Ambient-oriented Programming & AmbientTalk

Tom Van Cutsem Stijn Mostinckx Elisa Gonzalez Boix
Andoni Lombide Christophe Scholliers Wolfgang De Meuter

Software Languages Lab
Brussels, Belgium

Vrije Universiteit Brussel

Agenda

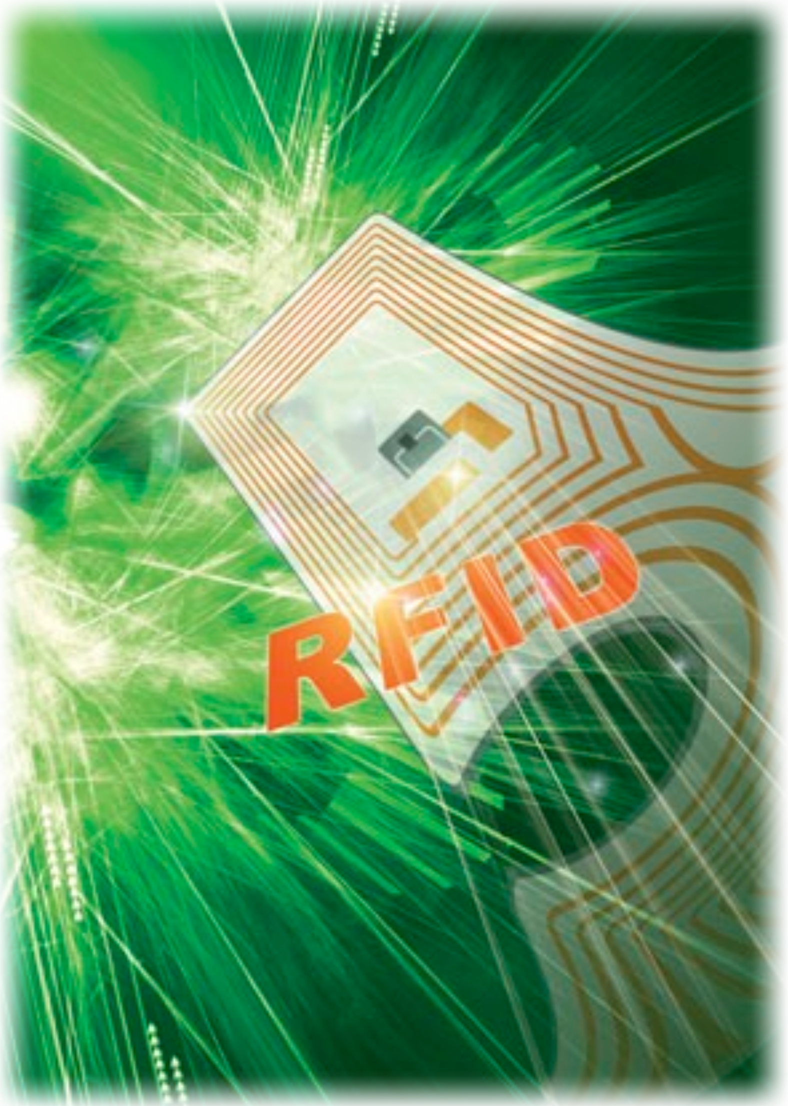
- Context: Mobile & Ubiquitous computing
- Approach: Ambient-oriented programming
- Tool: AmbientTalk
- Experiments: Demo applications

Context

Mobile & Ubiquitous Computing

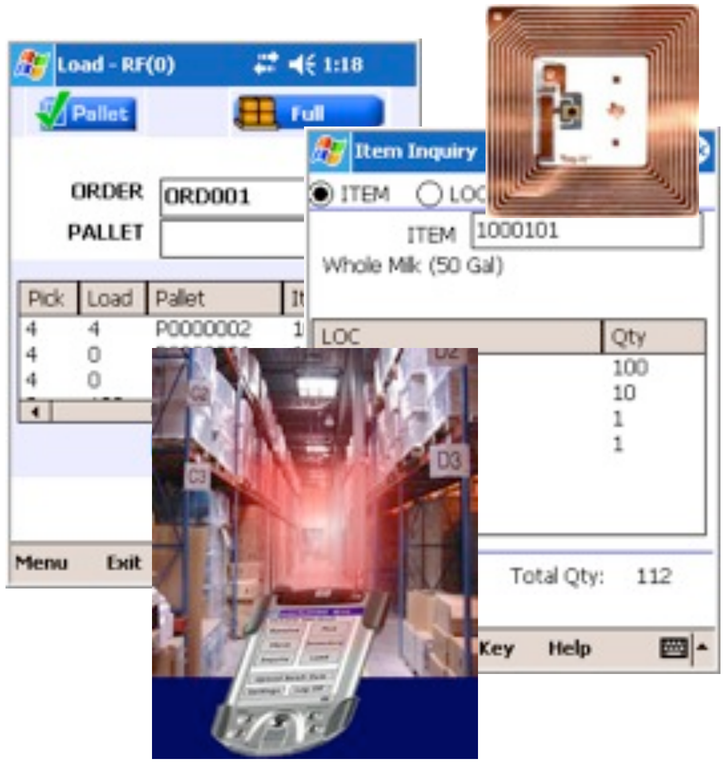
Ubiquitous Computing

- Research vision postulated by Mark Weiser (1988, Xerox PARC)



Today’s Applications

Smart Homes/Domotics



ORDER ORD001

PALLET

Pick	Load	Pallet	It
4	4	P0000002	1
4	0		
4	0		

● ITEM ○ LOC

ITEM 1000101

Whole Milk (50 Gal)

LOC	Qty
	100
	10
	1
	1

Total Qty: 112

RFID Inventory Management

Tourism/City Guide Software

Personal Area Networks

Issues

- Hardware Issues:
  - Miniaturisation
  - Device Autonomy
  - Interoperability
  - Processor Speed
  - Limited Memory
  - Integration
  - Cost

- Software Issues:
  - Context-awareness
  - Interaction with real world
  - Portability
  - New user interfaces
  - Standards
  - Distributed Applications

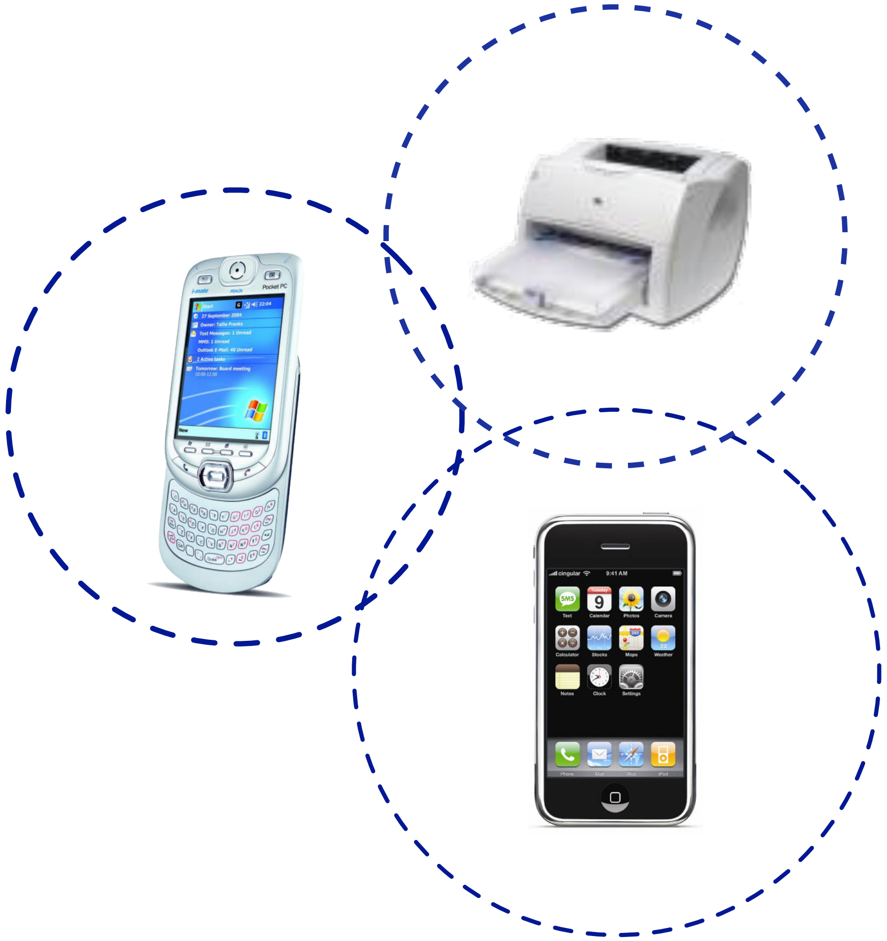
Issues

- Hardware Issues:
  - Miniaturisation
  - Device Autonomy
  - Interoperability
  - Processor Speed
  - Limited Memory
  - Integration
  - Cost

- Software Issues:
  - Context-awareness
  - Interaction with real world
  - Portability
  - New user interfaces
  - Standards
  - Distributed Applications

Mobile Ad Hoc Networks

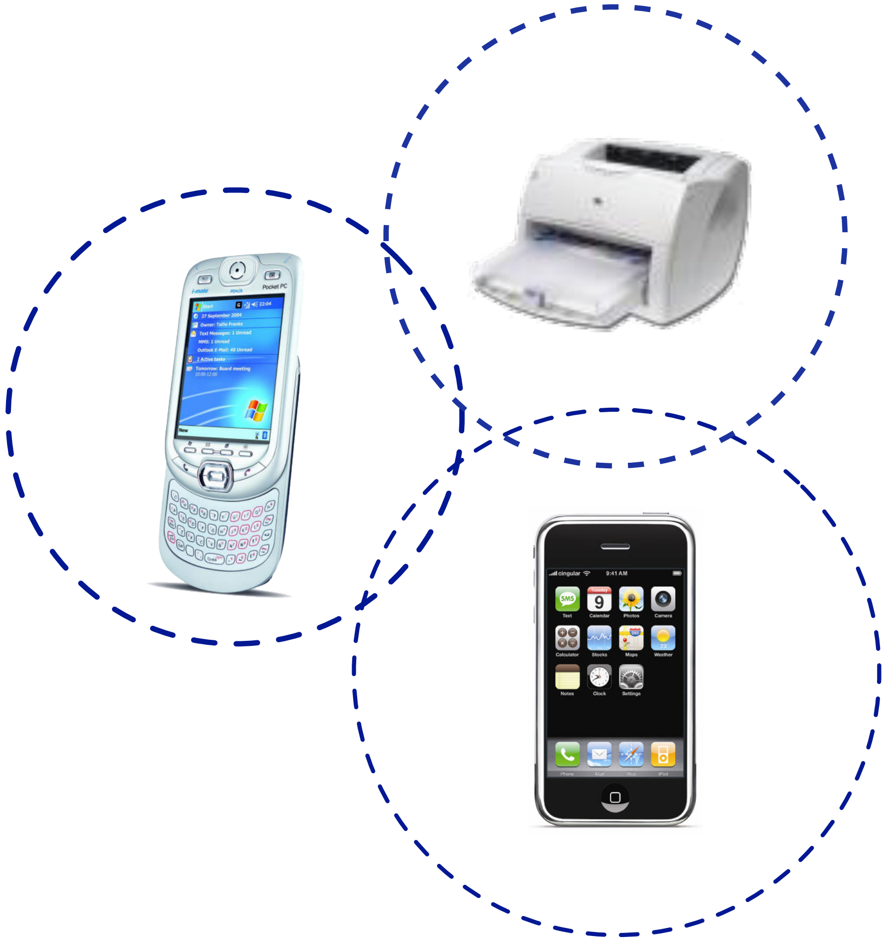
Networks composed of **mobile** devices that communicate **wirelessly**



Mobile Ad Hoc Networks

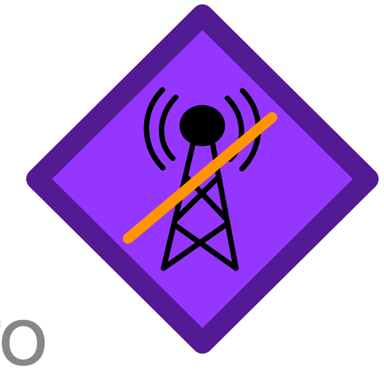
Networks composed of **mobile** devices that communicate **wirelessly**

