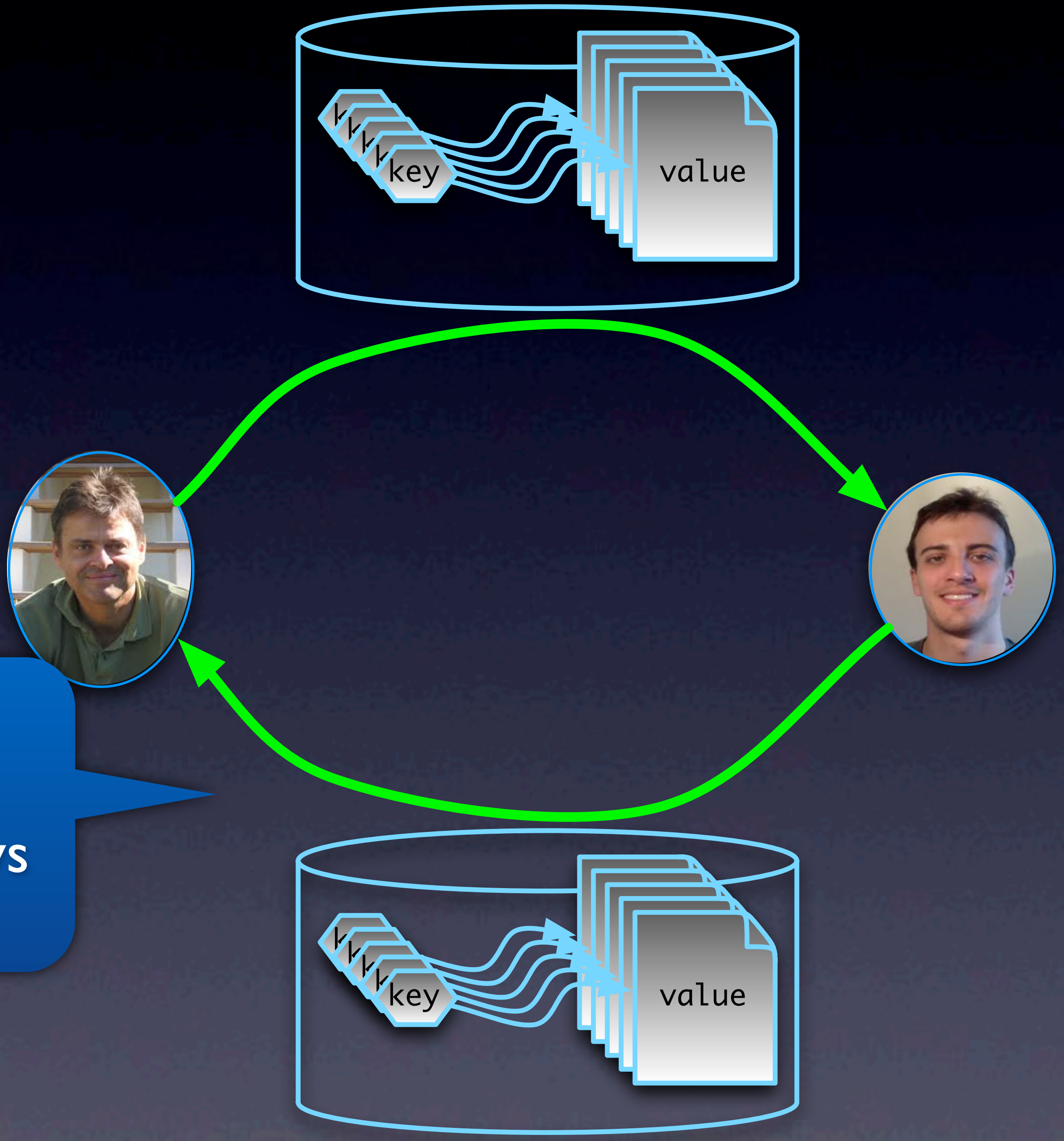


Each channel has its own store

A connection between two agents is treated as a pair of channels

ScalaBay October Meetup

How does this relate to agents in the Internet?



