AmbientTalk

Modern Actors for Modern Networks

Tom Van Cutsem

Elisa Gonzalez Boix Andoni Lombide Carreton Christophe Scholliers

Kevin Pinte Dries Harnie Wolfgang De Meuter

Software Languages Lab

Brussels, Belgium

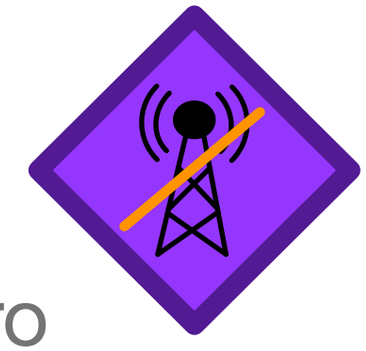
Vrije Universiteit Brussel

Mobile Ad Hoc Networks

Networks of **mobile** devices that use **wireless** p2p communication

Mobile Ad Hoc Networks

Networks of **mobile** devices that use **wireless** p2p communication

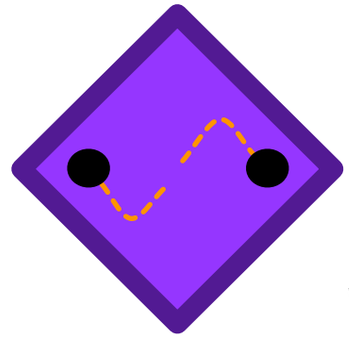


Zero Infrastructure

Mobile Ad Hoc Networks

Networks of **mobile** devices that use **wireless** p2p communication

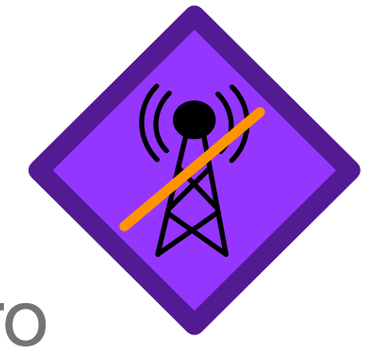
Zero Infrastructure



Volatile Connections

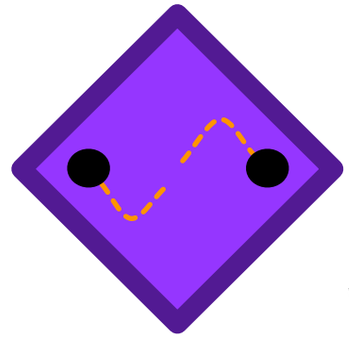
Mobile Ad Hoc Networks

Networks of **mobile** devices that use **wireless** p2p communication



Zero Infrastructure

Local interactions



Volatile Connections

Mobile P2P Apps

Mobile P2P Apps

Discovery

Mobile P2P Apps

Discovery Communication

Mobile P2P Apps

Discovery Communication Synchronisation

Mobile P2P Apps

Discovery Communication Synchronisation

Mobile P2P Apps

Discovery Communication Synchronisation

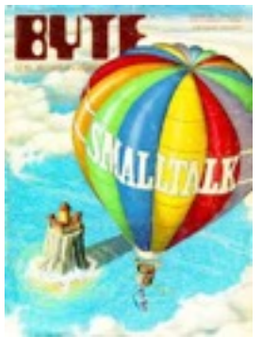
Mobile P2P Apps

Discovery Communication Synchronisation Failure handling

AmbientTalk: fact sheet

- Object-oriented, functional patterns, dynamically typed
- Actor-based concurrency/distribution
- Mirror-based reflection
- JVM as platform
- Runs on android