Zero Infrastructure

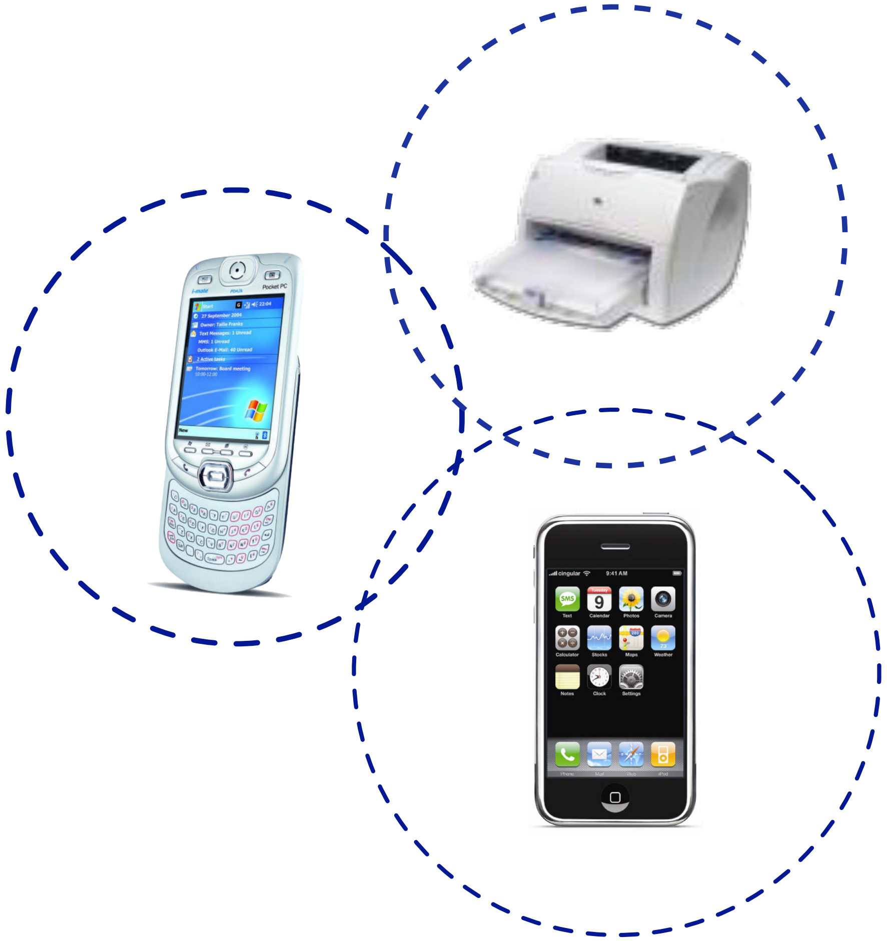


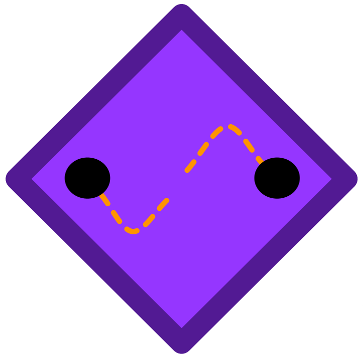
Mobile Ad Hoc Networks

Networks composed of **mobile** devices that communicate **wirelessly**



Zero Infrastructure





Volatile Connections

Approach

Ambient-oriented Programming

Observation #1: interaction with proximate peers

Example: match making between proximate peers

demand

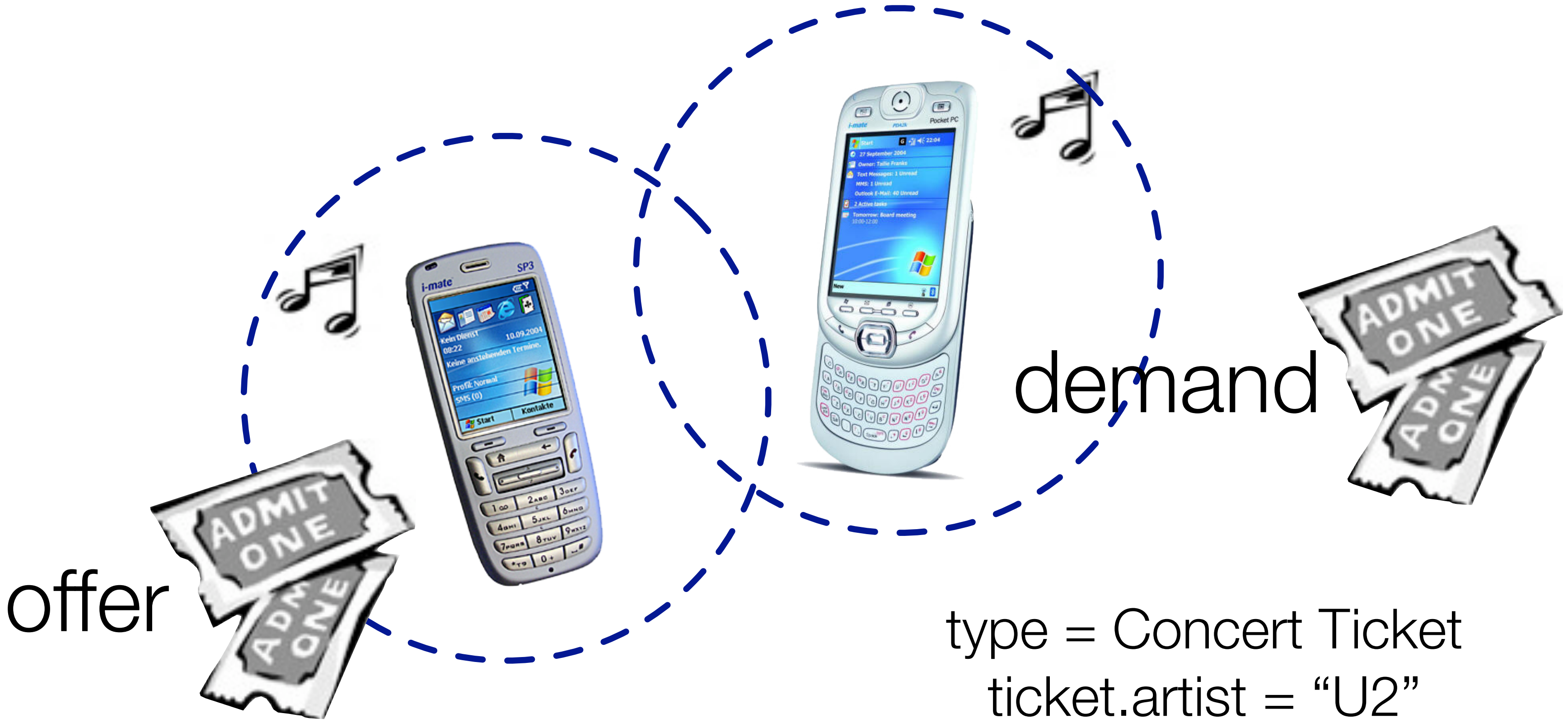
type = Concert Ticket
ticket.artist = “U2”
ticket.location = *

offer

type = Concert Ticket
ticket.artist = “U2”
ticket.location = “Hyde Park”

Observation #1: interaction with proximate peers

Example: match making between proximate peers



offer

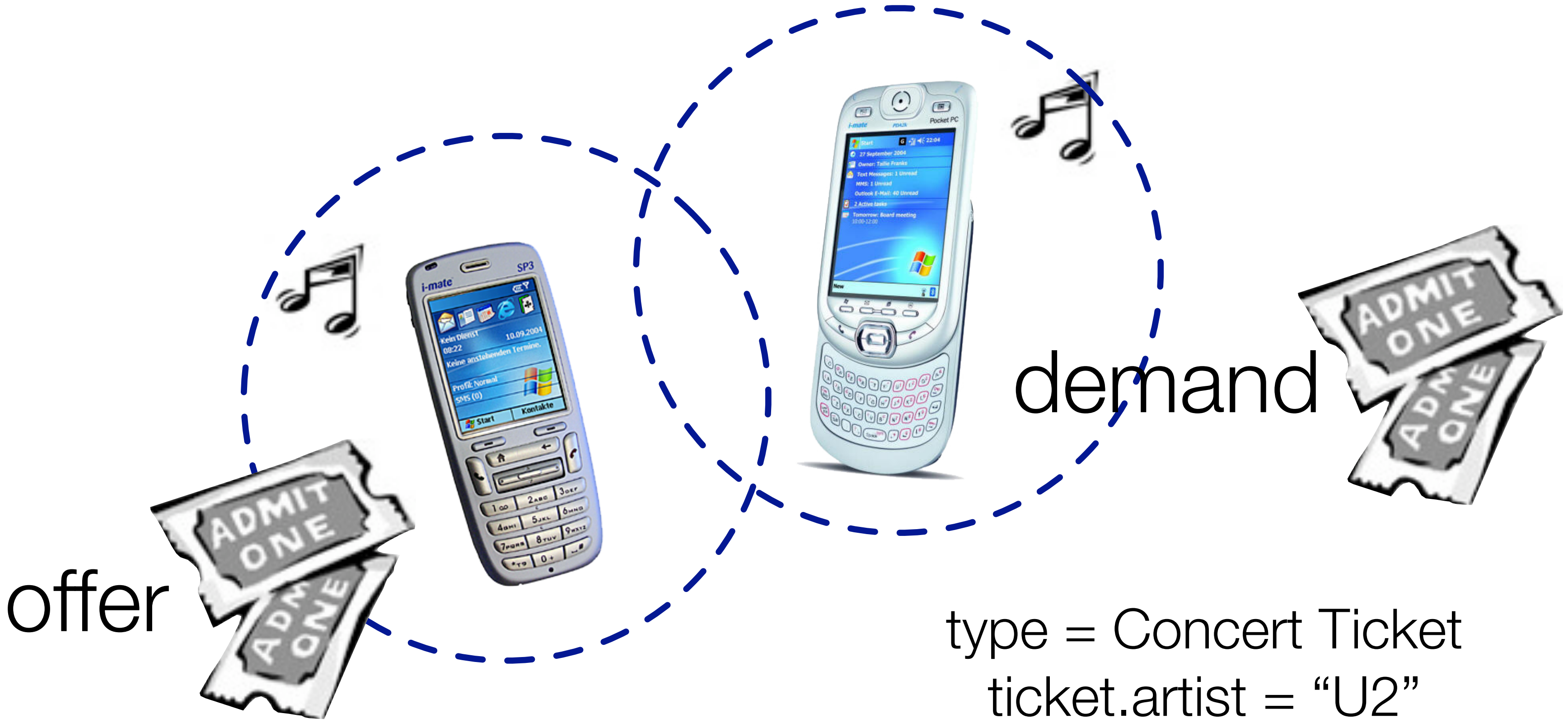
demand

type = Concert Ticket
ticket.artist = “U2”
ticket.location = “Hyde Park”