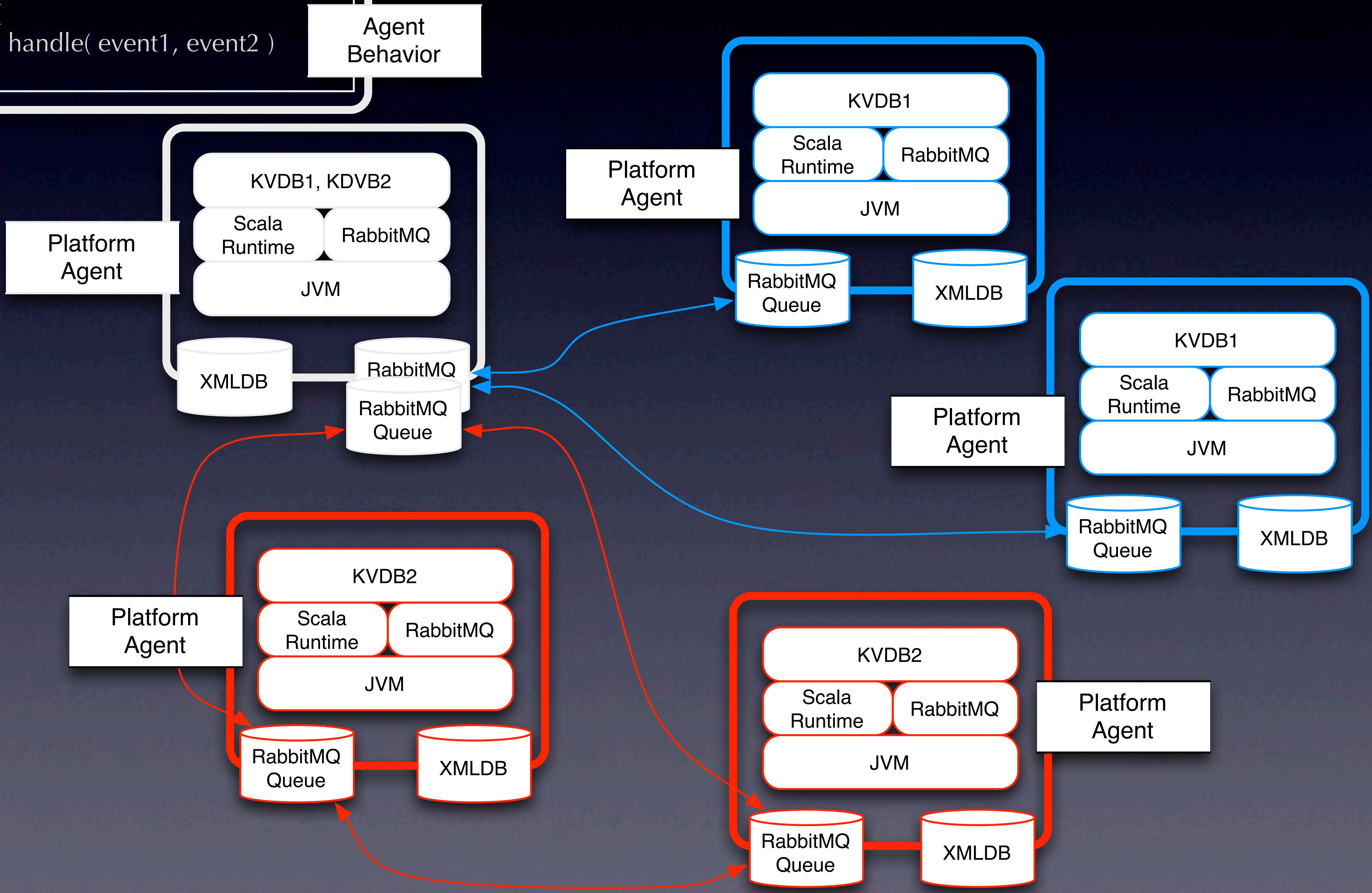
Channels are subpartitioned by keys

L.G. Meredith, Managing Partner, Biosimilarity LLC

Putting it all together

```
for(
  event1 <- kvdb1.get( pattern1 );
  event2 <- kvdb2.get( pattern2 );
  if ( cond( event1, event2 ) )
) {
   handle( event1, event2 )
}
```