Four Decades of Language Research



Smalltalk
‘70s-‘80s

Self
1986

Scheme
1975

E
1997

AmbientTalk
2006

Hewitt’s actors
(PLASMA)
‘73-’76

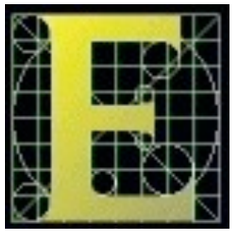
ABCL
1986

Four Decades of Language Research

Smalltalk
‘70s-‘80s

1986

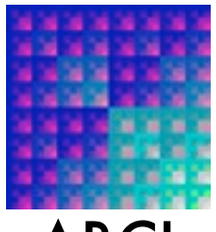
Scheme
1975



1997

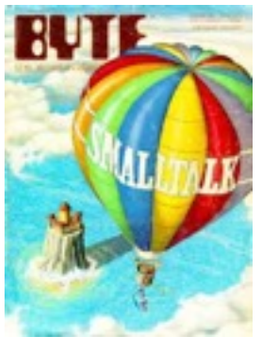
AmbientTalk
2006

Hewitt’s actors
(PLASMA)
‘73-’76



ABCL
1986

Four Decades of Language Research



Smalltalk
‘70s-‘80s

Self
1986

Scheme
1975

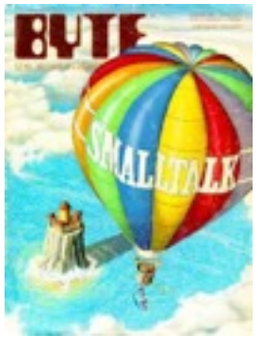
E
1997

AmbientTalk
2006

Hewitt’s actors
(PLASMA)
‘73-’76

ABCL
1986

Four Decades of Language Research



Smalltalk
‘70s-‘80s

Self
1986

Scheme
1975

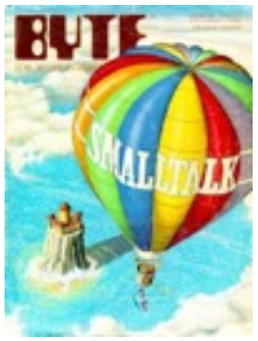
1997

AmbientTalk
2006

Hewitt’s actors
(PLASMA)
‘73-’76

ABCL
1986

Four Decades of Language Research



Smalltalk
‘70s-‘80s

Self
1986

Scheme
1975

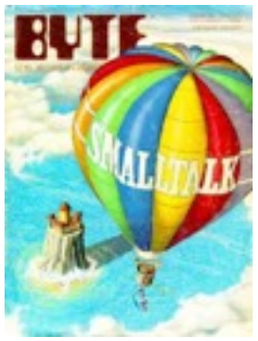
E
1997

AmbientTalk
2006

Hewitt’s actors
(PLASMA)
‘73-’76

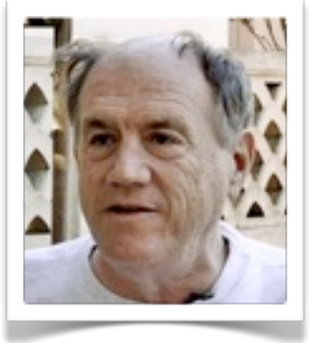
ABCL
1986

Four Decades of Language Research



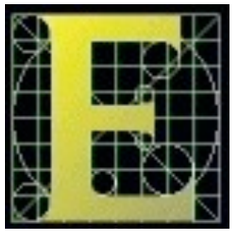
Smalltalk
‘70s-‘80s

Self
1986



Hewitt’s actors
(PLASMA)
‘73-’76

Scheme
1975

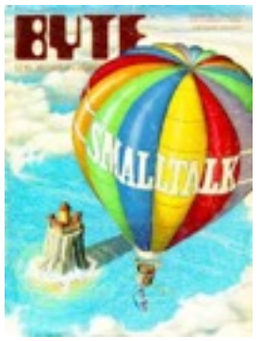


1997

ABCL
1986

AmbientTalk
2006

Four Decades of Language Research



Smalltalk
‘70s-‘80s

Self
1986

Scheme
1975

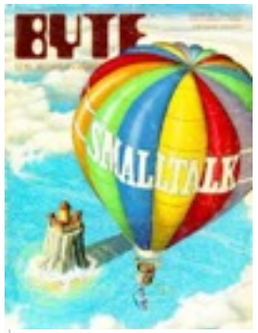
E
1997

AmbientTalk
2006

Hewitt’s actors
(PLASMA)
‘73-’76

ABCL
1986

Four Decades of Language Research



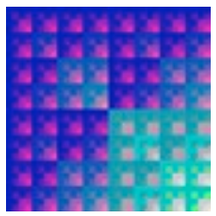
Self
1986

Scheme
1975

1997

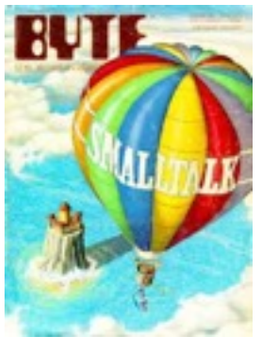
AmbientTalk
2006

Hewitt's actors
(PLASMA)
'73-'76



ABCL
1986

Four Decades of Language Research



Smalltalk
‘70s-‘80s

Self
1986

Scheme
1975

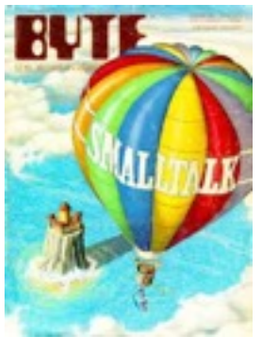
E
1997

AmbientTalk
2006

Hewitt’s actors
(PLASMA)
‘73-’76

ABCL
1986

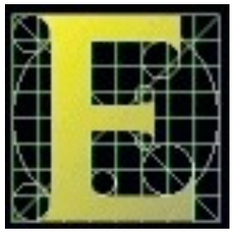
Four Decades of Language Research



Smalltalk
‘70s-‘80s

Self
1986

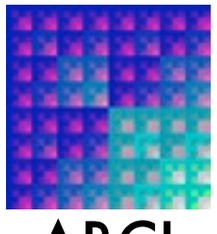
AmbientTalk
2006



1997

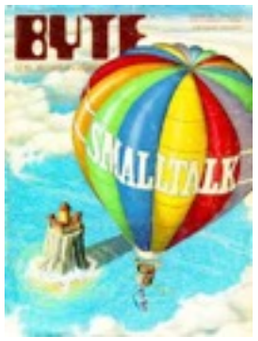
Scheme
1975

Hewitt’s actors
(PLASMA)
‘73-’76



ABCL
1986

Four Decades of Language Research



Smalltalk
‘70s-‘80s

Self
1986

Scheme
1975

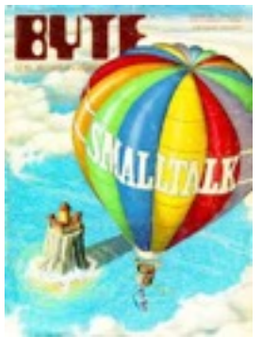
E
1997

AmbientTalk
2006

Hewitt’s actors
(PLASMA)
‘73-’76

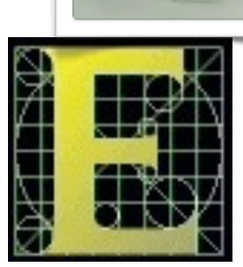
ABCL
1986

Four Decades of Language Research



Smalltalk
‘70s-‘80s

Self
1986

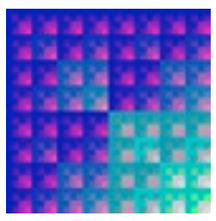


1997

AmbientTalk
2006

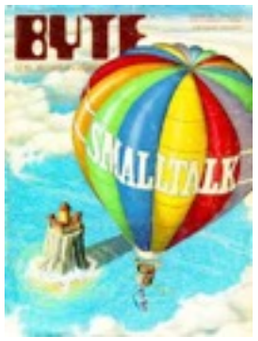
Scheme
1975

Hewitt’s actors
(PLASMA)
‘73-’76



ABCL
1986

Four Decades of Language Research



Smalltalk
‘70s-‘80s

Self
1986

Scheme
1975

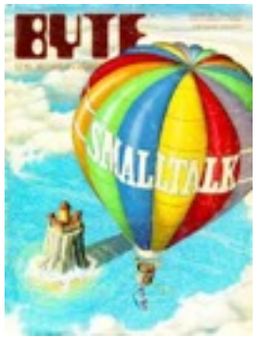
E
1997

AmbientTalk
2006

Hewitt’s actors
(PLASMA)
‘73-’76

ABCL
1986

Four Decades of Language Research



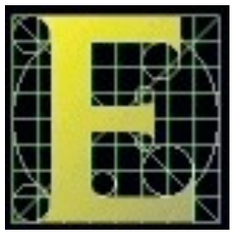
Smalltalk
‘70s-‘80s

Self
1986

ecma INTERNATIONAL

ECMAScript 6

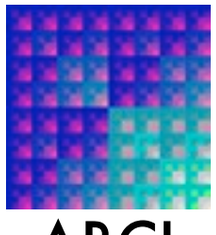
Scheme
1975



1997

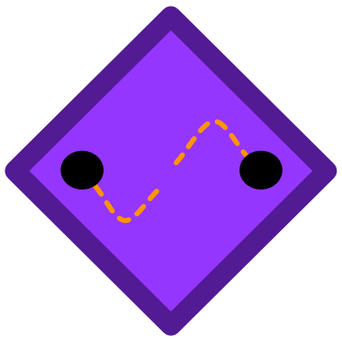
AmbientTalk
2006

Hewitt’s actors
(PLASMA)
‘73-’76



ABCL
1986

How does AmbientTalk help?



Volatile Connections

Asynchronous, buffered messaging
send messages, even when disconnected

No blocking synchronization
receive events, even when disconnected

Network failures ≠ exceptions
timeouts & leasing, whether connected or disconnected

Zero Infrastructure

Peer-to-peer service discovery protocol
decentralized, location-based

AmbientTalk Basics

Object-oriented


```
def makePoint(i, j) {
  object: {
    def x := i;
    def y := j;
    def distanceToOrigin() {
      ((self.x * self.x) + (self.y * self.y)).sqrt()
    }
  }
}
```

Object-oriented


```
def makePoint(i, j) {
  object: {
    def x := i;
    def y := j;
    def distanceToOrigin() {
      ((self.x * self.x) + (self.y * self.y)).sqrt()
    }
  }
}


def point := makePoint(1,1);
point.distanceToOrigin();
```

Blocks + keyworded message sends


```
def square := { |x| x * x }
square(5) // 25
```

```
[1,2,3].map: { |x| x * x } // [1,4,9]
[1,2,3].inject: 0 into: { |sum,x| sum + x } // 6
```

Control structures


```
if: (foo != nil) then: {
  foo.bar();
} else: {
  raise: Exception.new(“error”);
}
```

```
while: { x < 10 } do: {
  x := x + 1
}
```

```
0.to: 10 do: { |i|
  system.println(i);
}
```

Event handlers

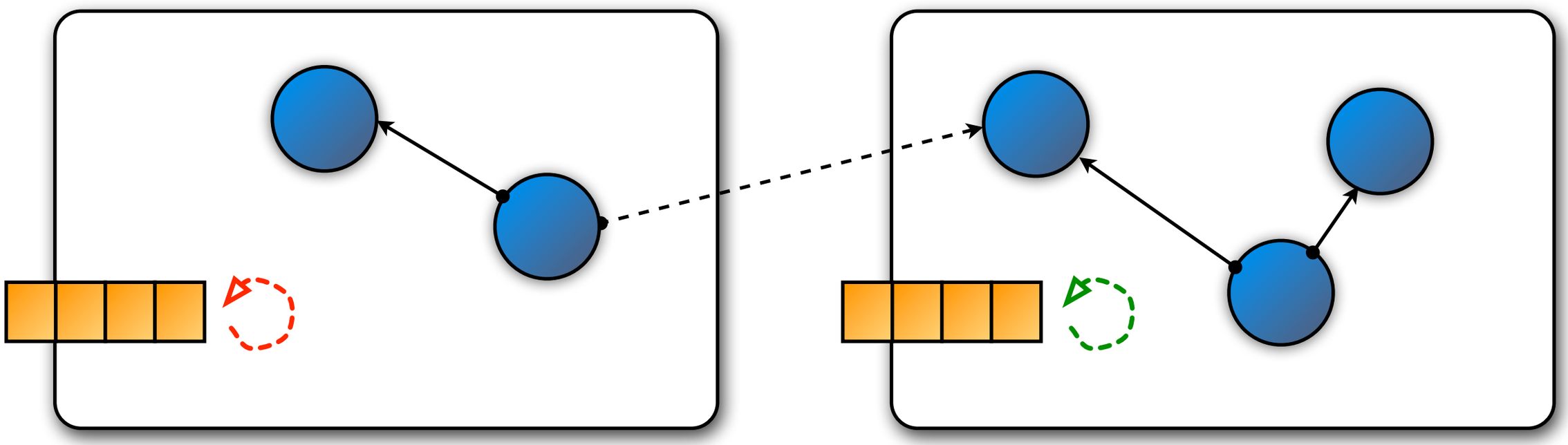

```
on: button.click do: {
  system.println(“clicked!”);
}
```

```
when: 10.seconds elapsed: {
  system.println(“time’s up!”);
}
```

Concurrency & Distribution

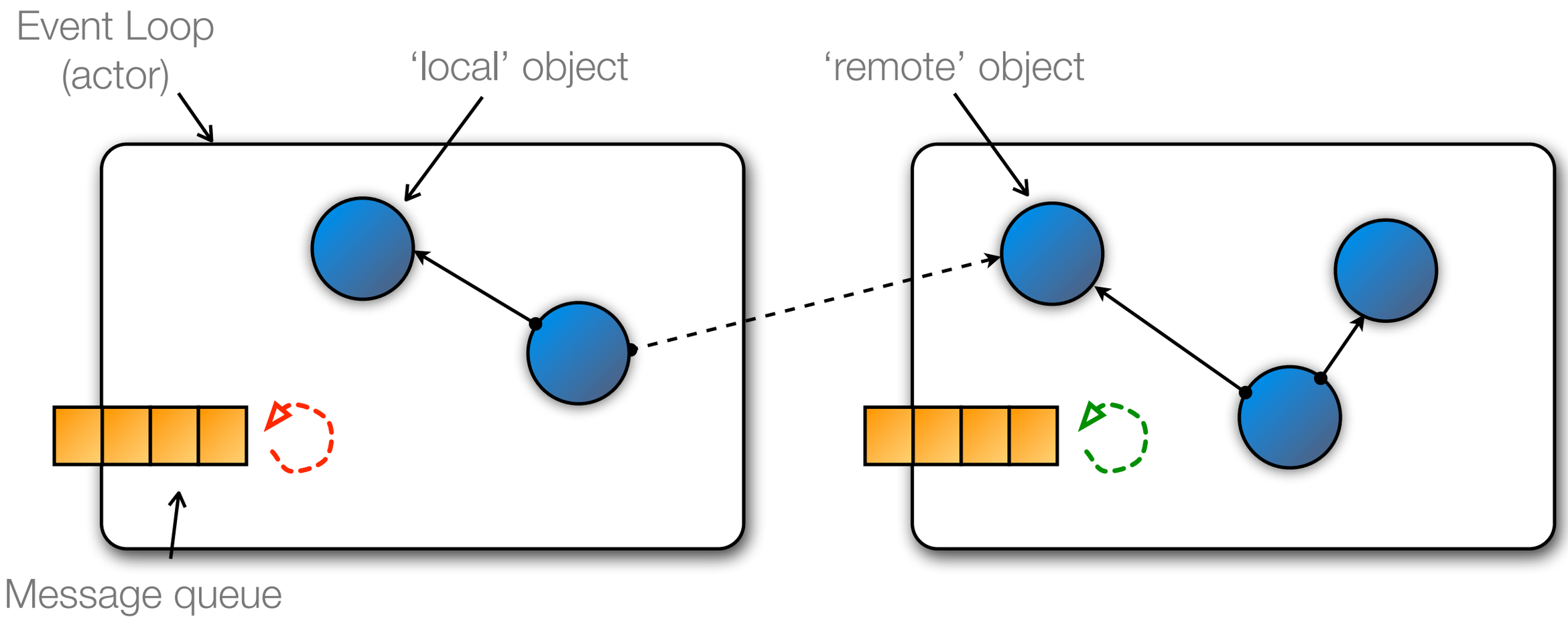
Event Loop Concurrency

- AmbientTalk programs are event loops
- They react to events from the outside world
- Inter-event loop communication is asynchronous