type = Concert Ticket
ticket.artist = “U2”
ticket.location = *

Observation #1: interaction with proximate peers

Example: match making between proximate peers

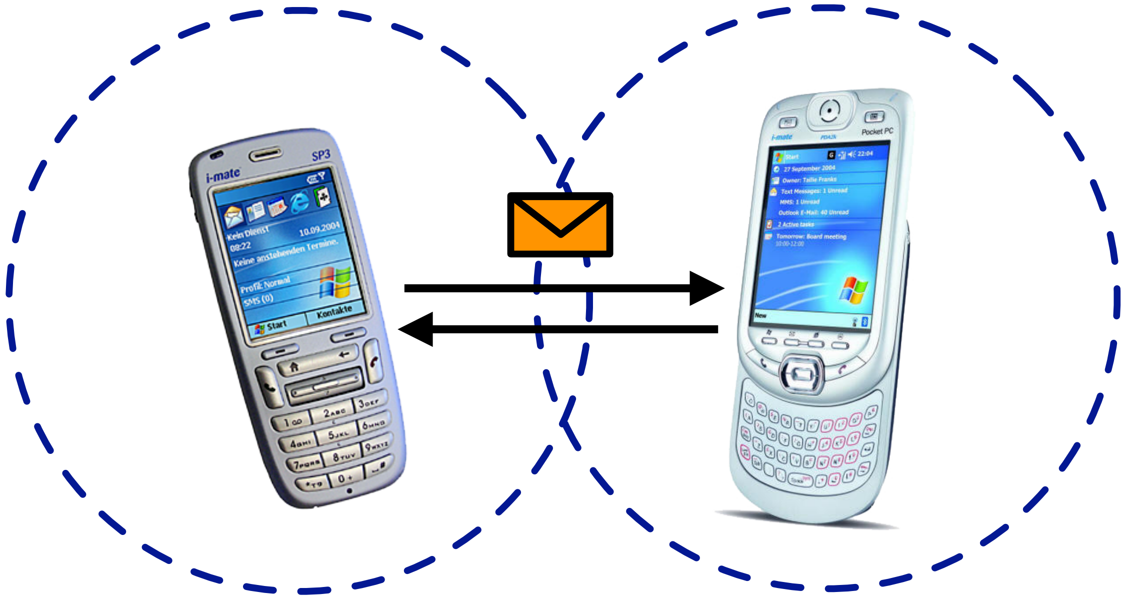


type = Concert Ticket
ticket.artist = “U2”
ticket.location = *

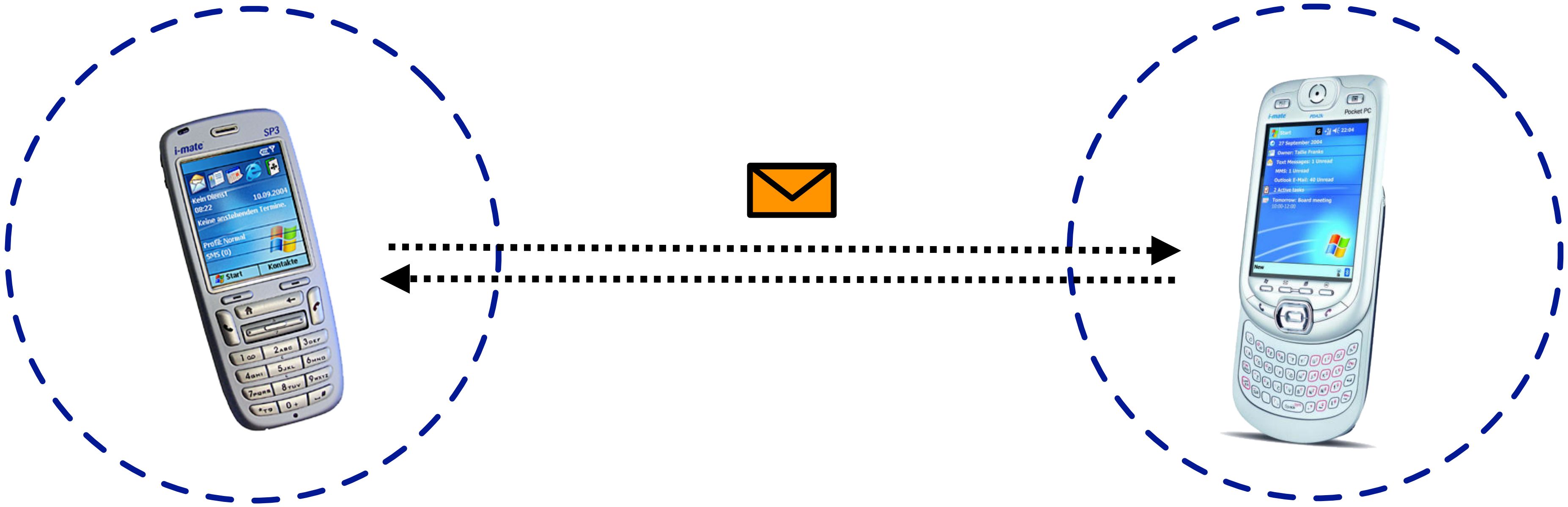
type = Concert Ticket
ticket.artist = “U2”
ticket.location = “Hyde Park”

No reliance on fixed, always-available server infrastructure

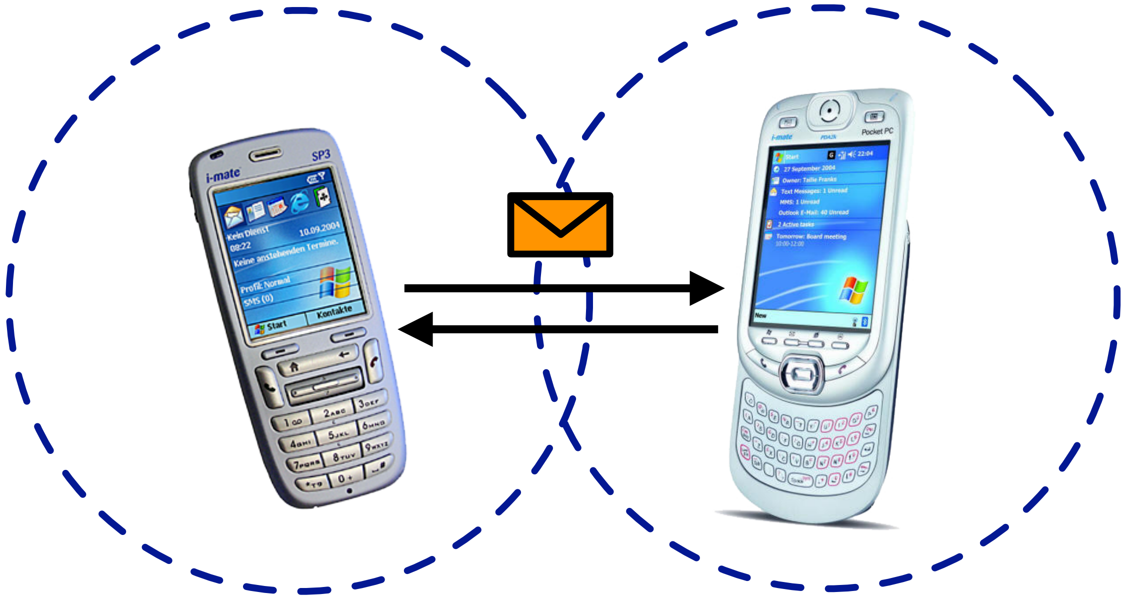
Observation #2: intermittent connectivity



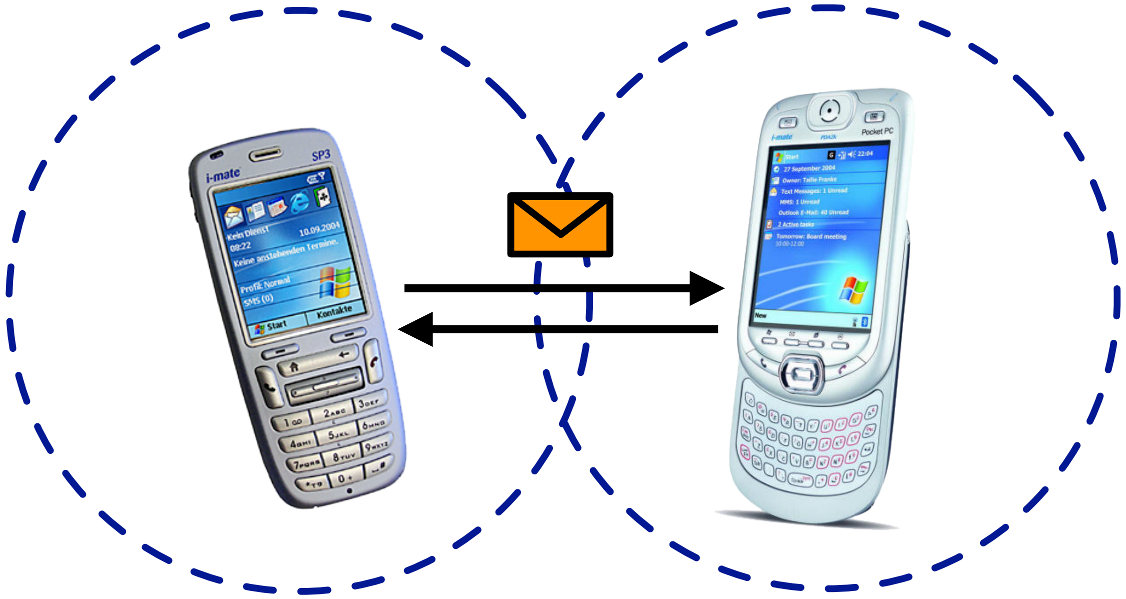
Observation #2: intermittent connectivity



Observation #2: intermittent connectivity



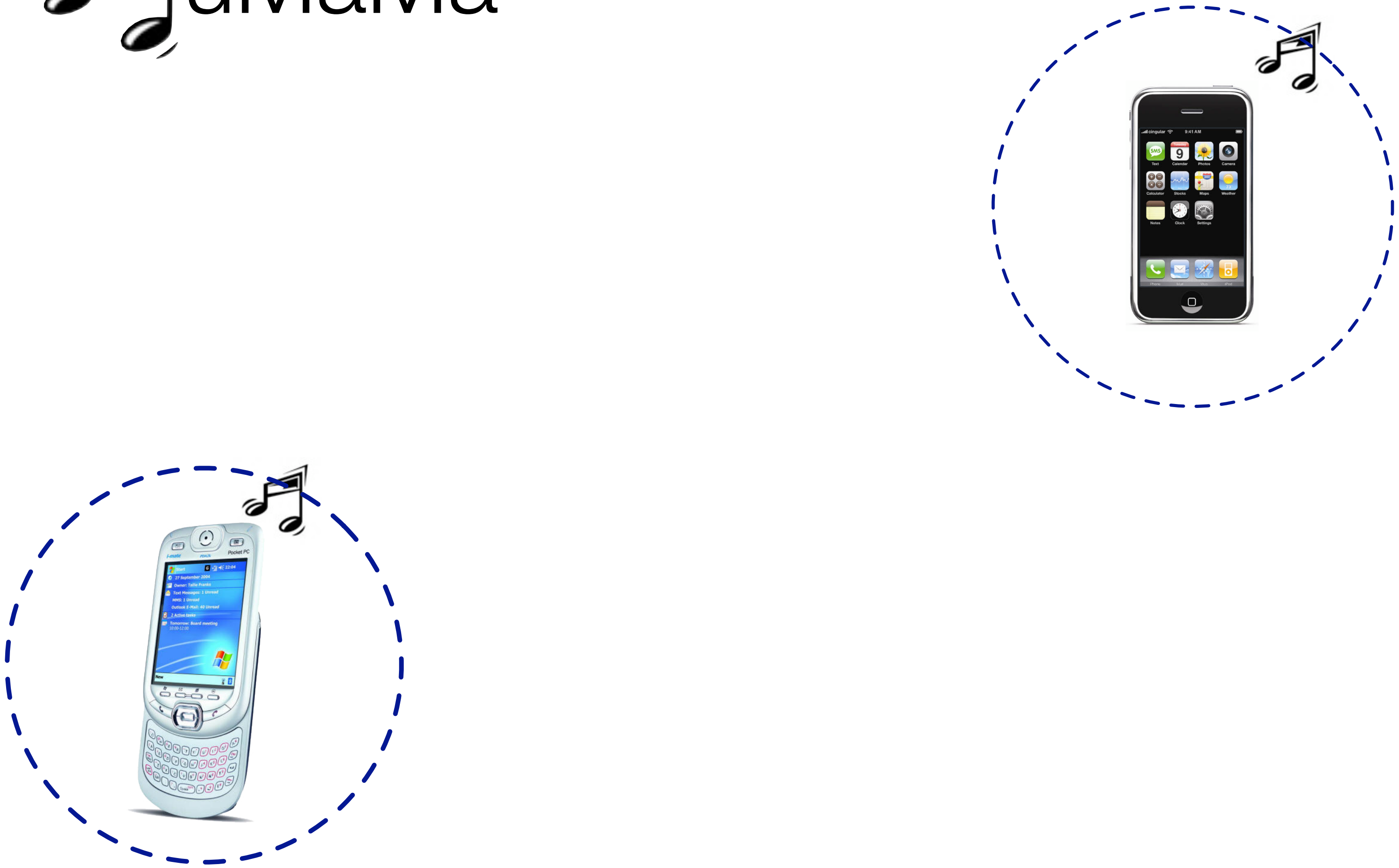
Observation #2: intermittent connectivity