A generic process agent is an instance of an agent behavior. These run on platform agents. An agent in the sense Fred defines is realized as an instance of a specific agent behavior that implements one or more of agent services protocols including introduction, invitation, verification, etc.



What is SpecialK?

Example only!
Not actual
implementation

Agent services agent

Process agent

KVDB verbs

Agent properties

Agent addressing

```
for(
  introReq <- kvdb1.get( introduce( reqAgentId, trgtAgentId ) )
) {
    kvdb1.put( acceptIntro?( reqAgentId, trgtAgentId ) )
    for( introRsp <- kvdb.get( introRsp( trgtAgentId, reqAgentId ) ) {
      introRsp match {
          case Confirm( sharingLevel ) => ...
          case Deny( reason ) => ...
      }
    }
}
```

Introduction
Behavior

```
for(
  event1 <- kvdb1.get( pattern1 );
  event2 <- kvdb2.get( pattern2 );
  if ( cond( event1, event2 ) )
) {
    handle( event1, event2 )
}
```

Agent
Behavior

```
@sequence(
    @act( verb, patternSpec )
    @separate( P1, P2 )
)
```

Property
Specification

Scala annotation
syntax not finalized!

get, put, subscribe, publish, fetch, store, ...
over patterns described by

```
pattern ::= groundValue
          | variable
          | symbol( pattern* )
```

KDVB
capability

A few verbs in action

Here’s a version of the introduction protocol for more realz and less lolz

Backup slides

Compositionality

In computing scaling sometimes refers to lots of copies the same thing

Lots of chips, lots of boxes, lots of databases, lots of servers, lots of data centers

ScalaBay October Meetup

Compositionality

Compositionality turns that upside down. It says that scaling is about making the large reflect the small.

L.G. Meredith, Managing Partner, Biosimilarity LLC

ScalaBay October Meetup

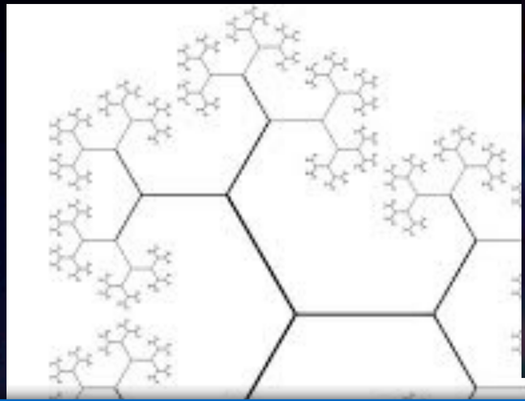
Compositionality

This is how nature scales.

L.G. Meredith, Managing Partner, Biosimilarity LLC

ScalaBay October Meetup

Compositionality

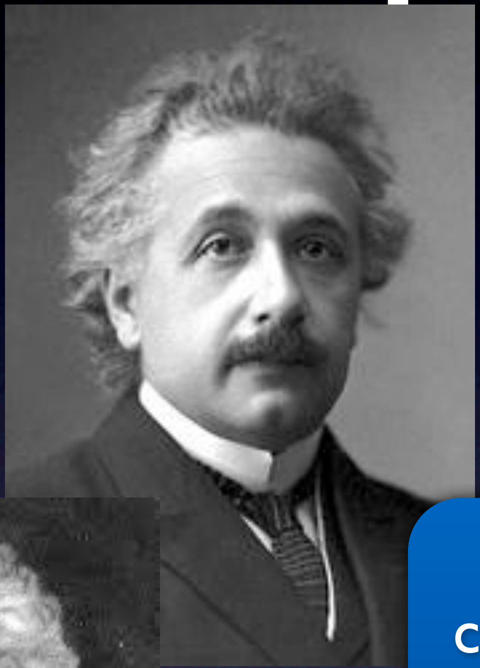


But, it's actually a fairly new thing that humans have begun to structuring their thinking and doing this way