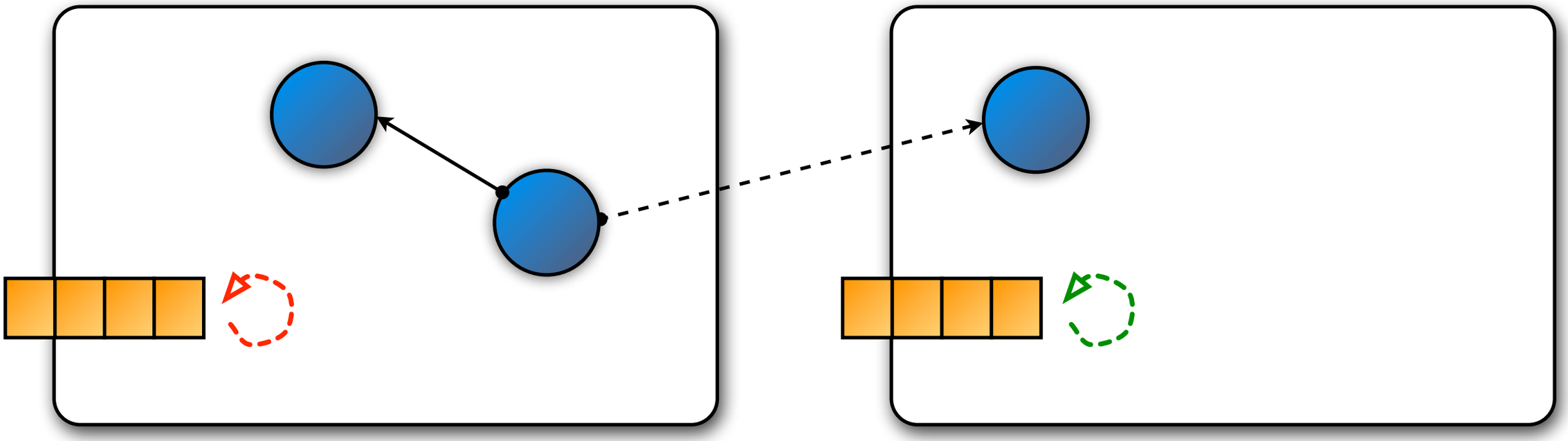


Event Loop Concurrency

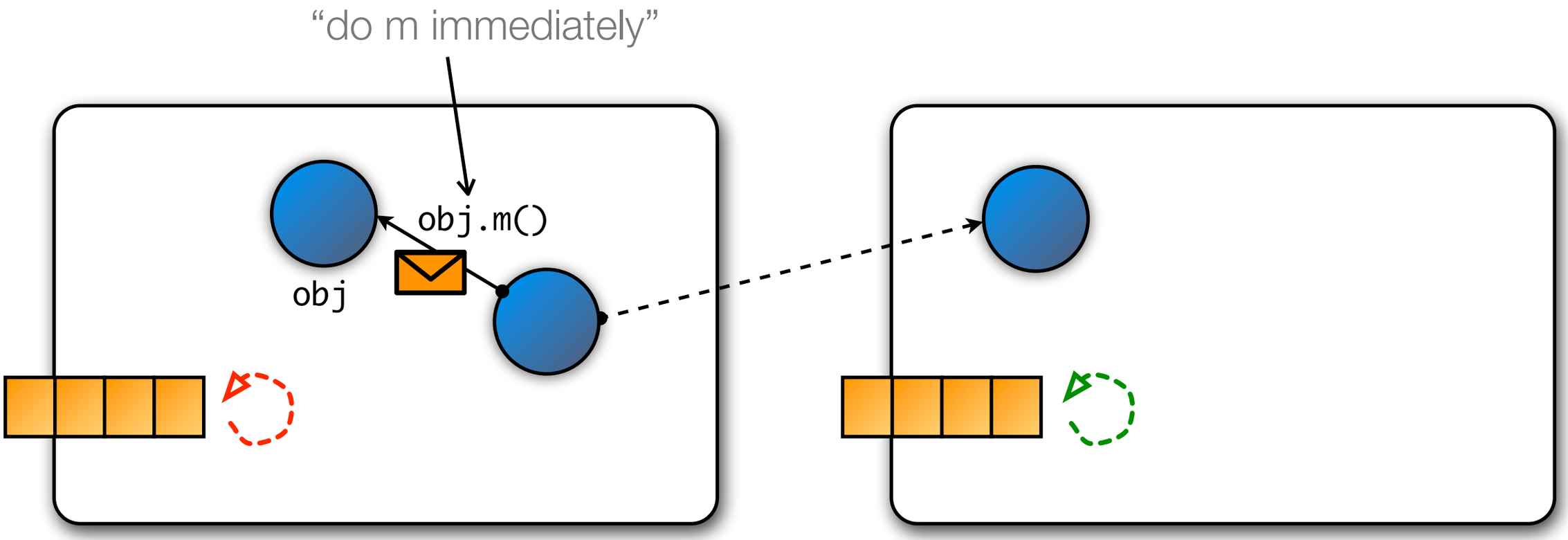
- AmbientTalk programs are event loops
- They react to events from the outside world
- Inter-event loop communication is asynchronous