Tolerate disconnections, because they occur frequently rather than exceptionally

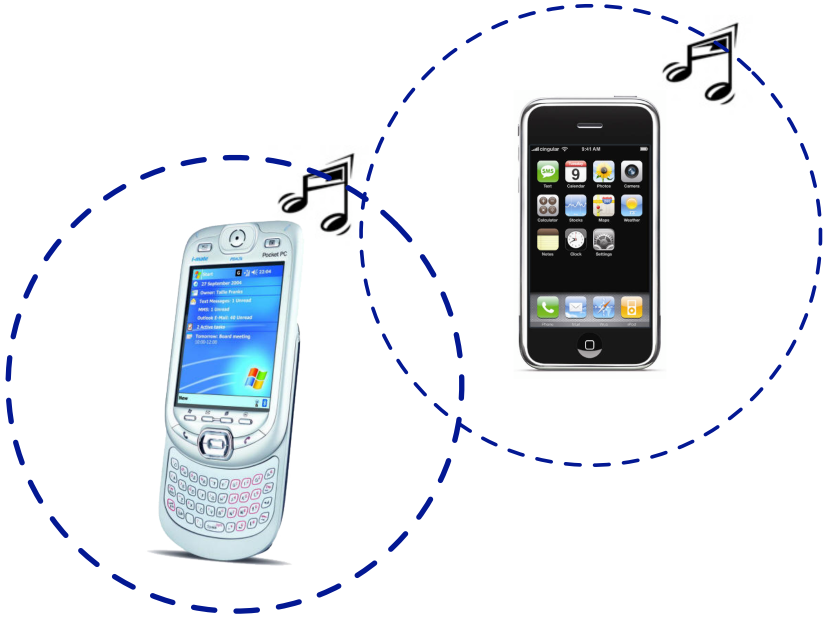
Software concerns

uMaMa



Software concerns

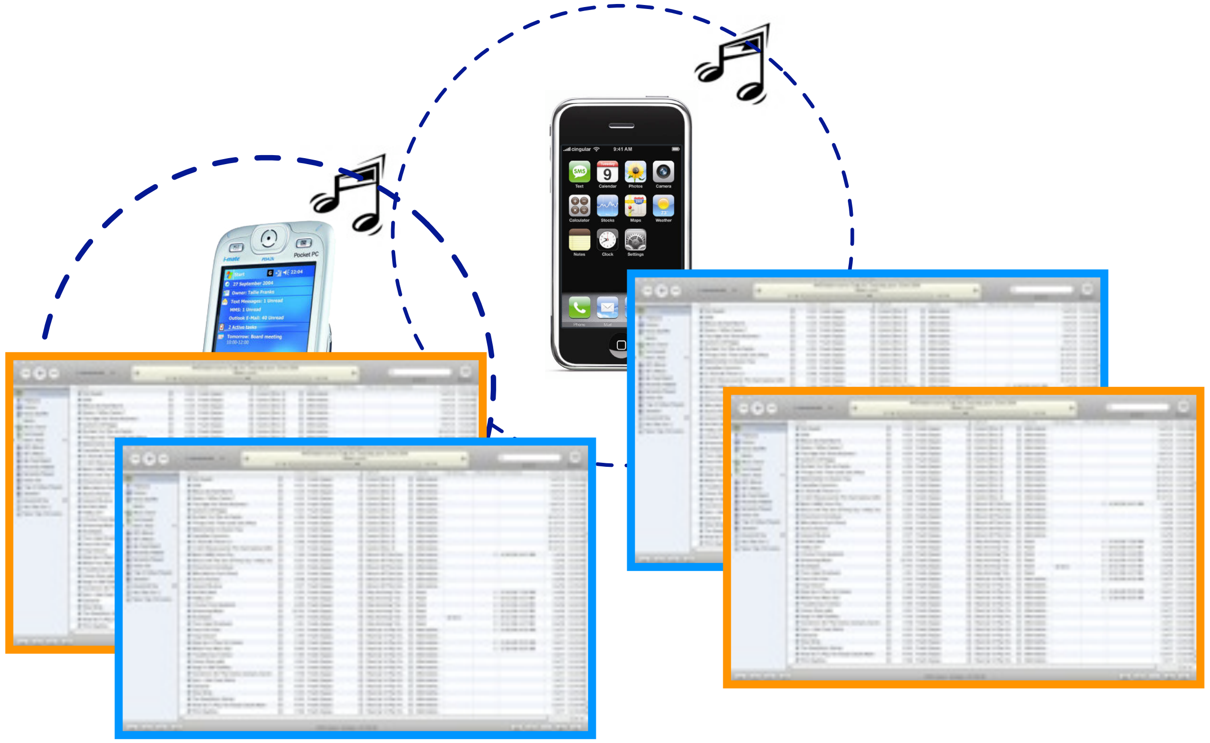
uMaMa



Discovery

Software concerns

uMaMa

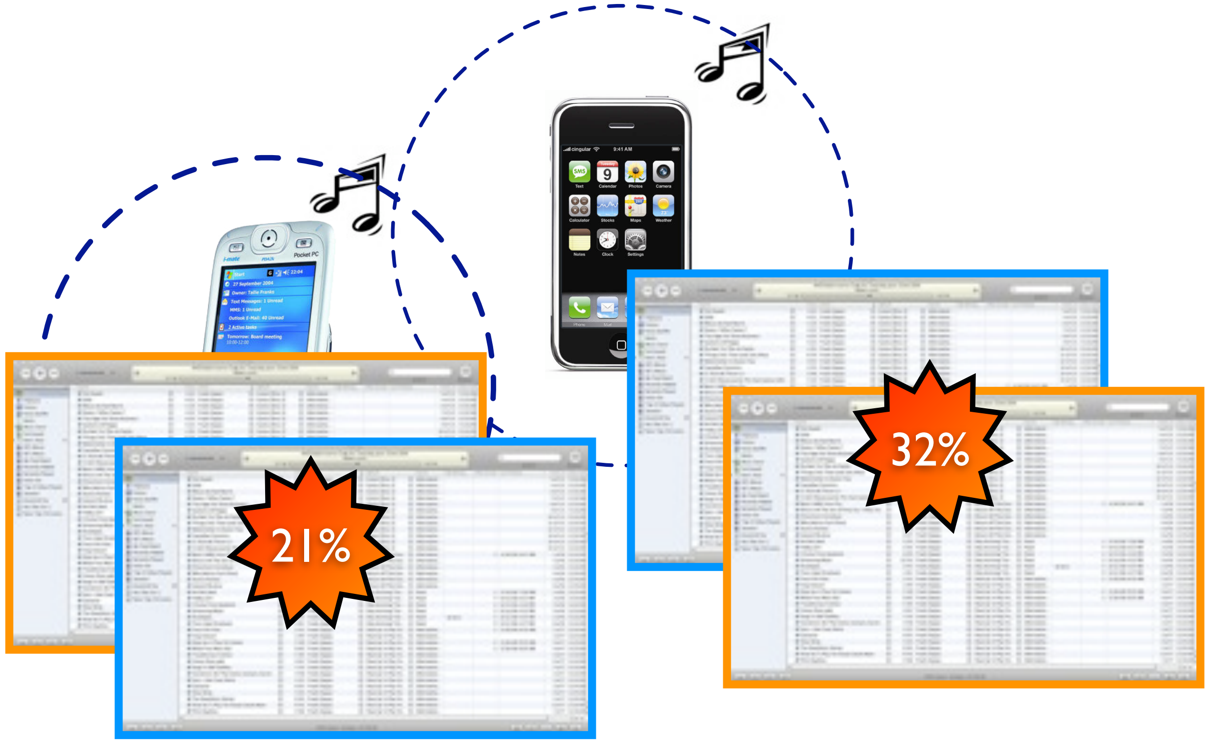


Discovery

Communication

Software concerns

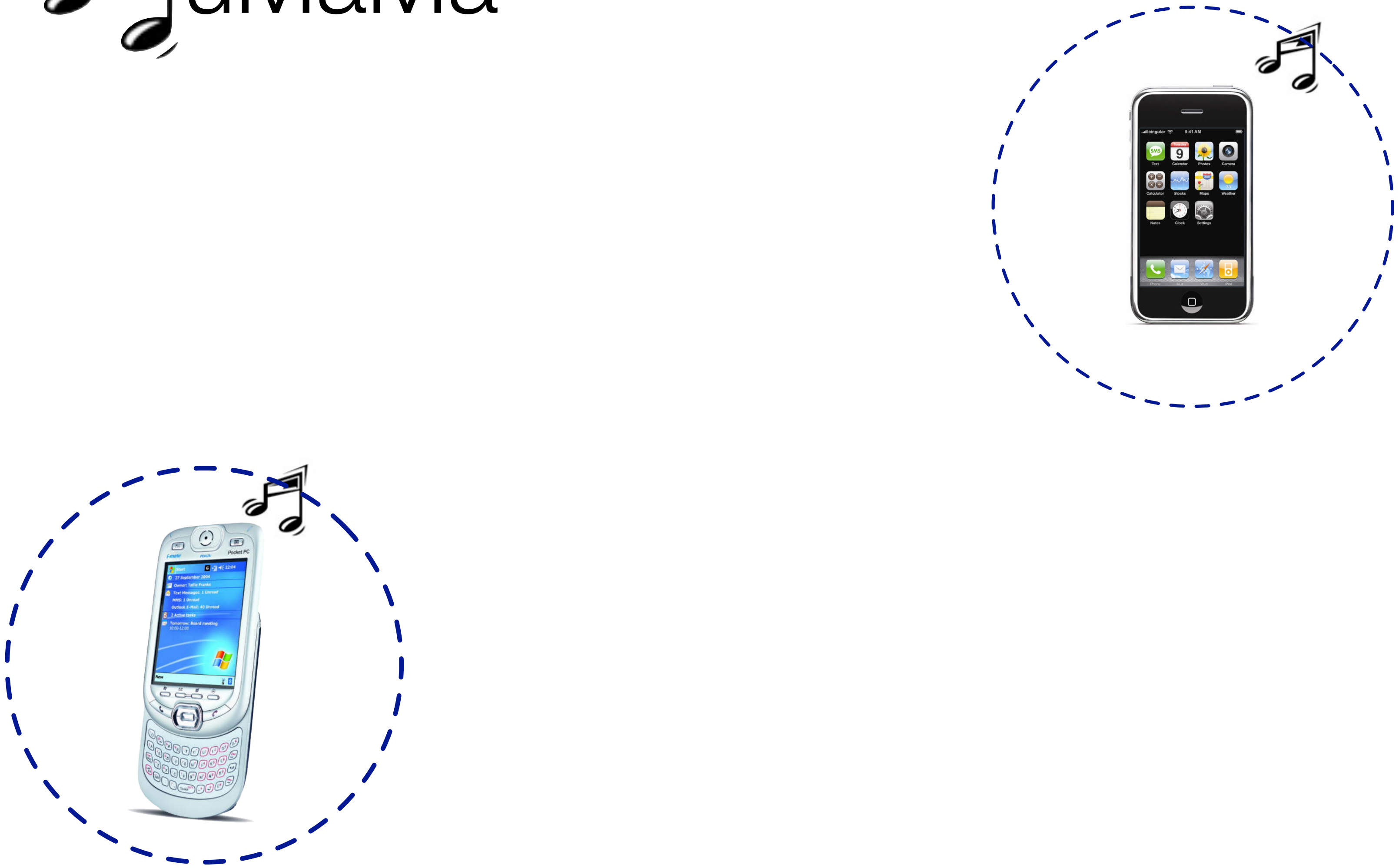
uMaMa



Discovery Communication Synchronisation

Software concerns

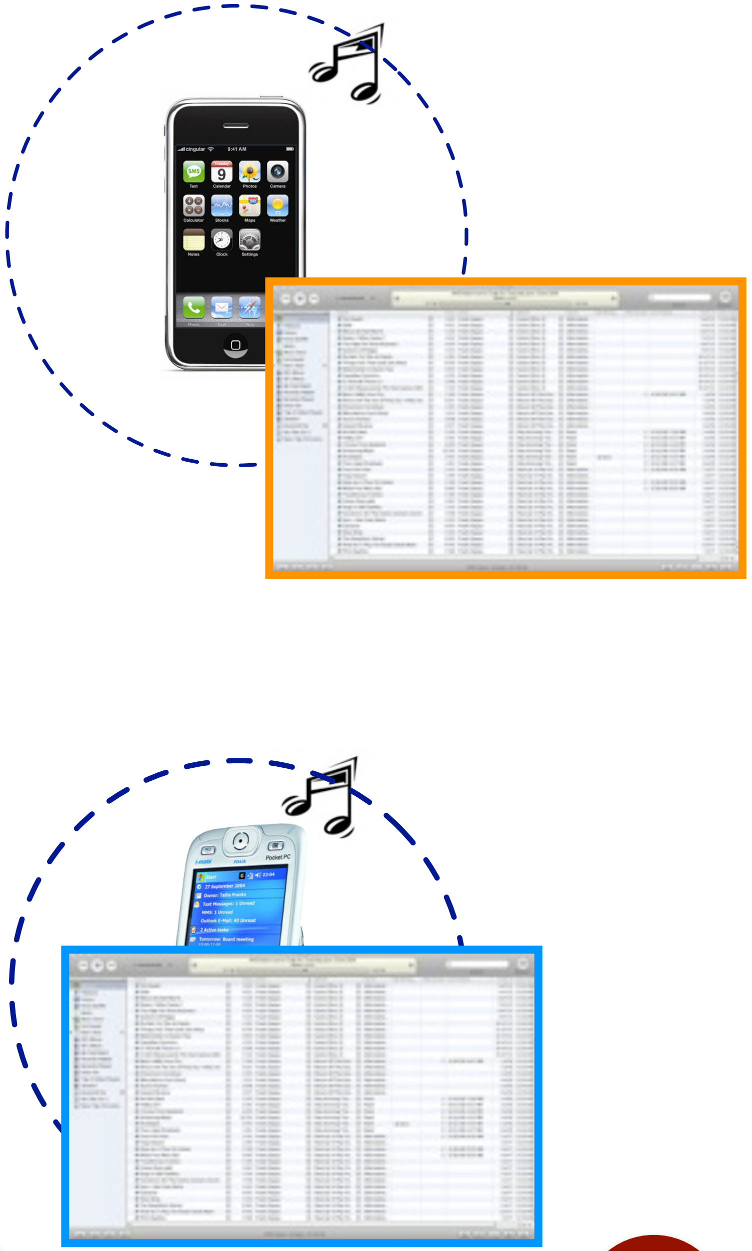
uMaMa



Discovery Communication Synchronisation

Software concerns

uMaMa



● Discovery ⇄ Communication ✓ Synchronisation ✖ Failure handling

Tool

AmbientTalk

AmbientTalk: fact sheet

- Object-oriented scripting language
- Started in 2005
- Pure Java implementation
- Runs on J2ME/CDC phones
- Open source implementation

code.google.com/p/ambienttalk

Google Code

AMBIENTTALK

How does AmbientTalk help?