L.G. Meredith, Managing Partner, Biosimilarity LLC

ScalaBay October Meetup

Compositionality



Here are a few people whose computationally effective ideas were not compositional

As a result, the physics of the small doesn't scale to the physics of the large (or vice versa)

L.G. Meredith, Managing Partner, Biosimilarity LLC

ScalaBay October Meetup

Compositionality

Computing is naturally organized compositionally

L.G. Meredith, Managing Partner, Biosimilarity LLC

Compositionality

Data structures are organized compositionally

- Lists
- Trees
- ...

Compositionality

The syntax of programming languages are organized compositionally

```
P ::= ...
   val identifier [: typeDecl] = valueExpr ; P
```

Compositionality

The fundamental organization of the mathematics of computation is compositional

- λ-calculus
  - M, N ::= x | λ x. M | M N
- π-calculus
  - P, Q ::= 0 | x?(y)P | x!(v) | P|Q | (new x)P | !P
- Category Theory

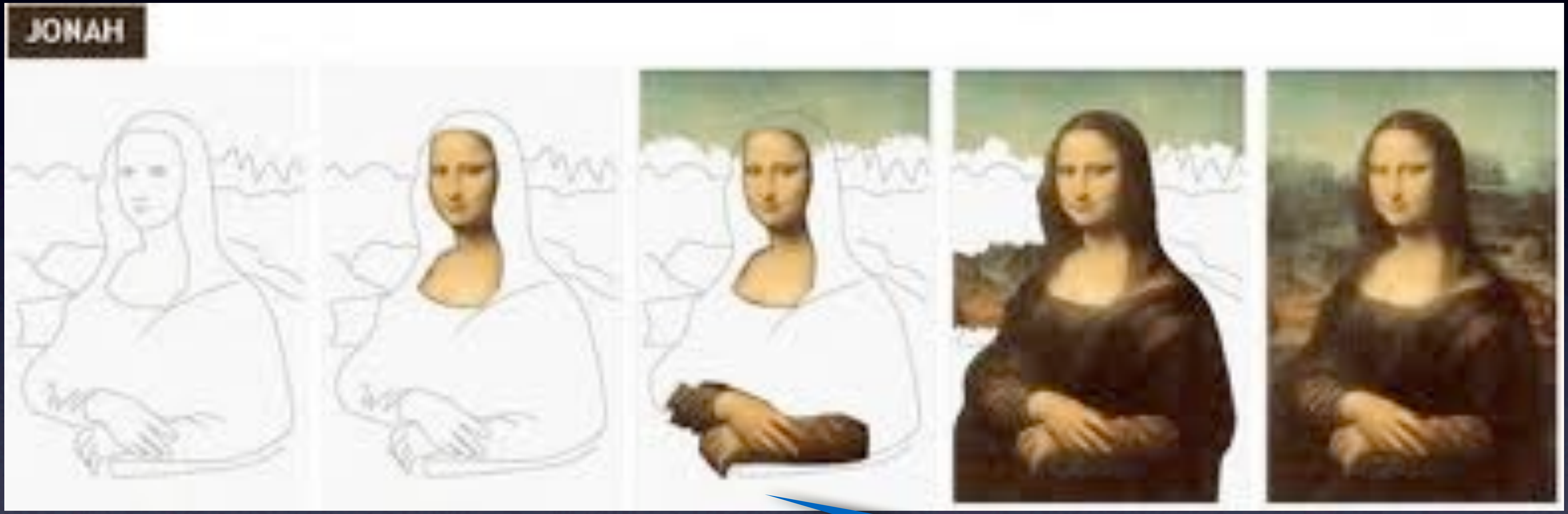
ScalaBay October Meetup

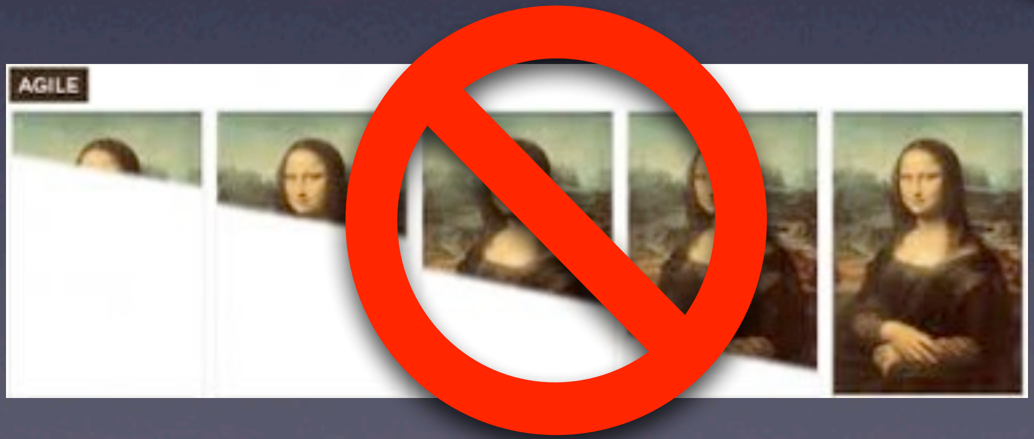
Agility

Agility is not just about being quick, light and responsive

L.G. Meredith, Managing Partner, Biosimilarity LLC

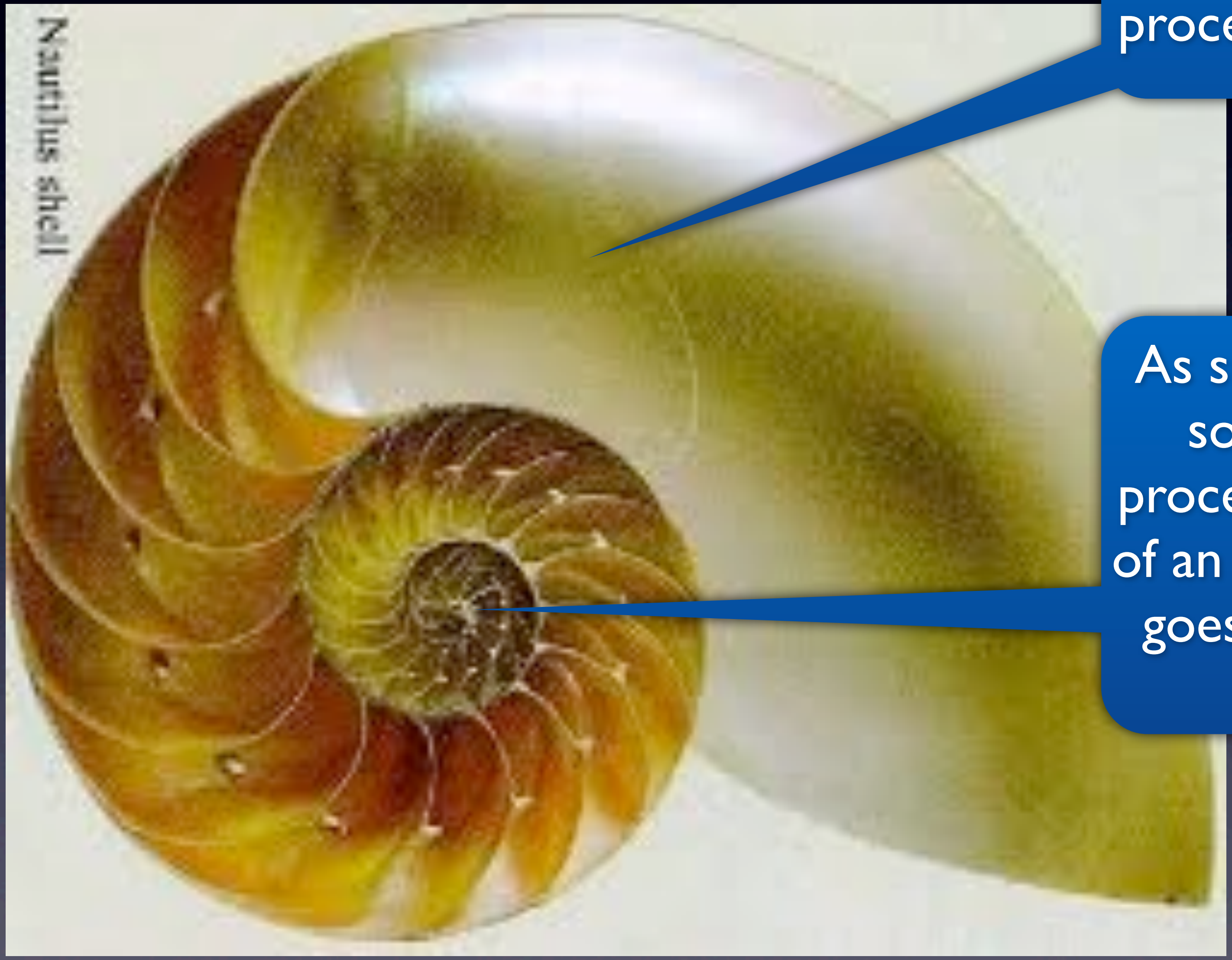
Agility





It’s primarily about being iterative -- moving from one expression of the whole to the next

Agility



This is really applying compositionality to scaling processes that occur in time