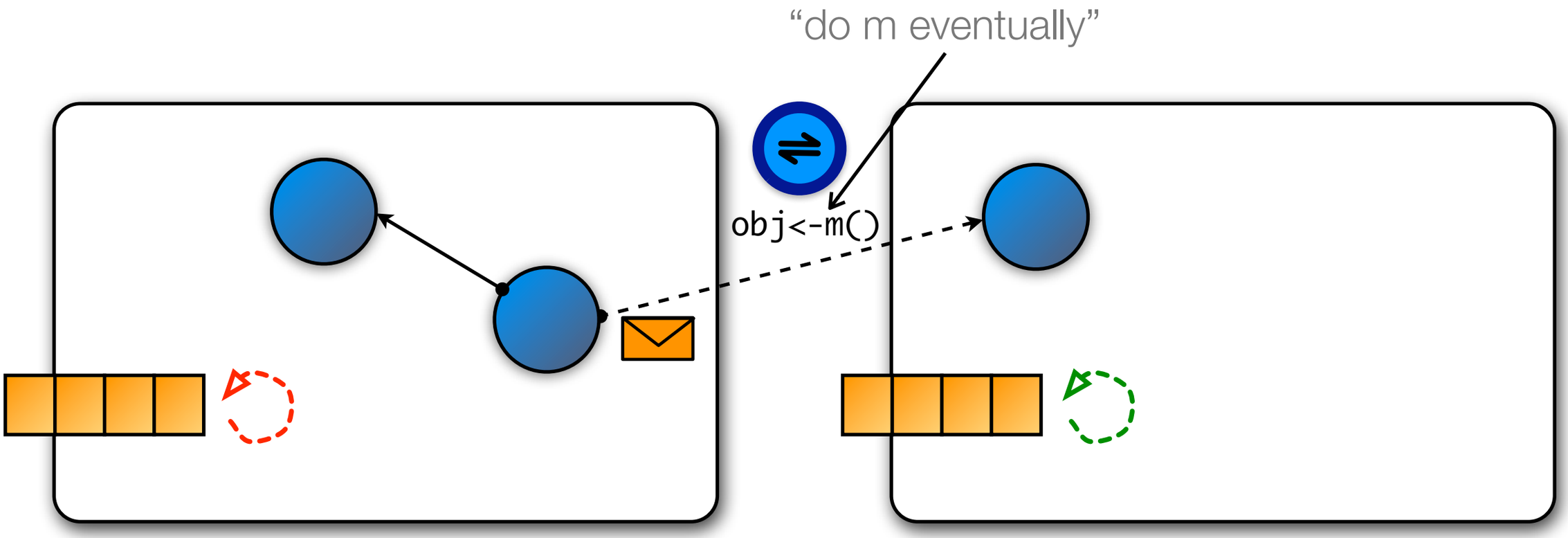
Event Loop Concurrency in AmbientTalk



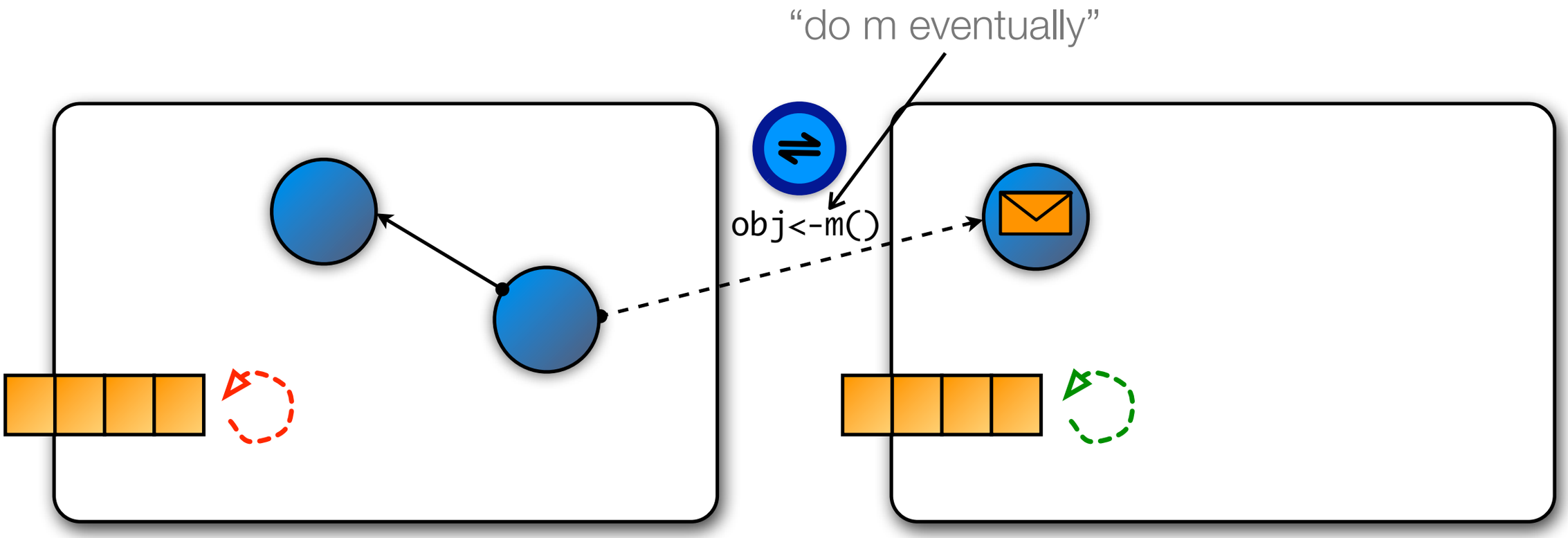
Event Loop Concurrency in AmbientTalk



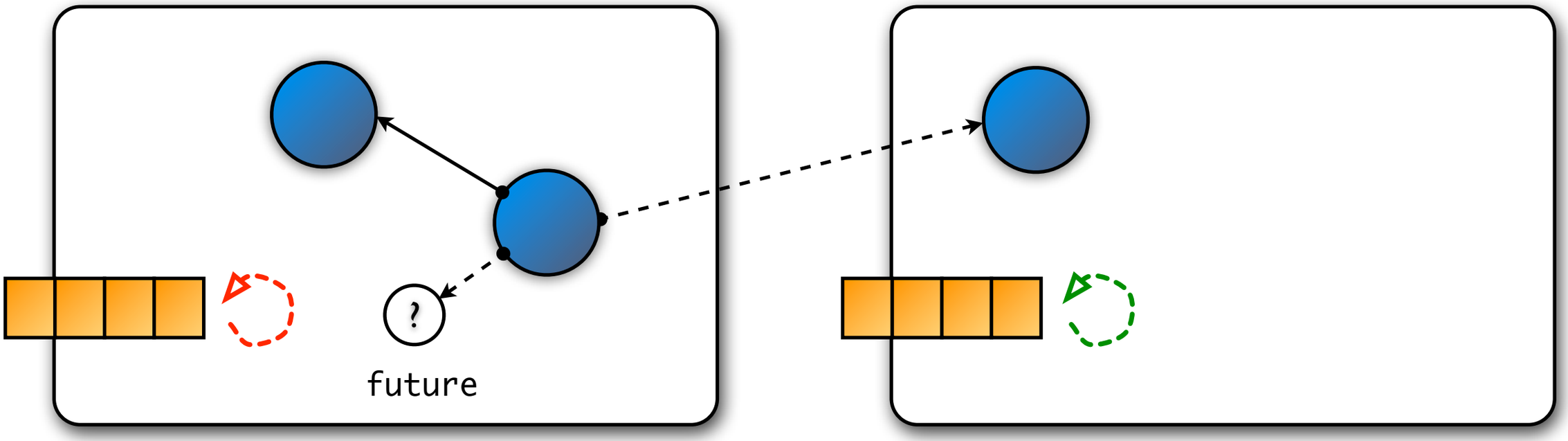
Event Loop Concurrency in AmbientTalk



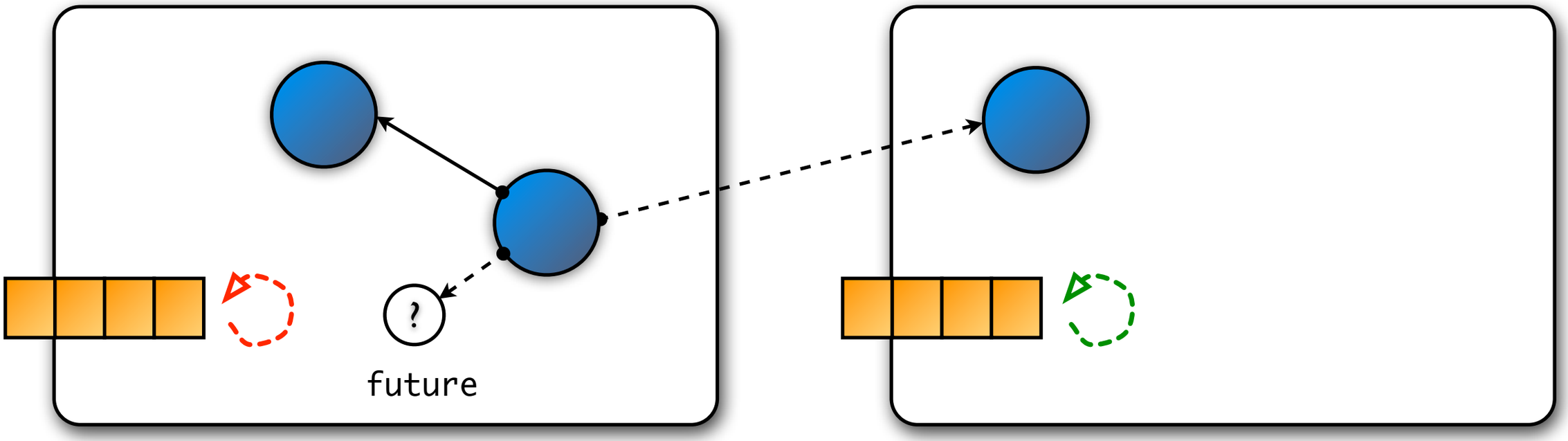
Event Loop Concurrency in AmbientTalk



Event Loop Concurrency in AmbientTalk

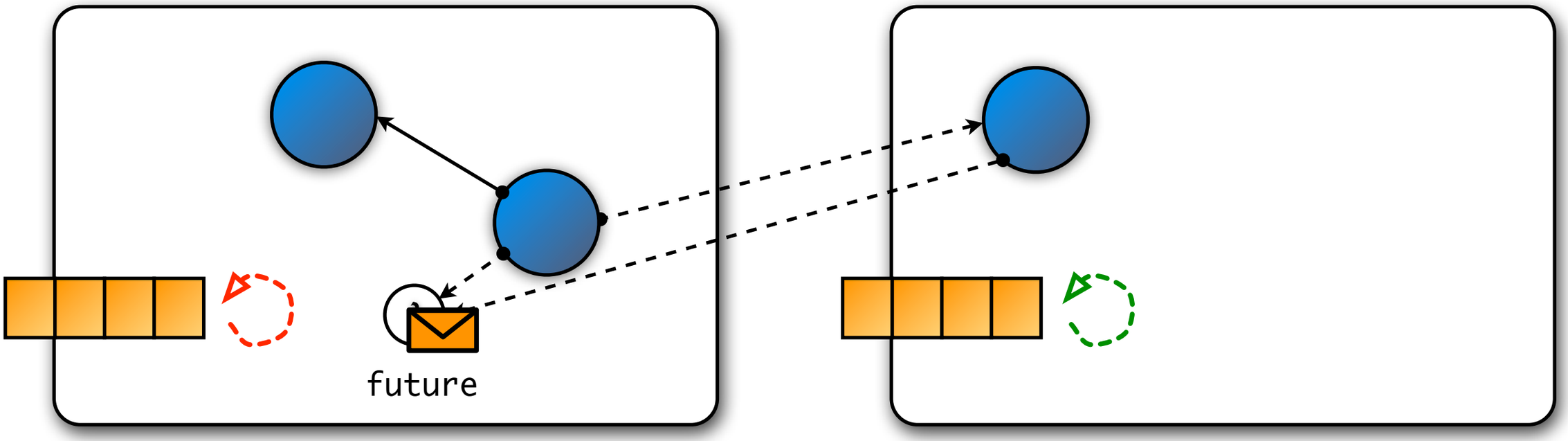


Event Loop Concurrency in AmbientTalk



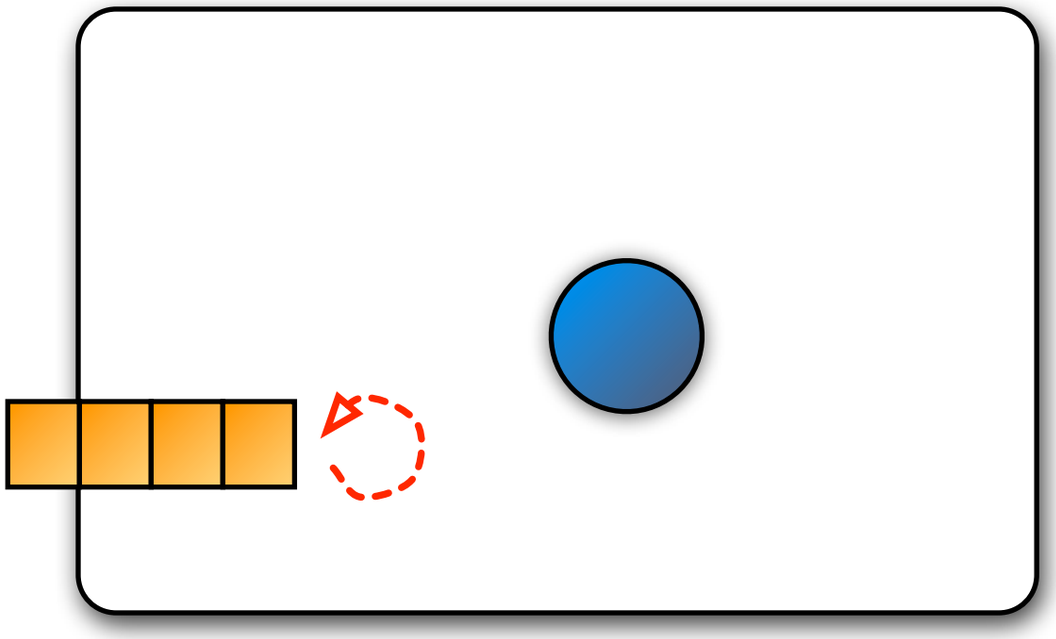
```
def future := obj<-m()@TwoWay
when: future becomes: { |value|
  // process reply
}
```

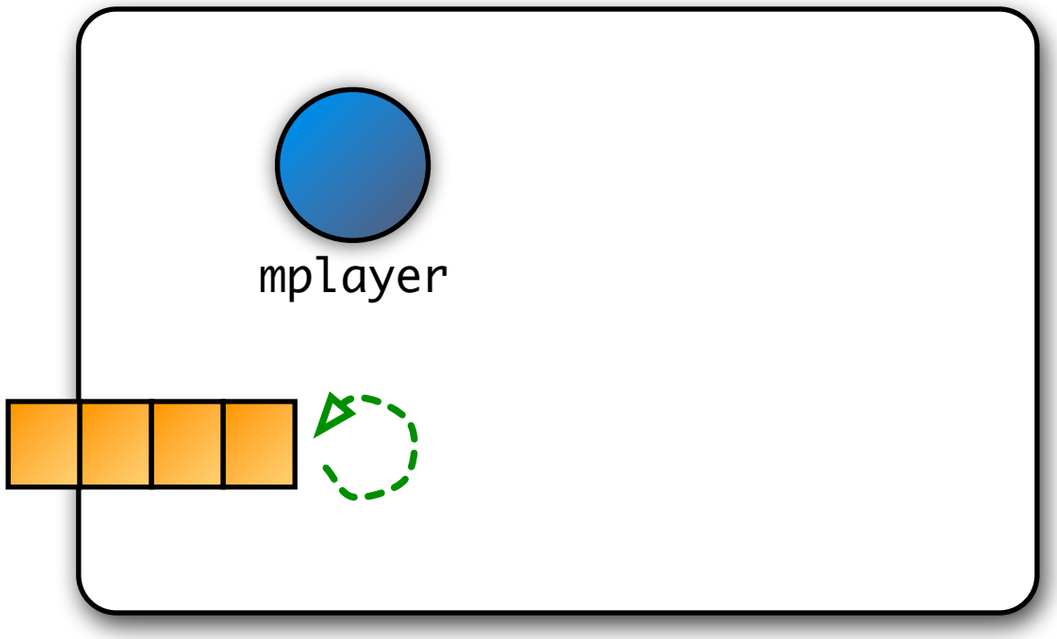
Event Loop Concurrency in AmbientTalk



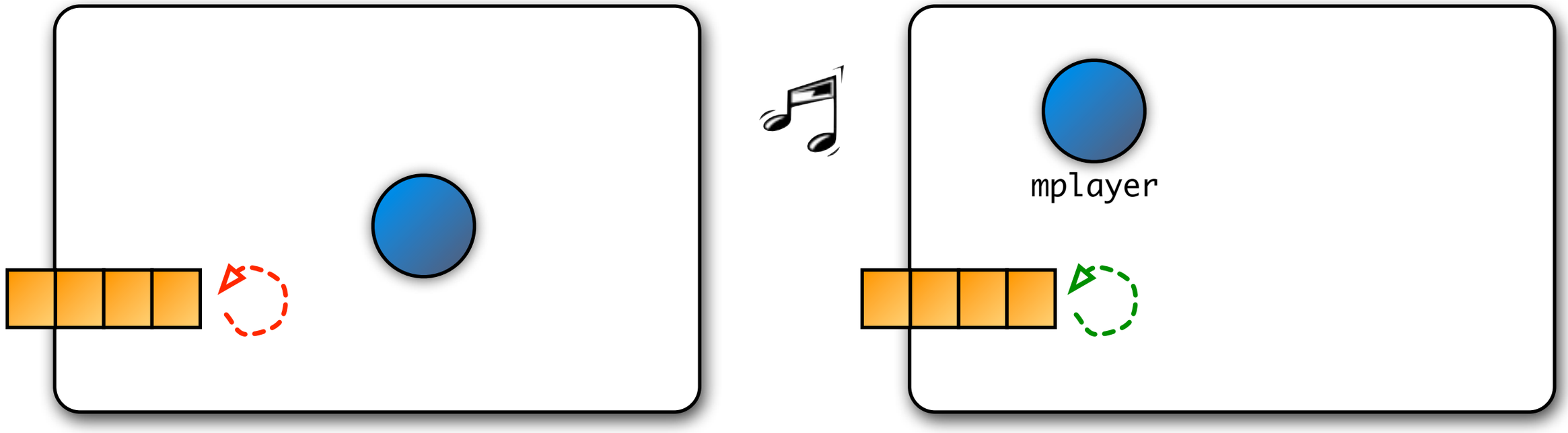

```
def future := obj<-m()@TwoWay
when: future becomes: { |value|
  // process reply
}
```