Volatile Connections

network connections are resilient to failures by default

Zero Infrastructure

service discovery protocol built into the language

AMBIENTTALK

Object-oriented

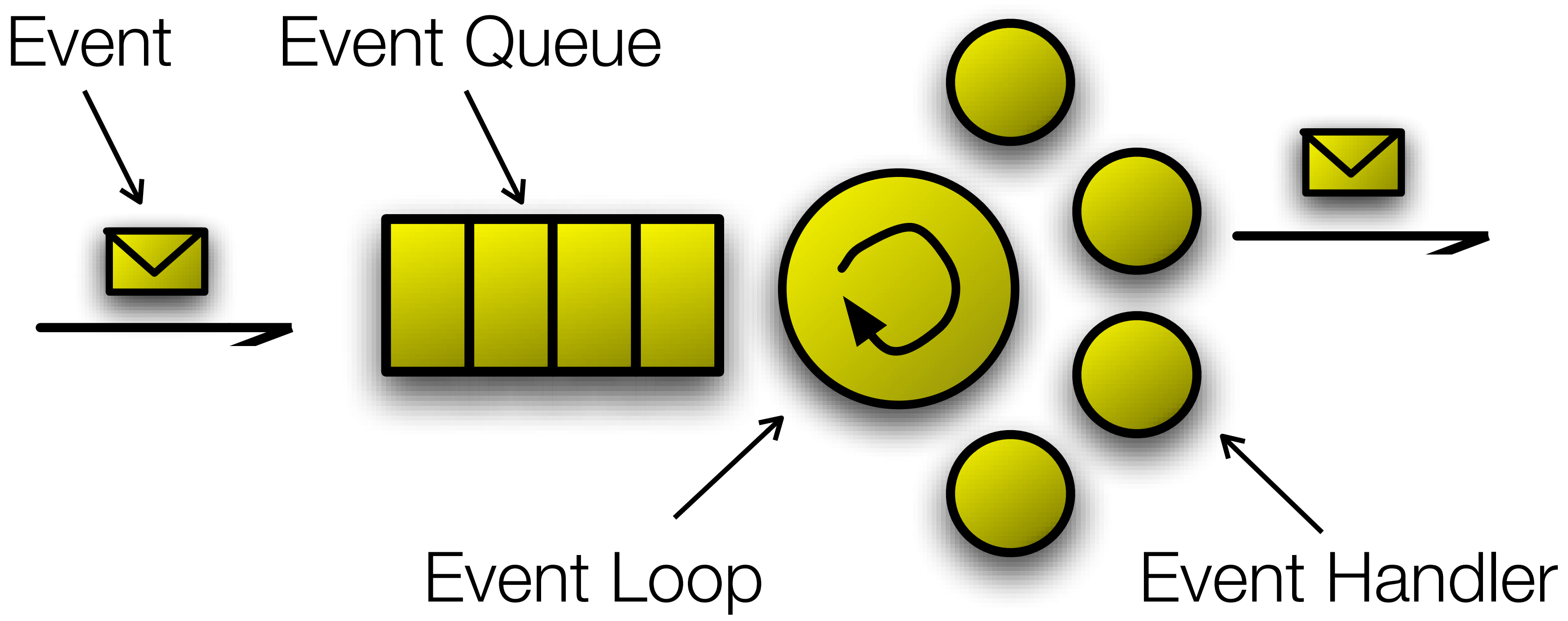
```
def makeSong(artist, title) {
  object: {
    def printArtist() {
      if: (artist == nil) then: {
        “unknown artist”;
      } else: {
        artist;
      }
    }
  }
}
```

Object-oriented

```
def makeSong(artist, title) {
  object: {
    def printArtist() {
      if: (artist == nil) then: {
        “unknown artist”;
      } else: {
        artist;
      }
    }
  }
}

def song := makeSong(“U2”, “One”);
song.printArtist();
```

Event Loop Concurrency

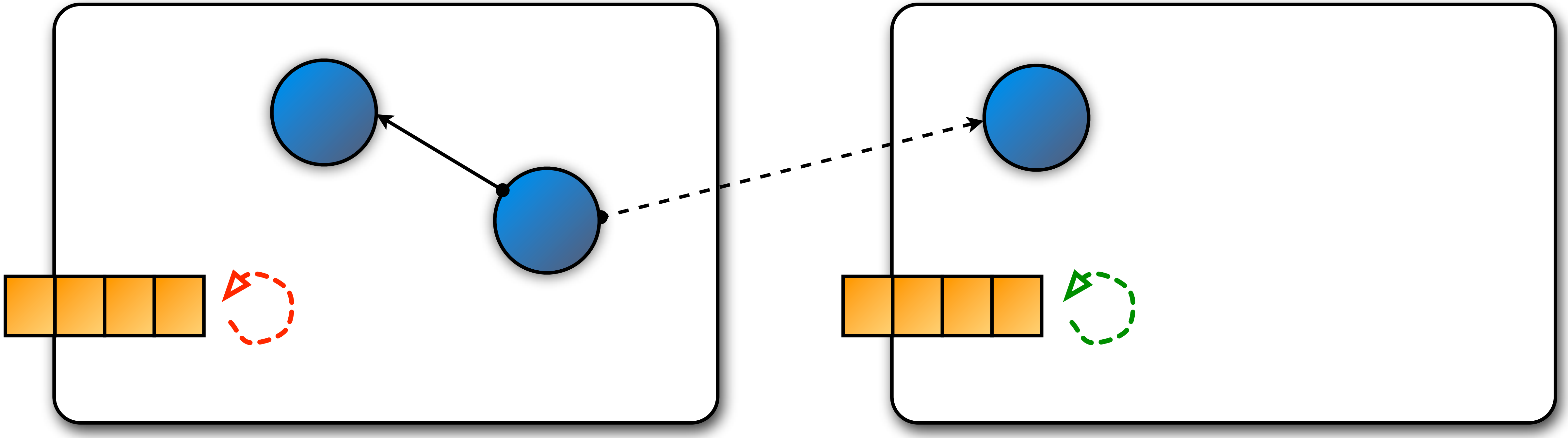
- AmbientTalk programs are event loops
- They react to events from the outside world
- They communicate asynchronously



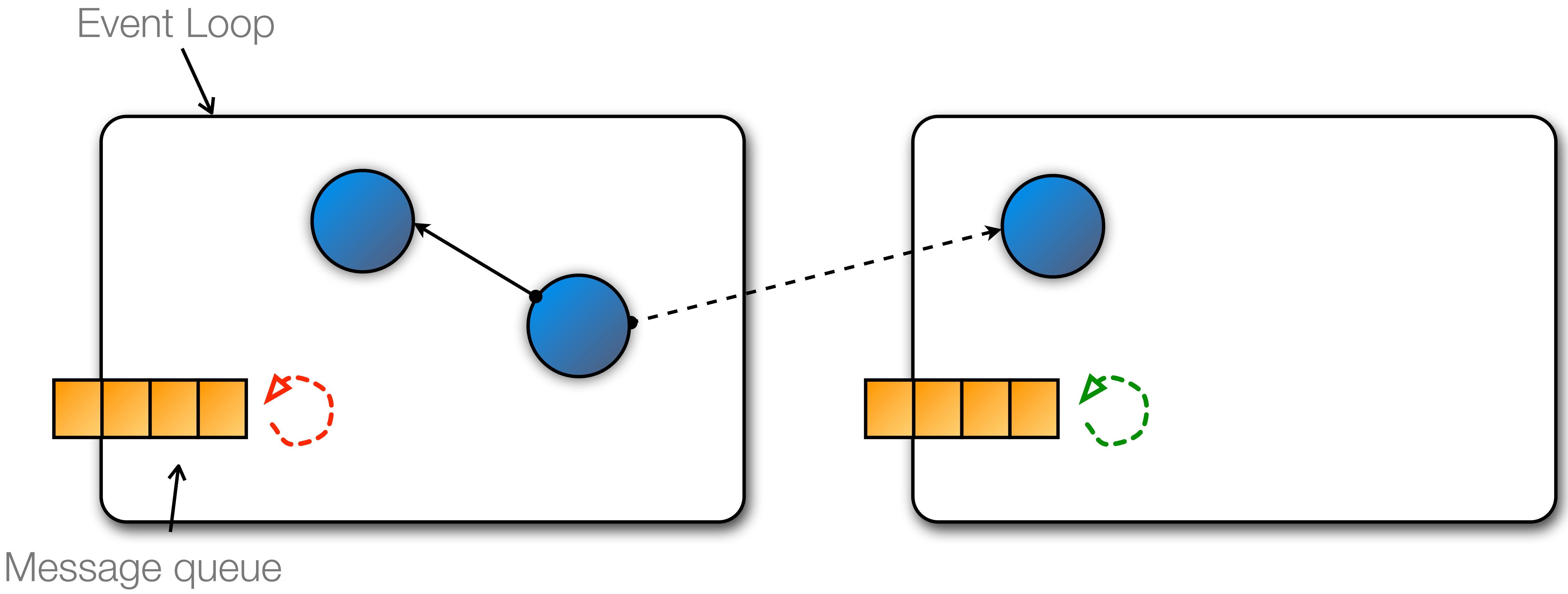
Examples of event loops

- GUI Frameworks (e.g. Java AWT)
- Highly interactive applications (e.g. games)
- IPC in Operating Systems
- Discrete Event Modelling (e.g. simulations)
- Web servers