As such, thinking about the software development process as finding a fix-point of an iterated transformation goes at least as far back as the 70's

Agility

This also lines up with the organization of the maths underlying computation

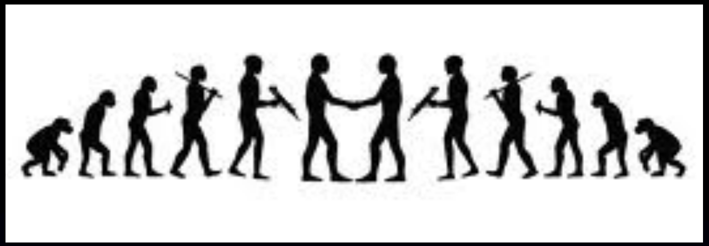
$$\mathbf{Y} = \lambda f.(\lambda x.f\ (x\ x))\ (\lambda x.f\ (x\ x))$$

Agility

```
def Y[A,B](f: (A=>B)=>(A=>B)) = {
  case class W(wf: W=>A=>B) {
    def apply(w: W) = wf(w)
  }
  val g: W=>A=>B = w => f(w(w))(_)
  g(W(g))
}
```

Agility

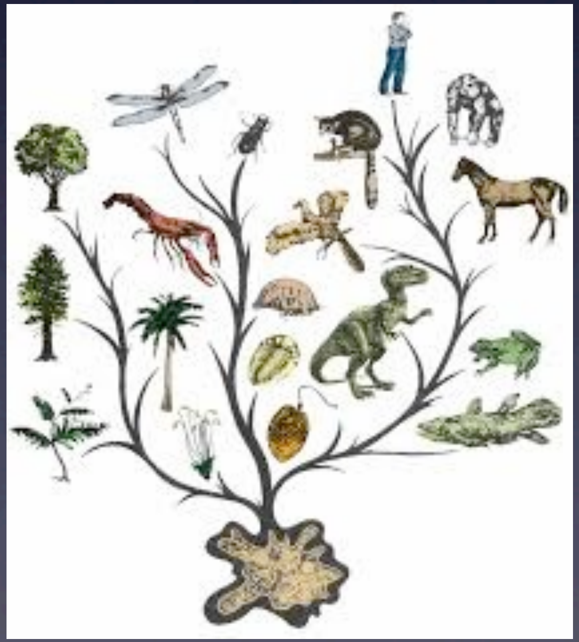
```
   val fac = Y[Int, Int](f => i => if (i <= 0) 1 else f(i - 1) *
   i)
fac(6)  //> res0: Int = 720

val fib = Y[Int, Int](f => i => if (i < 2) i else f(i - 1) + f(i
- 2))
fib(6)  //> res1: Int = 8
```