Exporting & discovering objects





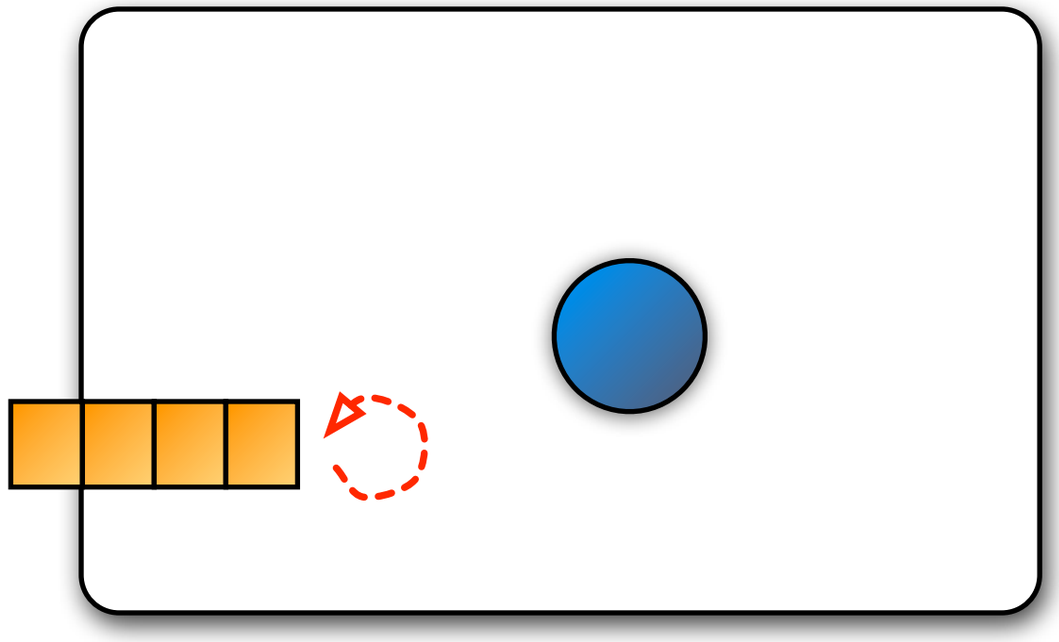
Exporting & discovering objects



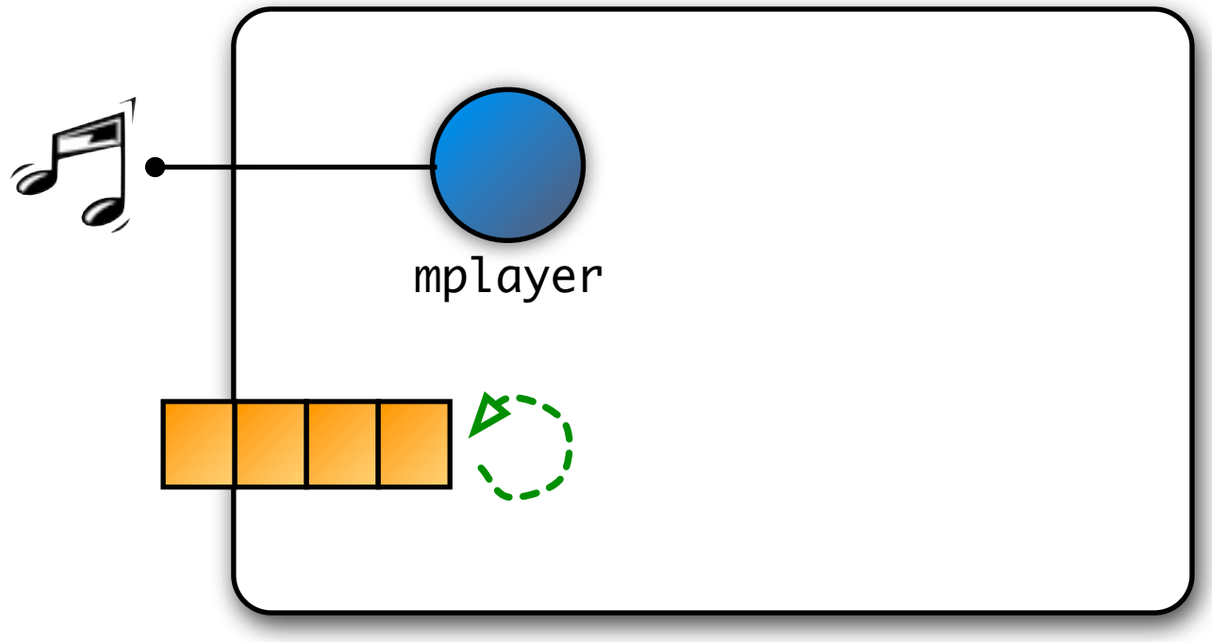
deftype MusicPlayer

deftype MusicPlayer

Exporting & discovering objects

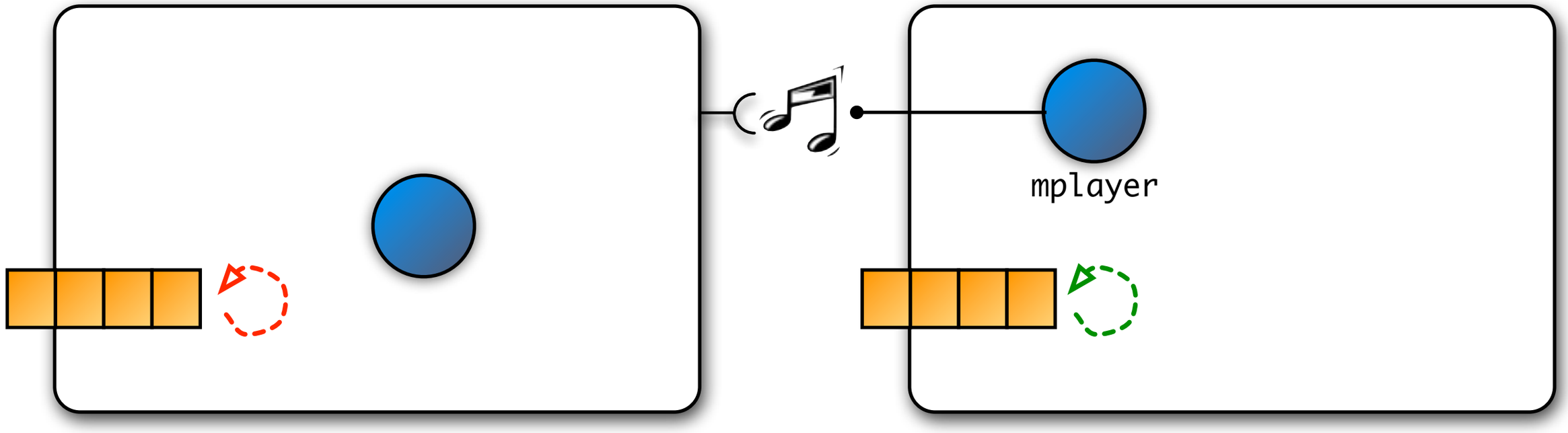


```
deftype MusicPlayer
```




```
deftype MusicPlayer

export: mplayer as: MusicPlayer
```

Exporting & discovering objects

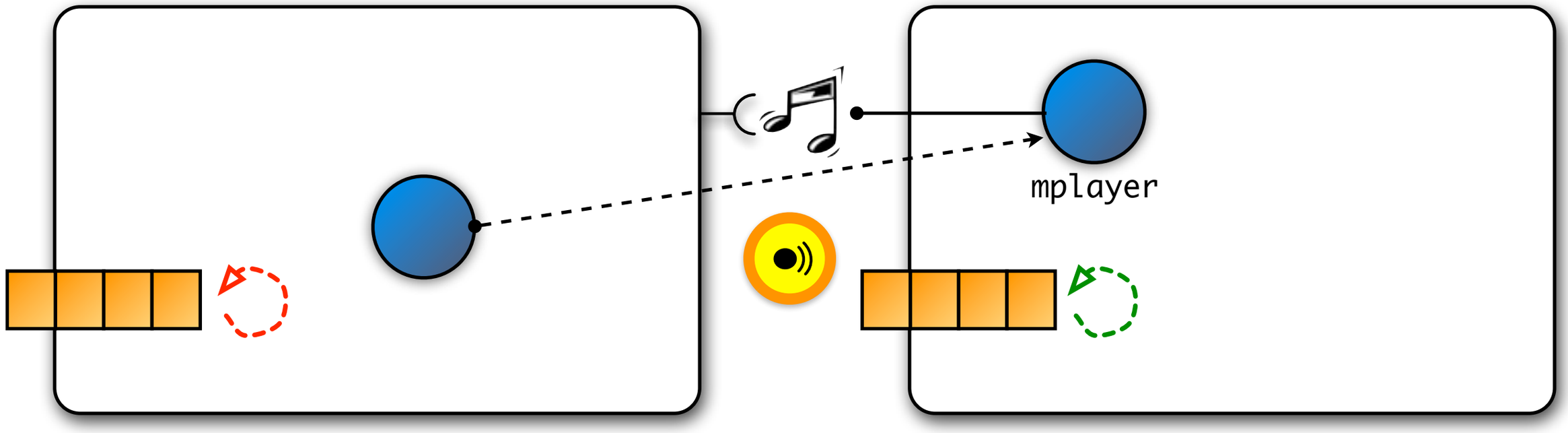



```
deftype MusicPlayer
```

```
deftype MusicPlayer

export: mplayer as: MusicPlayer
```

```
whenever: MusicPlayer discovered: { |mplayer|
  // open a session
}
```