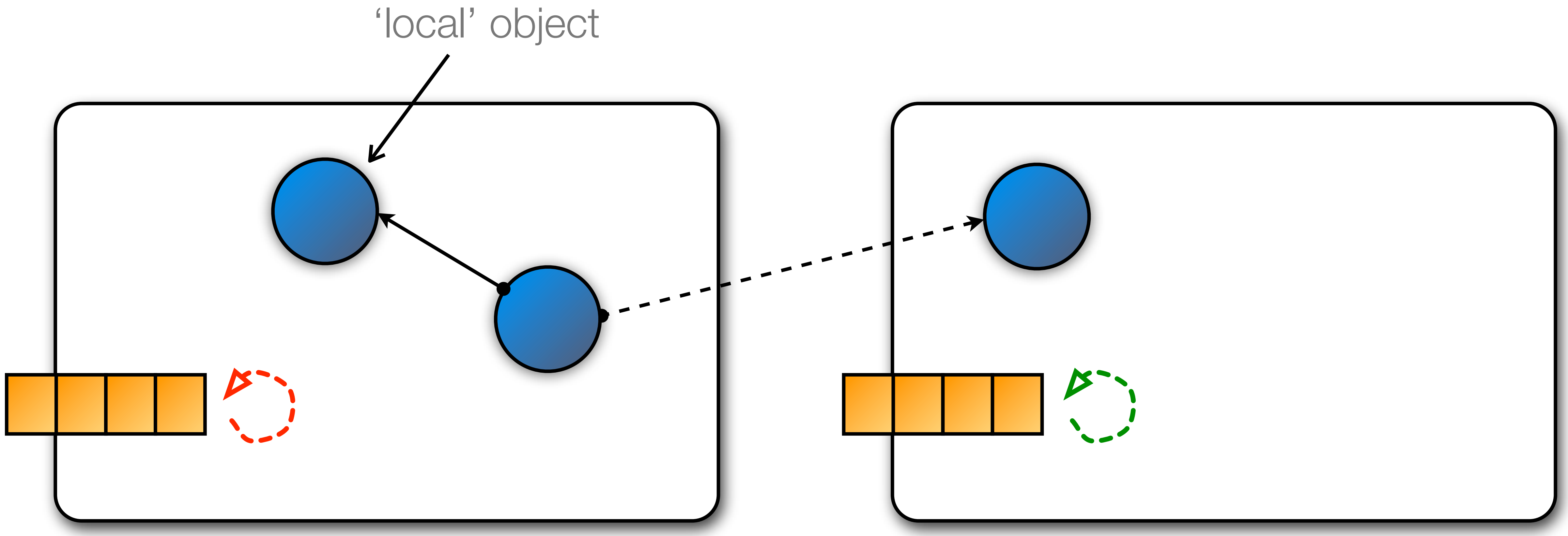
Event Loop Concurrency in AmbientTalk



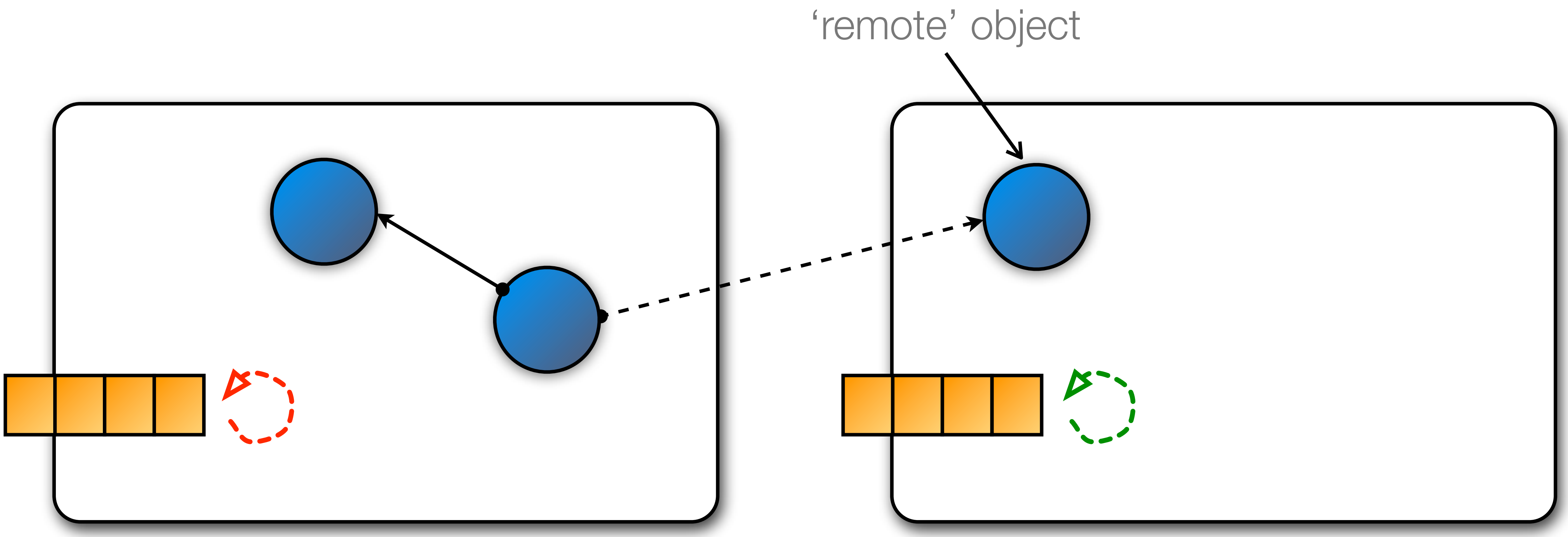
Event Loop Concurrency in AmbientTalk



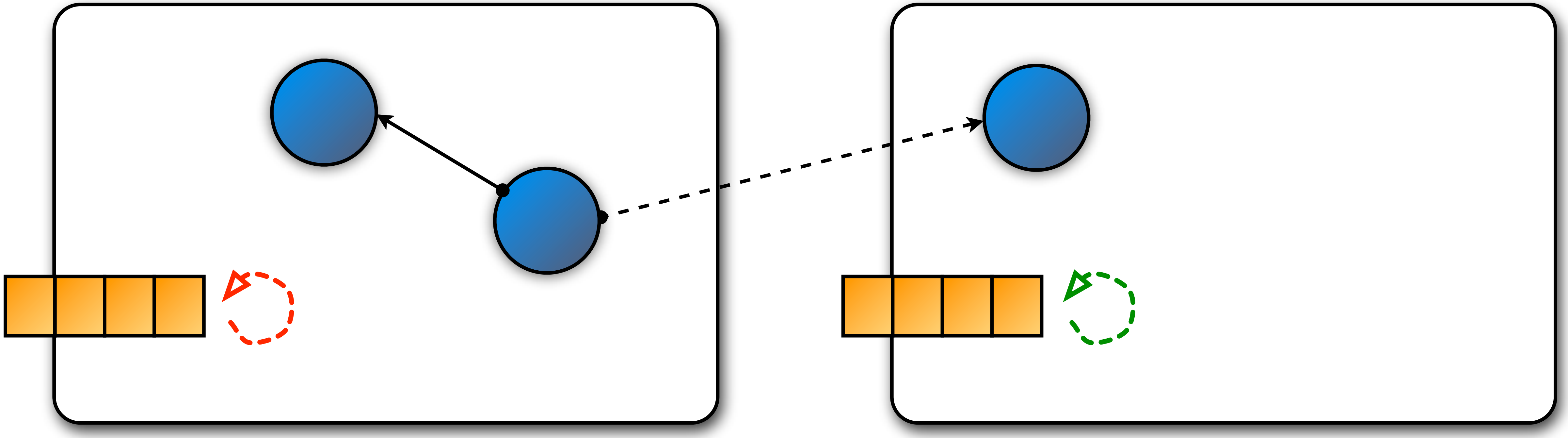
Event Loop Concurrency in AmbientTalk



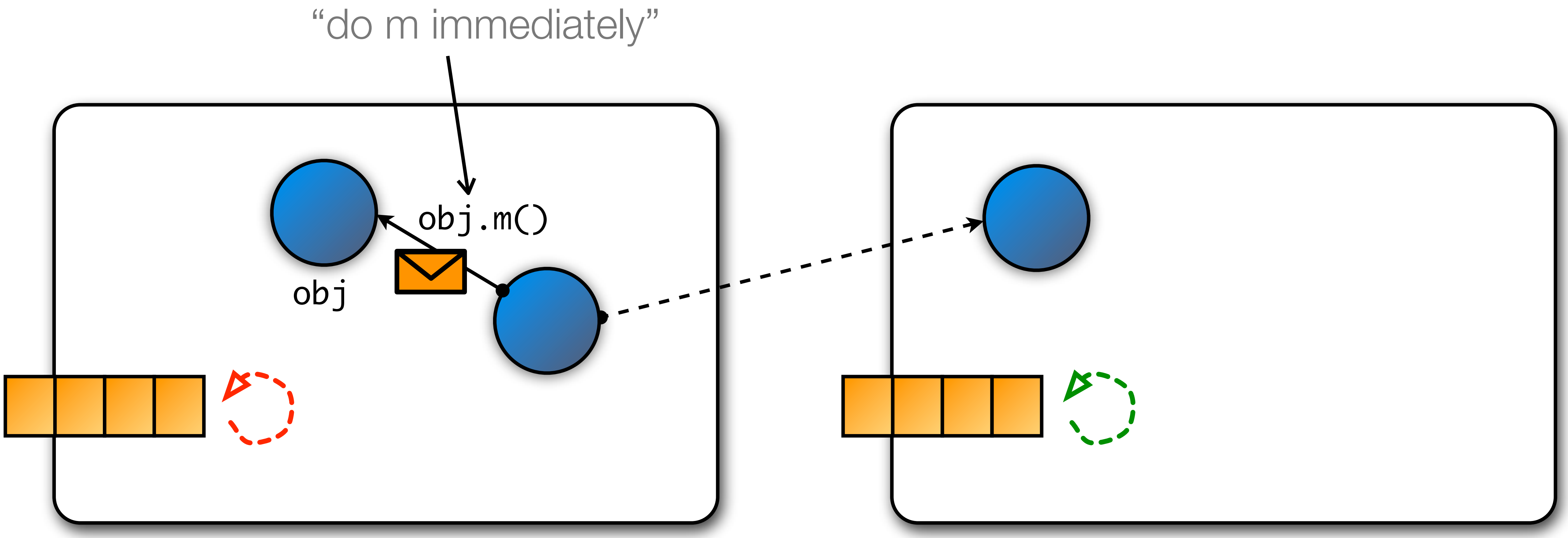
Event Loop Concurrency in AmbientTalk



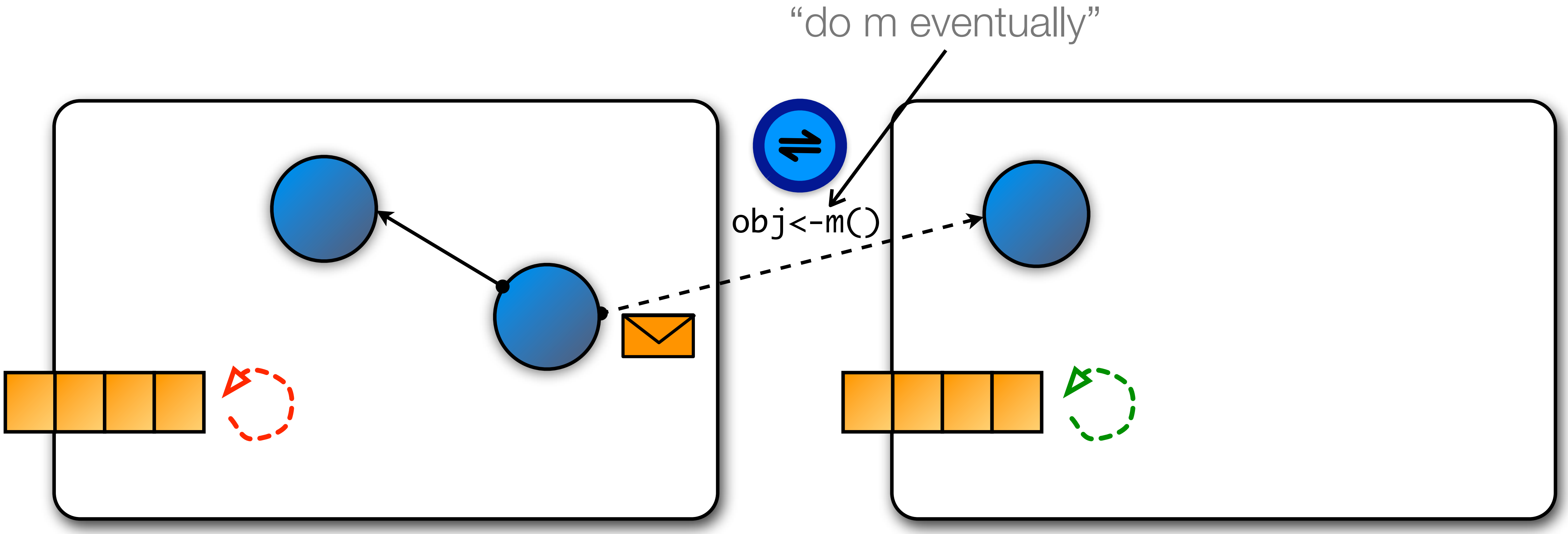
Event Loop Concurrency in AmbientTalk



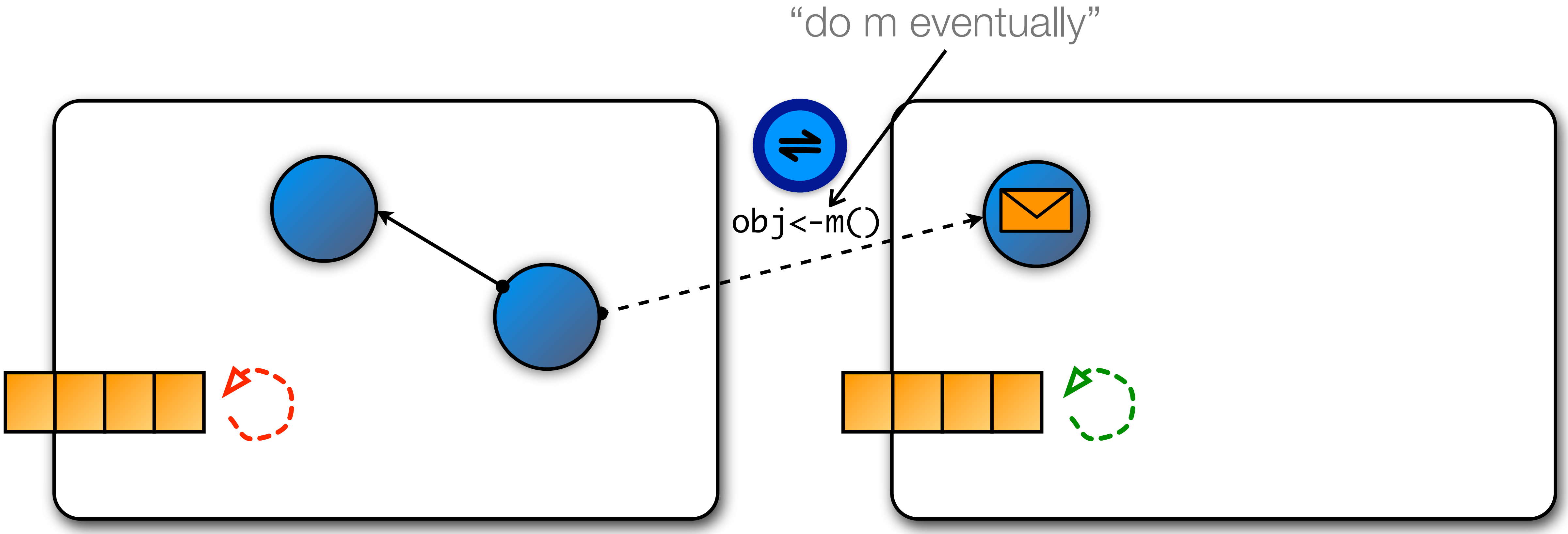
Event Loop Concurrency in AmbientTalk



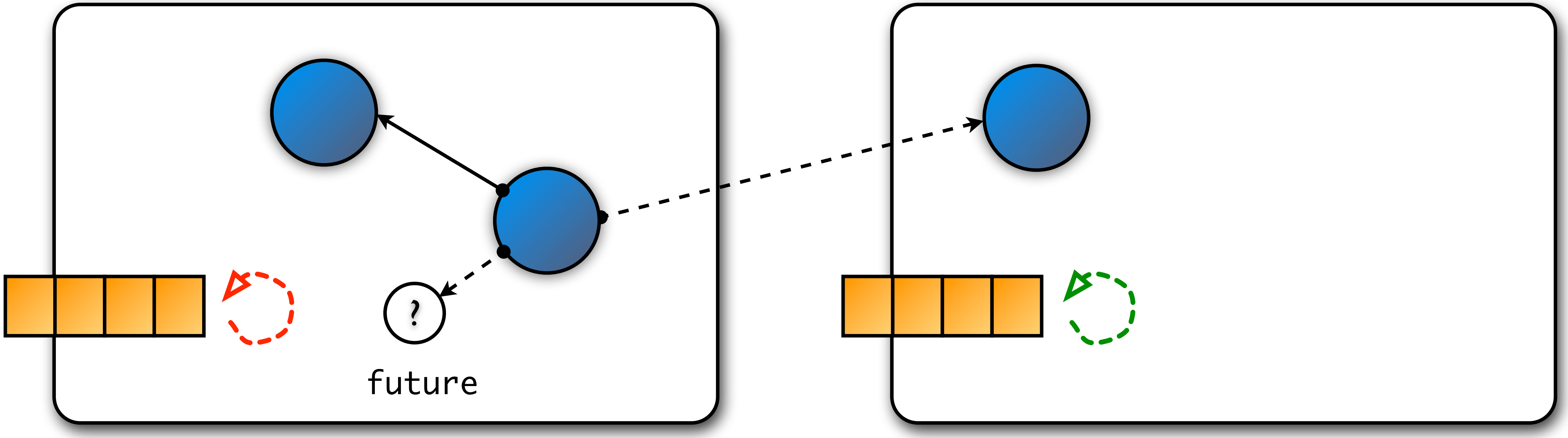