Agility

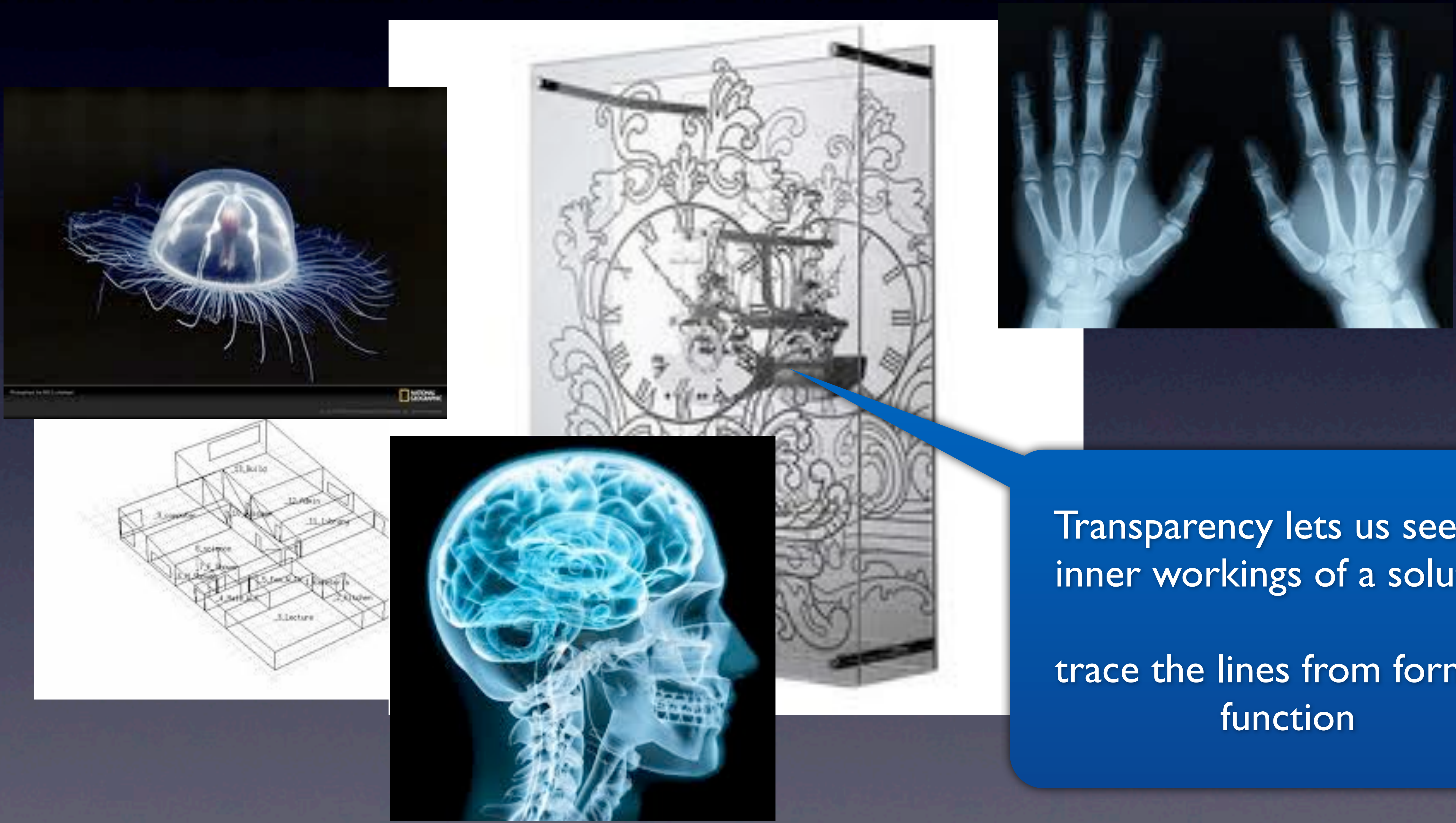
Iteratively moving from a whole expression of a solution to a more refined version of a whole expression of a solution:

- makes sense mathematically
- makes sense organizationally
- is what happens in nature



ScalaBay October Meetup

Transparency



Transparency lets us see the inner workings of a solution

trace the lines from form to function

L.G. Meredith, Managing Partner, Biosimilarity LLC

Transparency

In computing transparency is a form of compositionality -- a semantics for a model of computation is compositional if we follow the form of a program to determine its semantics, i.e. what it is and/or what it does

ScalaBay October Meetup

Transparency

In software development transparency is the ability to trace from requirements all the way down to lines of code

L.G. Meredith, Managing Partner, Biosimilarity LLC

Transparency

In organizations transparency is the ability for any one member to trace information and communication all the way through the chain or collection of people in the organization who come together to achieve a goal

Transparency

In each case there is an inner structure that supports and guides the process

Making that inner structure explicit and visible is a vital part of that support

ScalaBay October Meetup

Accountability

We stand by the results of our work

We are clear about the conditions under which people may rely on the results of our work

L.G. Meredith, Managing Partner, Biosimilarity LLC

Accountability

In computing accountability is closely connected to proof. We stand by a claim about how computation works by being able to produce, on demand, a proof of that claim. Proofs only work when the necessary hypothesis explicit and clear.

Accountability

In software development -- like science -- accountability is closely connected to testing. We stand by a claim about what our software can and cannot do by producing the tests that demonstrate the software’s behavior. Tests results are only reproduceable if we are clear about the test conditions.

Accountability

In organizations accountability is closely connected to responsibility. Standing by the results of our work doesn't mean we don't make mistakes. It means we own them and are willing to use them as a means of improving our offering (see agility).

Summary

Fostering

compositionality

agility

transparency

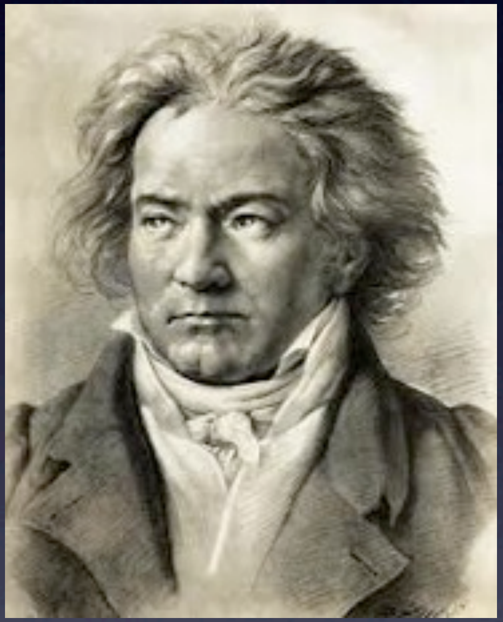
accountability

is like following a hidden score

ScalaBay October Meetup

Summary

Code and systems scale



because they are

composed of

reconfigurable

components

the behavior of which we stand by

and support

L.G. Meredith, Managing Partner, Biosimilarity LLC

ScalaBay October Meetup

Summary

Teams scale

because they are

composed of

agile and

responsible

players supported by the

inner working of their organization

L.G. Meredith, Managing Partner, Biosimilarity LLC

Summary

In short, these are the
building blocks
of self-organization
that recognize
selves are made of selves are made
of ... that become coherent when
they are both supported and relied on

ScalaBay October Meetup

That’s how we do it if how we get there is where we are going

L.G. Meredith, Managing Partner, Biosimilarity LLC