Exporting & discovering objects

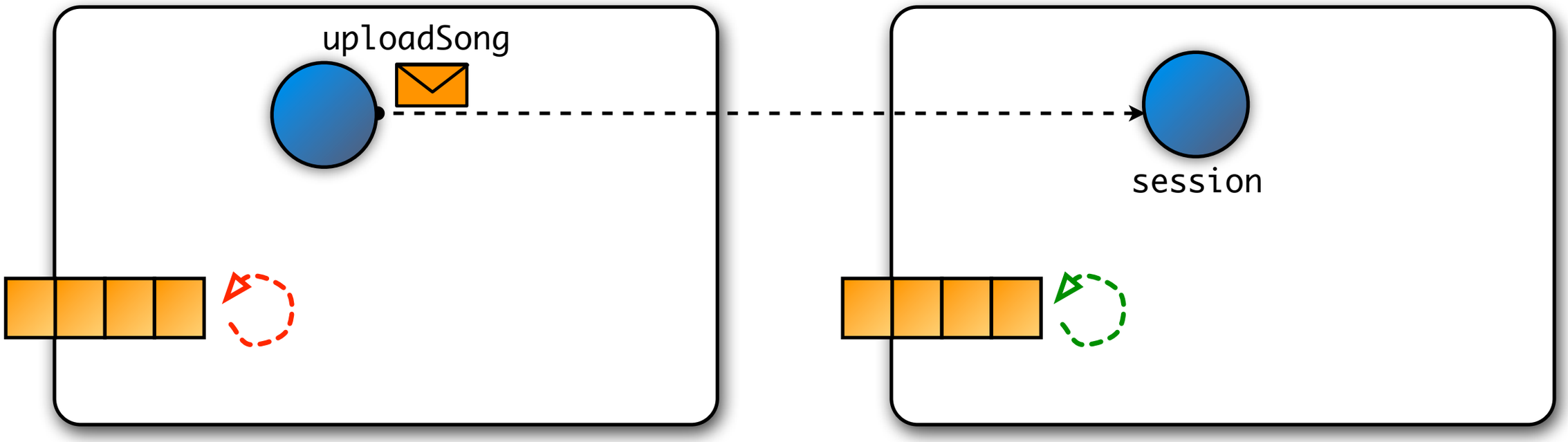



```
deftype MusicPlayer
```

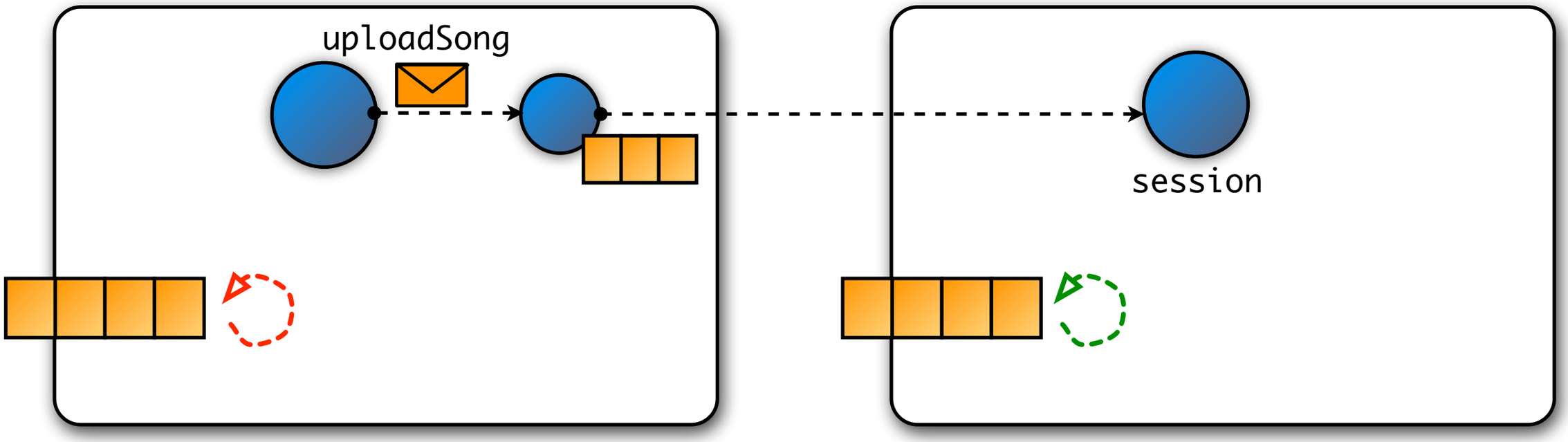
```
deftype MusicPlayer

export: mplayer as: MusicPlayer
```

```
whenever: MusicPlayer discovered: { |mplayer|
  // open a session
}
```