Event Loop Concurrency in AmbientTalk



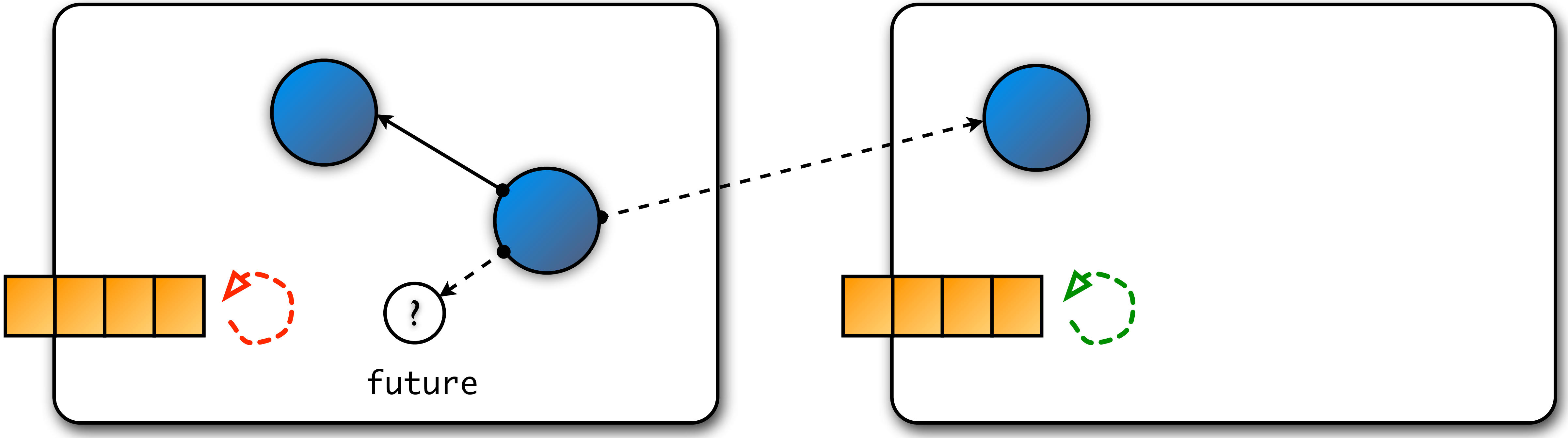
Event Loop Concurrency in AmbientTalk



Event Loop Concurrency in AmbientTalk

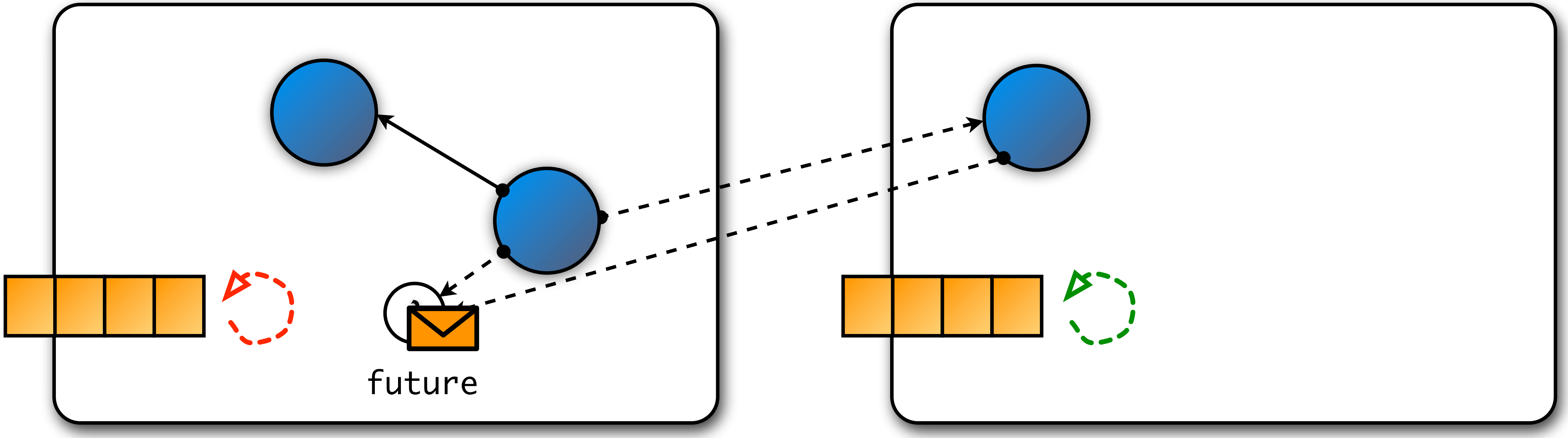


Event Loop Concurrency in AmbientTalk



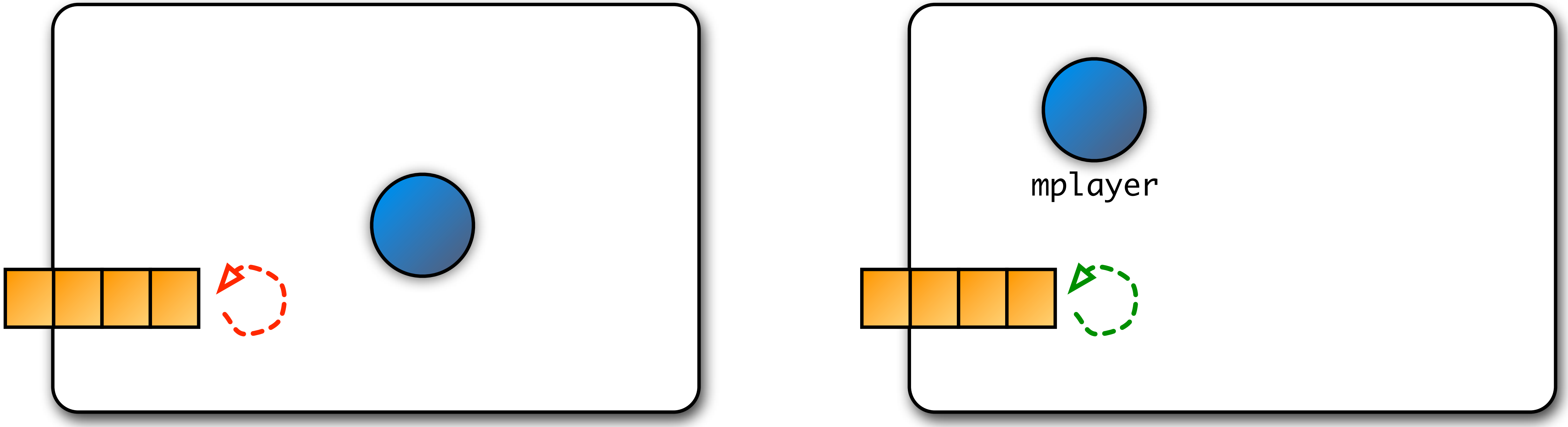
```
when: future becomes: { |value|
  // process reply
}
```

Event Loop Concurrency in AmbientTalk

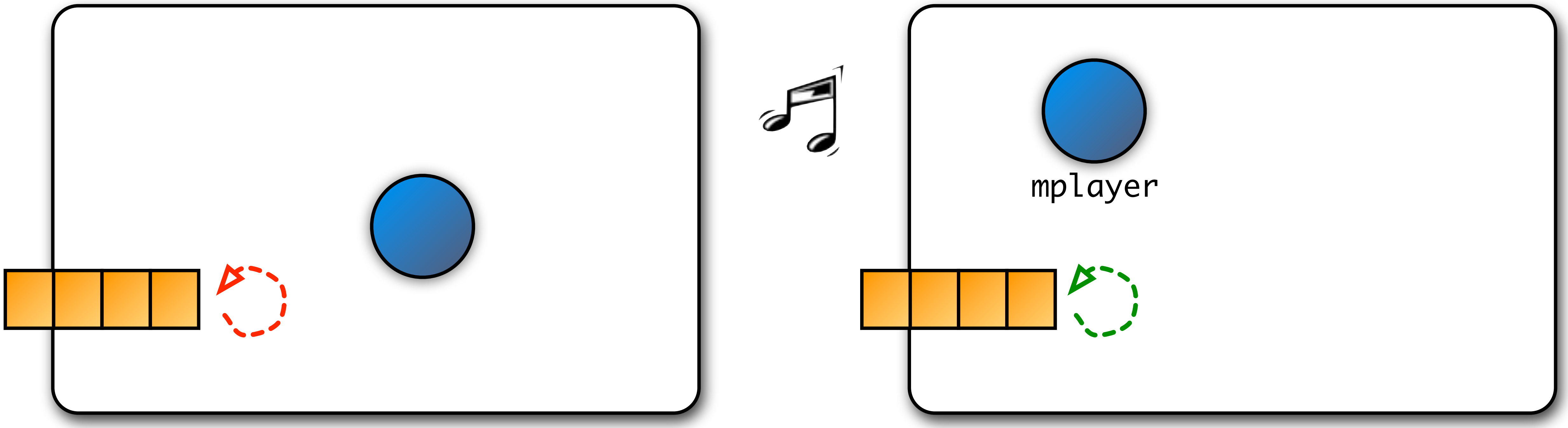


```
when: future becomes: { |value|
  // process reply
}
```