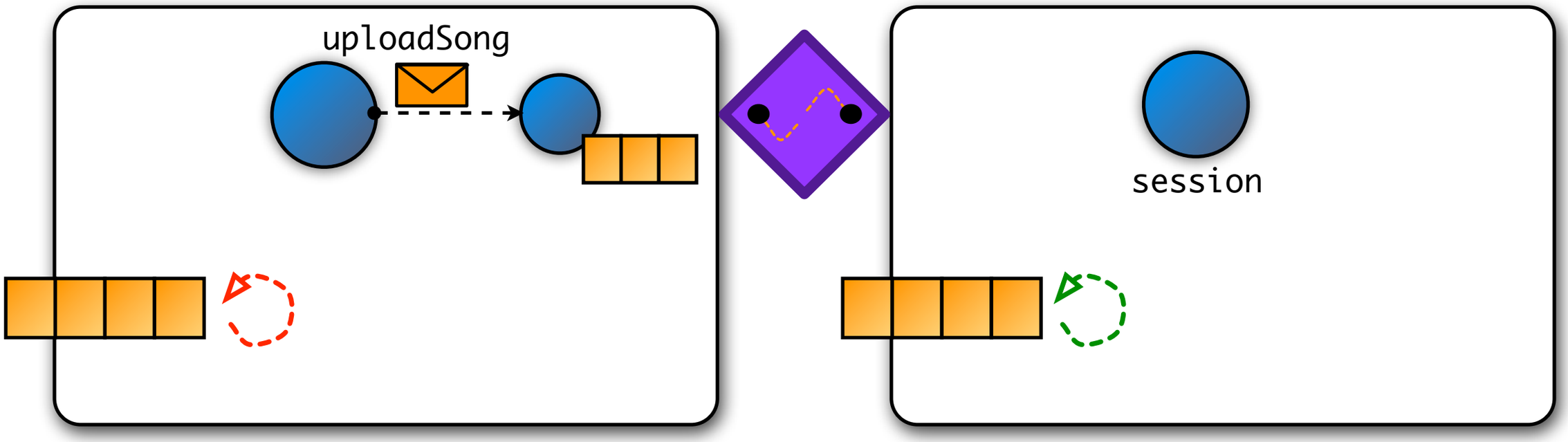
Far References



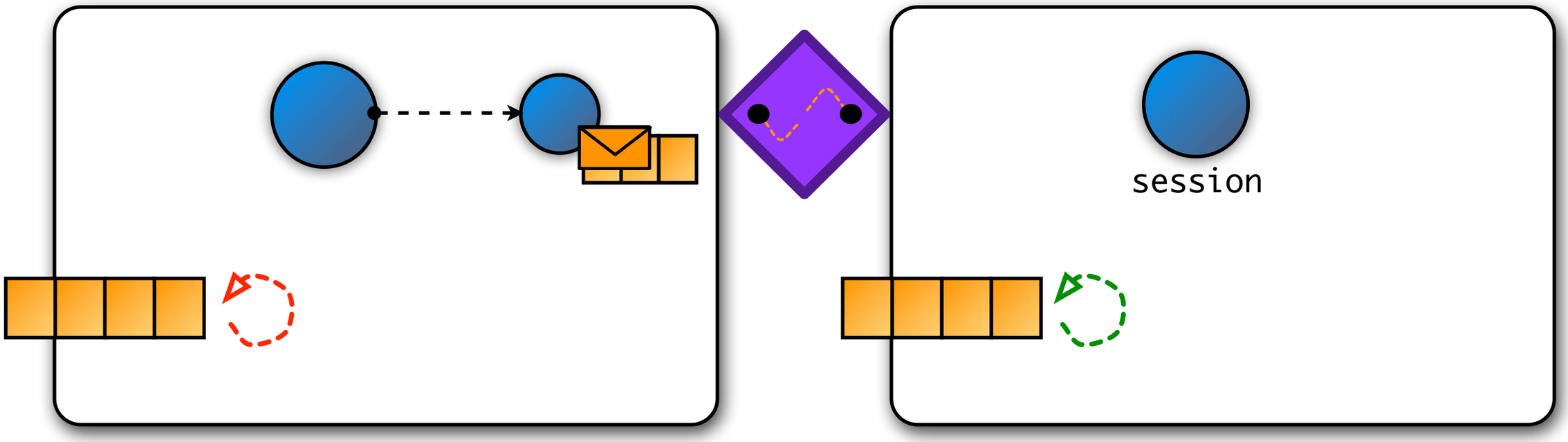
Far References



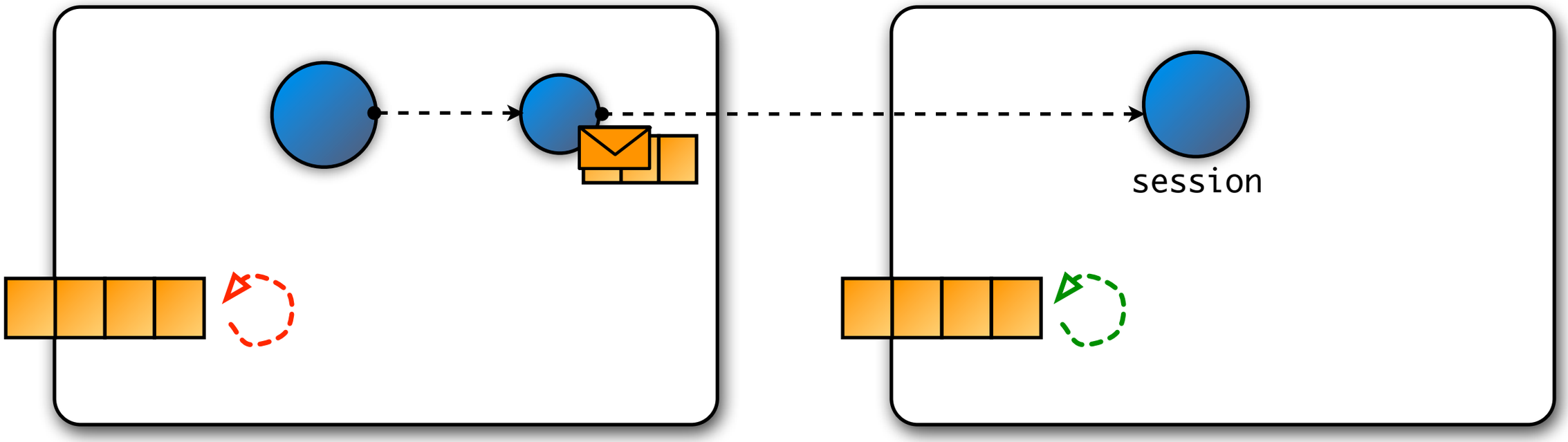
Far References



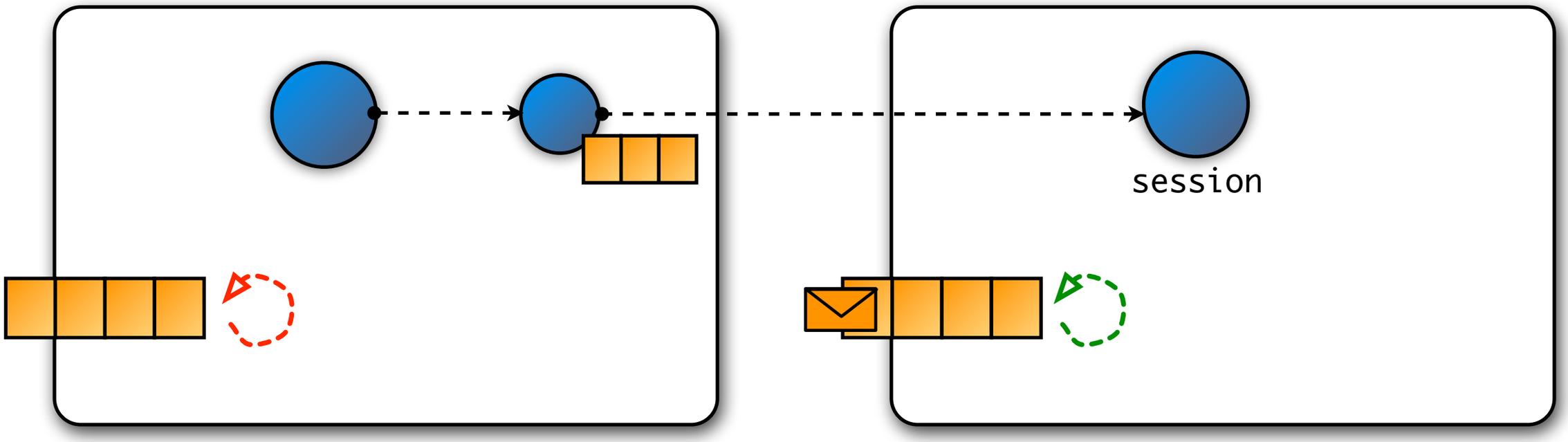
Far References



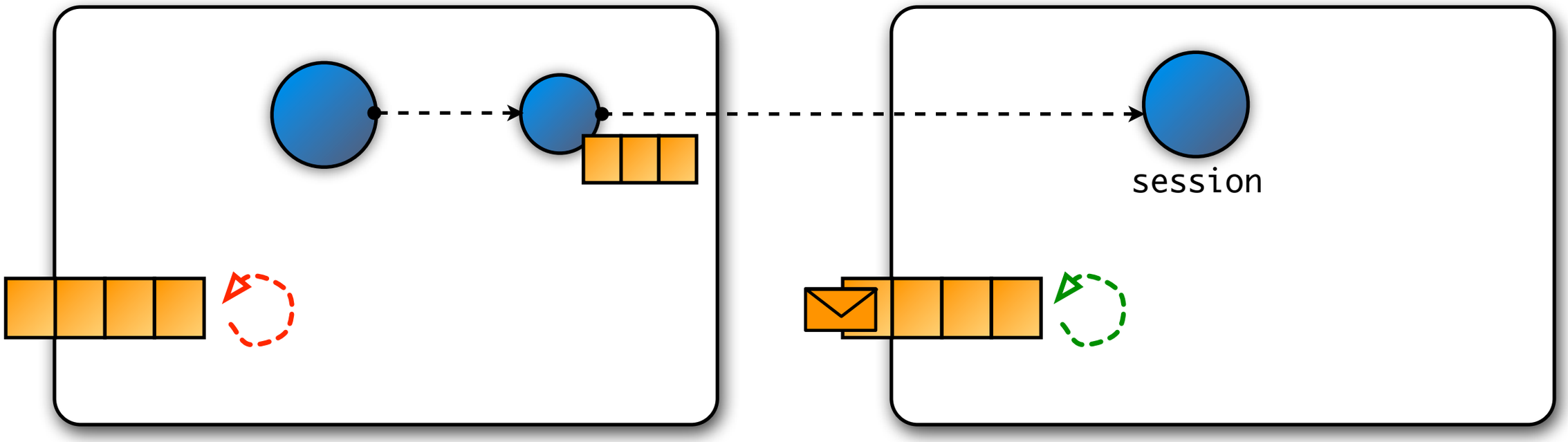
Far References



Far References



Far References




```
when: session<-uploadSong(s)@Due(timeout) becomes: { |ack|
  // continue exchange
} catch: TimeoutException using: { |e|
  // stop exchange
}
```

AmbientTalk = OO + Events

Generate and receive application requests

```
obj<-msg(arg)
def msg(param) { ... }
```

Follow-up on outstanding requests

```
when: future becomes: { |result| ... }
```

React to services appearing and disappearing

```
when: type discovered: { |ref| ... }
```

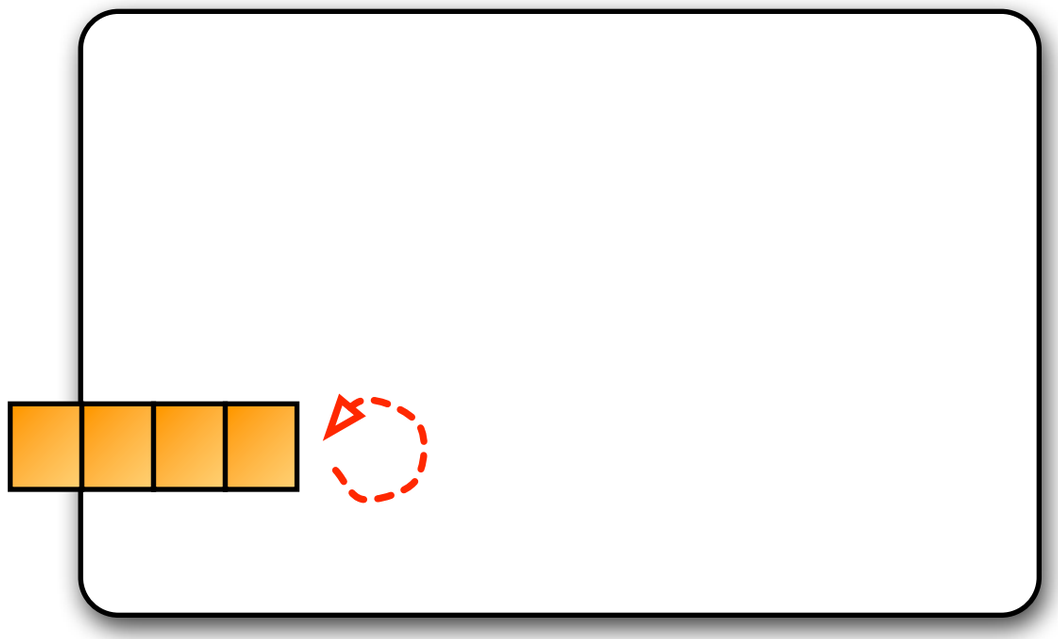
React to references disconnecting, reconnecting, expiring

```
when: ref disconnected: { ... }

when: ref reconnected: { ... }

when: ref expired: { ... }
```

AmbientTalk = OO + Events

- Event notification = sending an `apply` message to a block
- `apply` message is executed in its own event loop turn

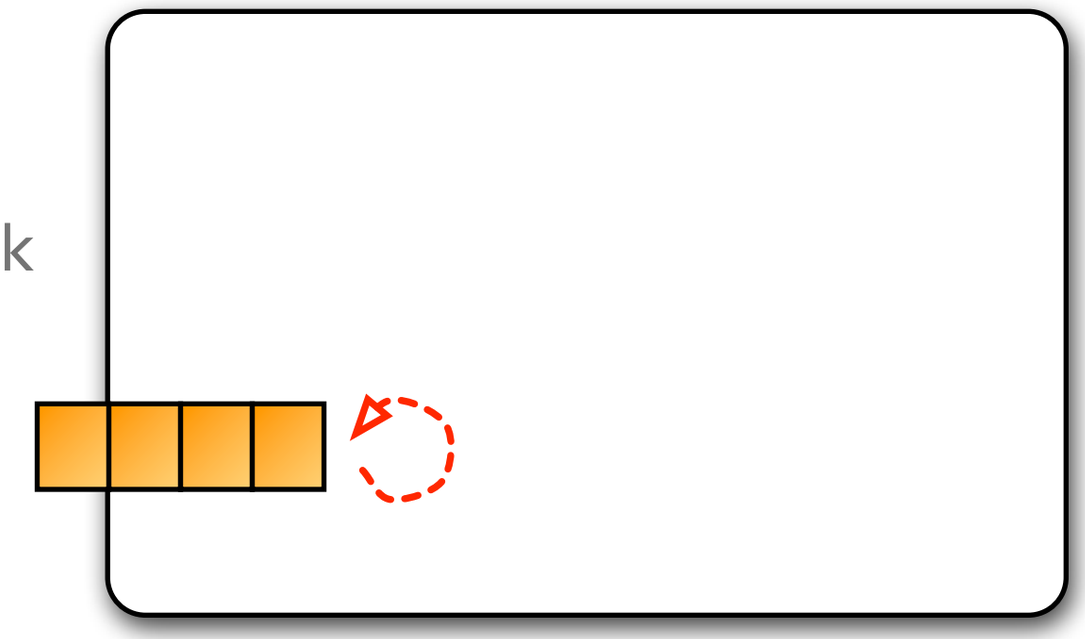
```
when: MusicPlayer discovered: { |p|
   ...
}
```



AmbientTalk = OO + Events

- Event notification = sending an `apply` message to a block
- `apply` message is executed in its own event loop turn

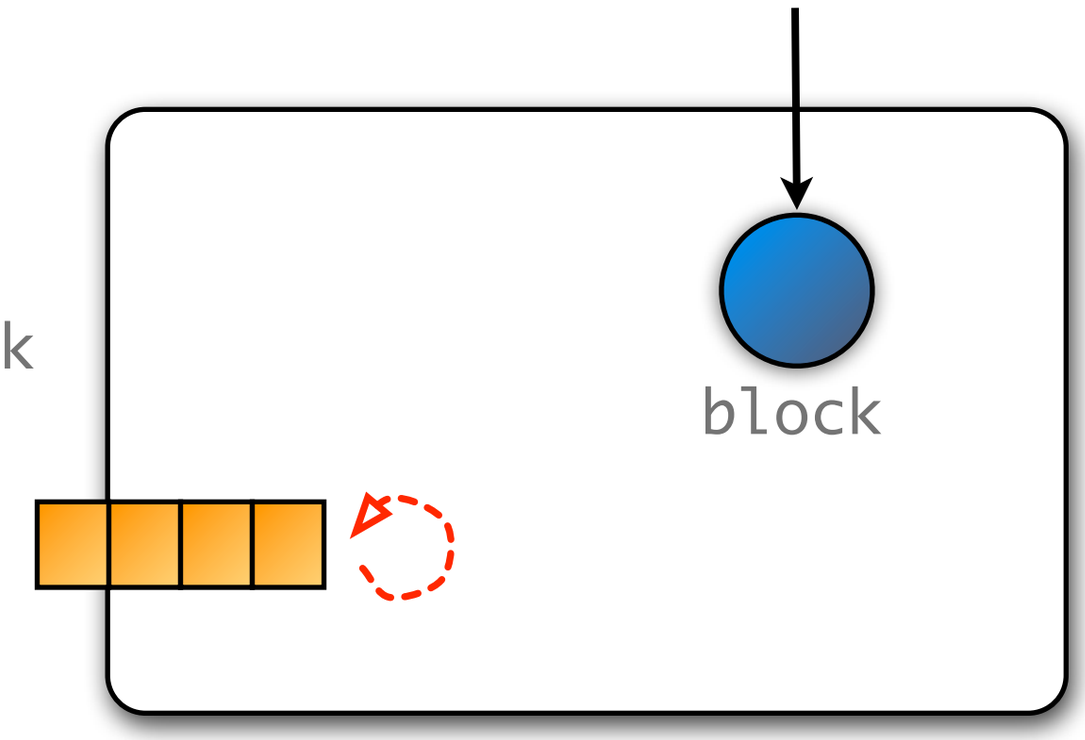
```
def block := { |p| ... }
when: MusicPlayer discovered: block
```



AmbientTalk = OO + Events

- Event notification = sending an `apply` message to a block
- `apply` message is executed in its own event loop turn

```
def block := { |pl| ... }
when: MusicPlayer discovered: block
```

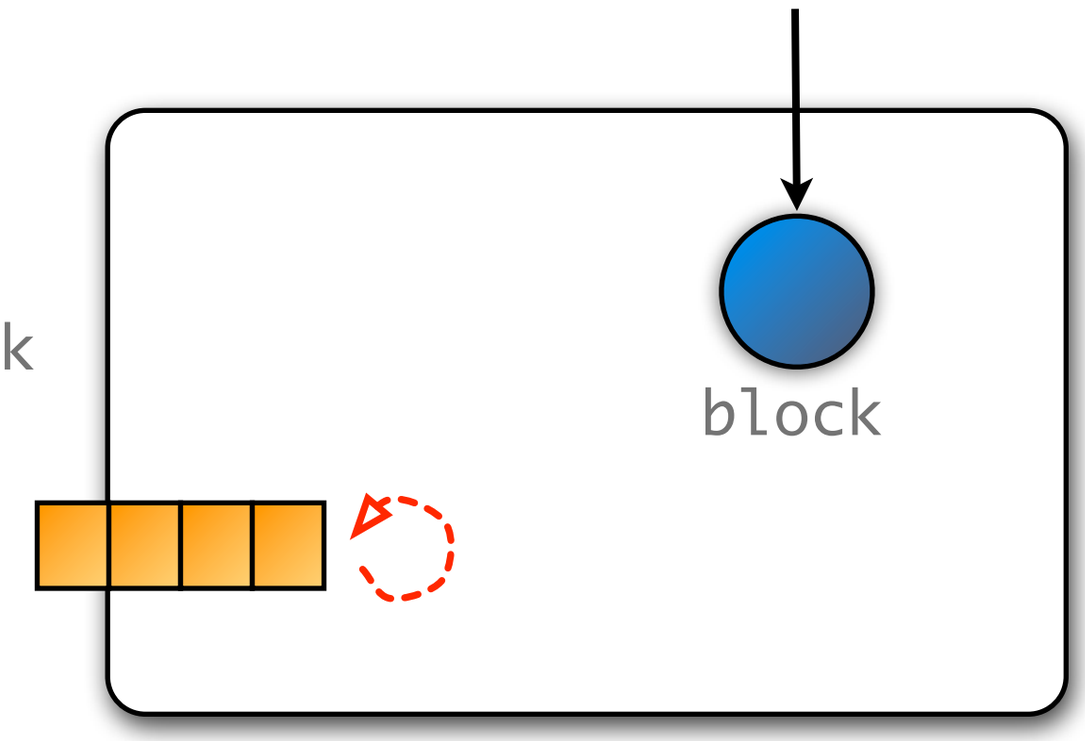


AmbientTalk = OO + Events

- Event notification = sending an `apply` message to a block
- `apply` message is executed in its own event loop turn

```
block<-apply(ref)
```

```
def block := { |p| ... }
when: MusicPlayer discovered: block
```



Demo

Demo

Mobile echo service

Demo

Mobile echo service

1. Discover

Demo

Mobile echo service

1. Discover

2. Communicate

Demo

Mobile echo service

1. Discover

2. Communicate

3. Deal with failures

```
deftype EchoService;

def echoF := when: EchoService discovered: { |echoSvc|
  system.println("Discovered an echo service");
  echoSvc;
} within: 2.minutes

echoF<-echo("test1");

def resultF := echoF<-echo("test2")@TwoWay;
when: resultF becomes: { |value|
  system.println("Reply: " + value);
}

echoF<-echo("test3");
```

```
def service := object: {
  def echo(text) {
    system.println("Received: "+text);
    text
  }
}

deftype EchoService;

def pub := export: service as: EchoService;
```

Experiences

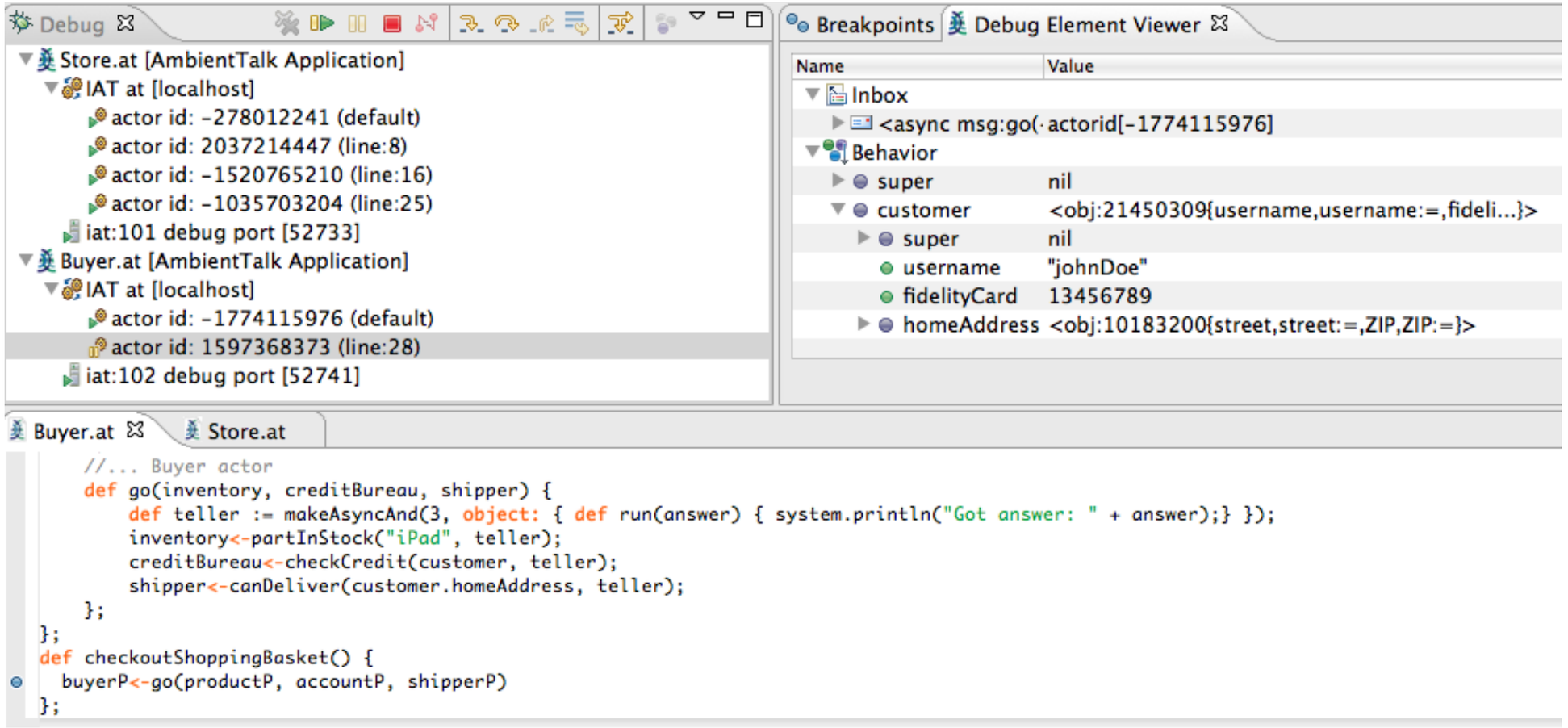
Applications

- P2P chat, music match maker, picture sharing, ...
- P2P multiplayer games (Atari Pong game, rock-paper-scissors, urban game using GPS coordinates)
- Collaborative drawing app


we
scribble

REME-D: Distributed Debugger

- Editor, debugger (inspect actor state, mailbox, breakpoints on messages)
- Eclipse plug-in



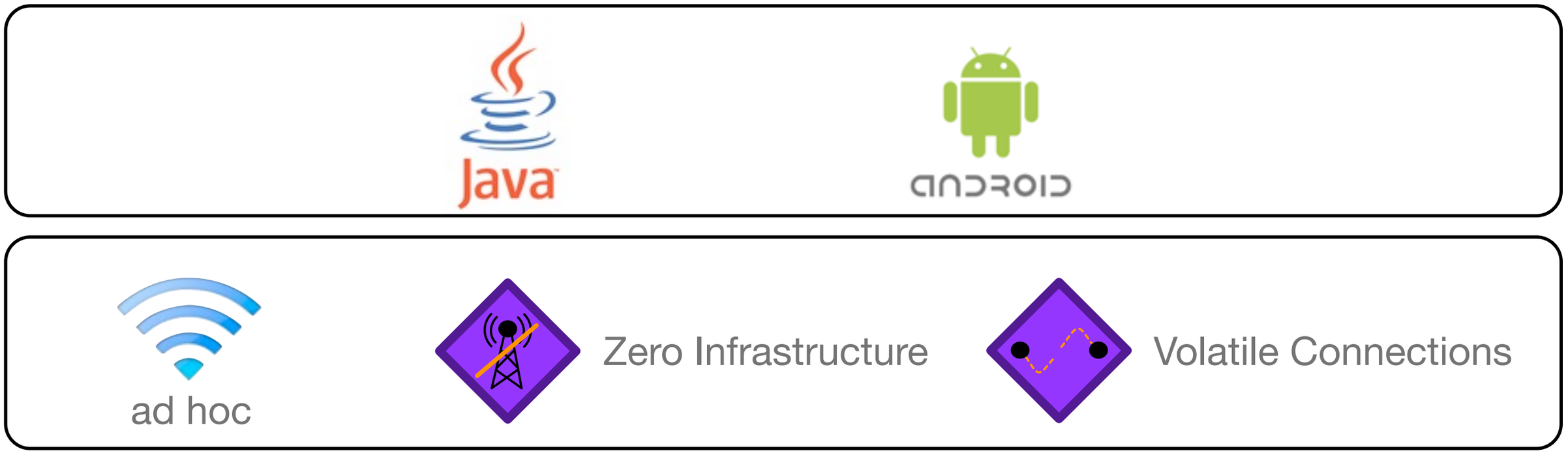
Operational semantics

Small-step operational semantics

- Covers actors, objects, futures, discovery, fault-tolerant async messages
- Executable in PLT Redex

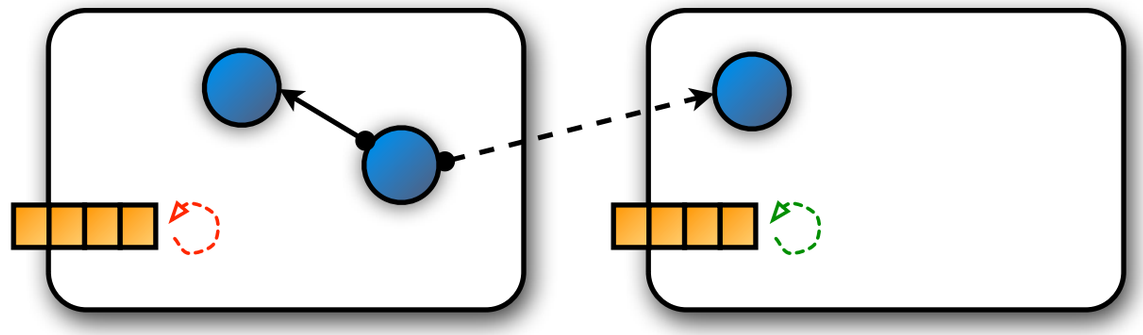
Tom Van Cutsem, Christophe Scholliers, Dries Harnie, Wolfgang De Meuter.
An operational semantics of Event Loop Concurrency in AmbientTalk
Tech. report VUB-SOFT-TR-12-04, April 2012

Summary


Java
android
ad hoc
Zero Infrastructure
Volatile Connections

Summary

AMBIENTTALK



Decentralized Discovery

Asynchronous Communication

Non-blocking Synchronisation

Disconnections **≠ Failures**

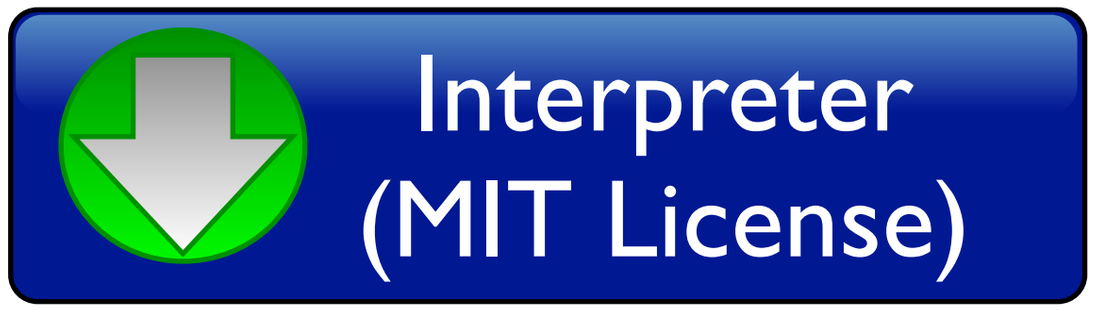
Java

ANDROID

ad hoc

Zero Infrastructure

Volatile Connections



ambienttalk.googlecode.com