Exporting & discovering objects



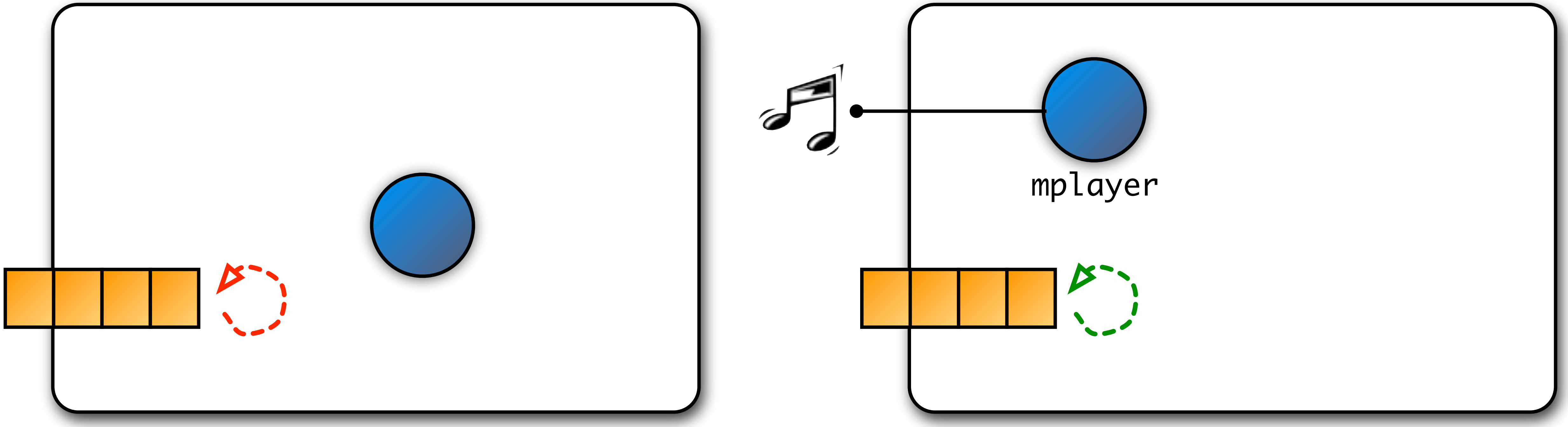
Exporting & discovering objects



`deftype MusicPlayer`

`deftype MusicPlayer`

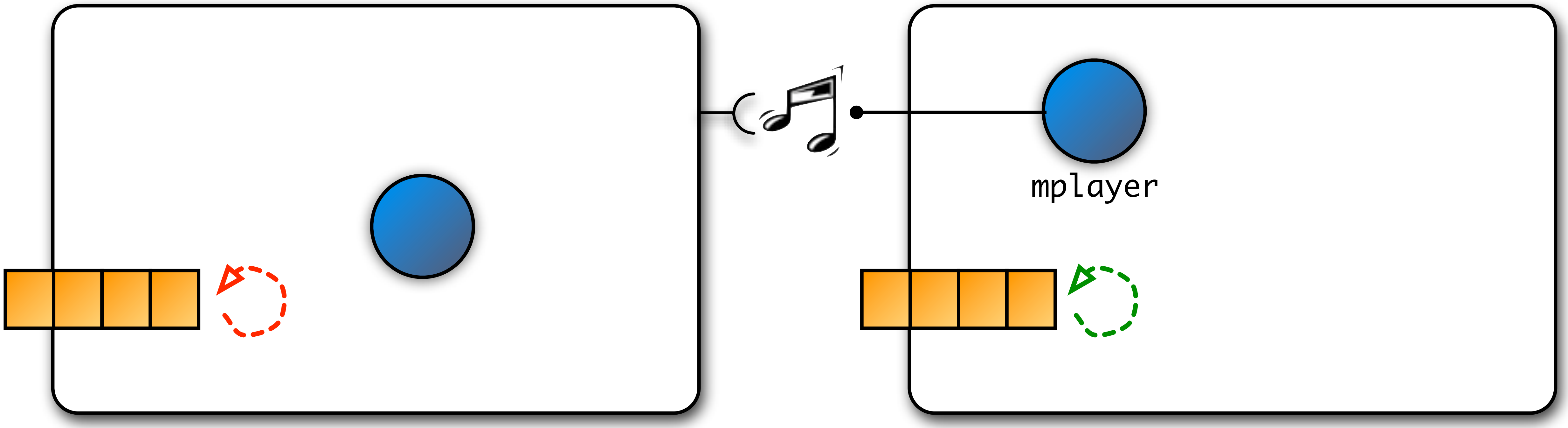
Exporting & discovering objects



```
deftype MusicPlayer
```

```
deftype MusicPlayer

export: mplayer as: MusicPlayer
```

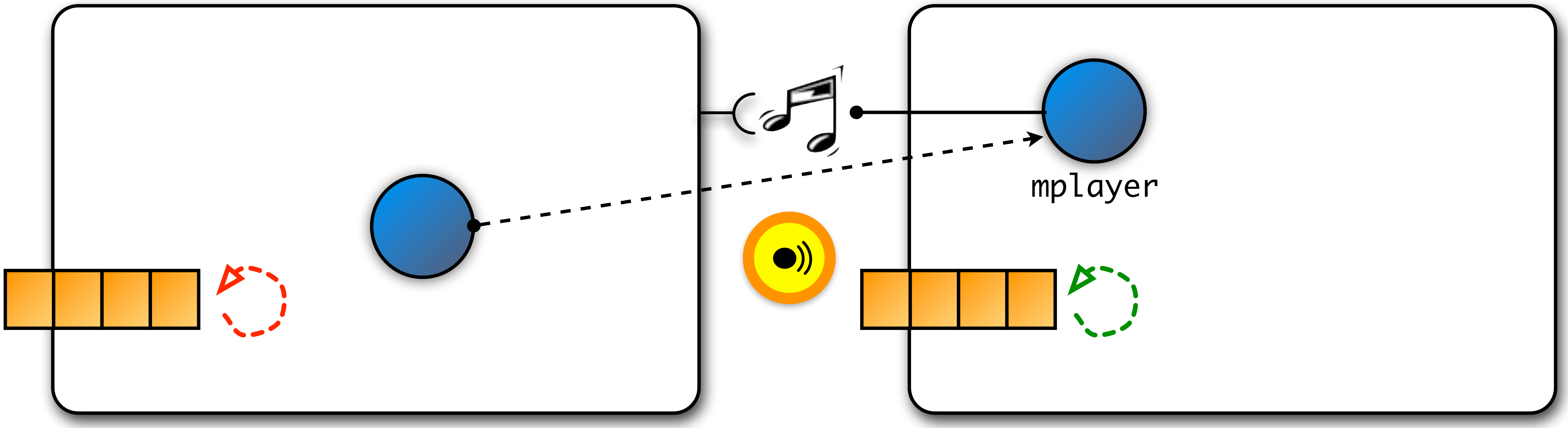
Exporting & discovering objects



```
deftype MusicPlayer                     deftype MusicPlayer

                                        export: mplayer as: MusicPlayer
```

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

Exporting & discovering objects

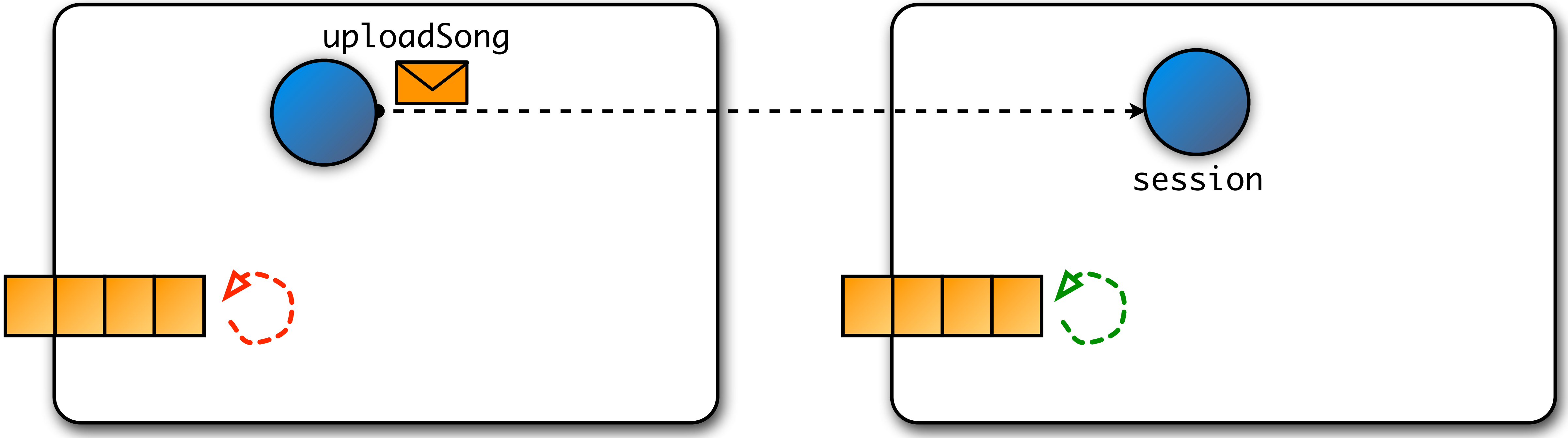


```
deftype MusicPlayer
```

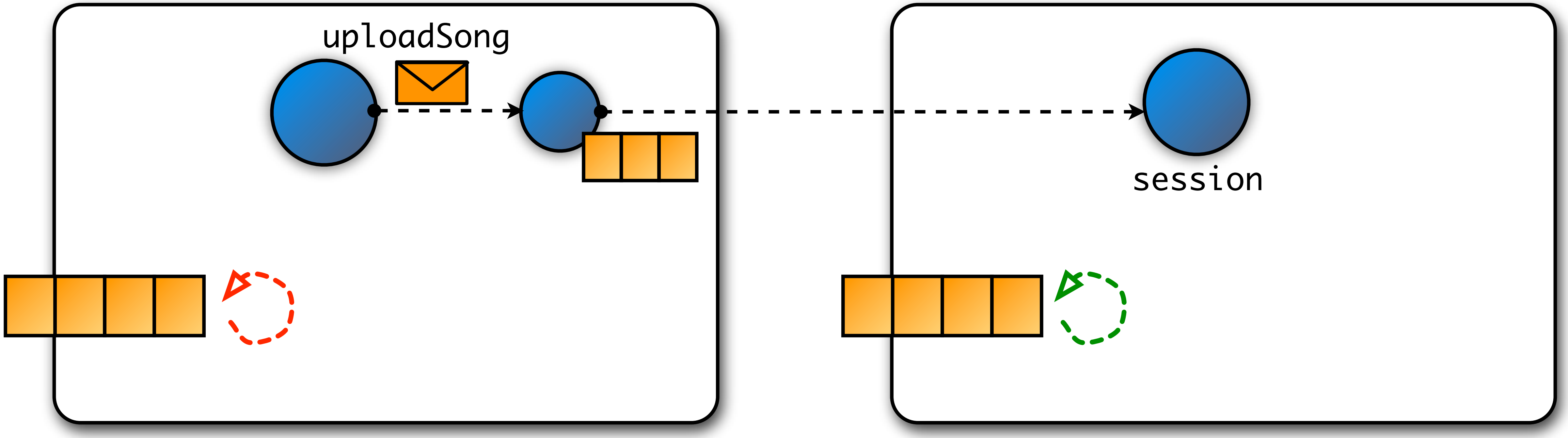
```
deftype MusicPlayer

export: mplayer as: MusicPlayer
```

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

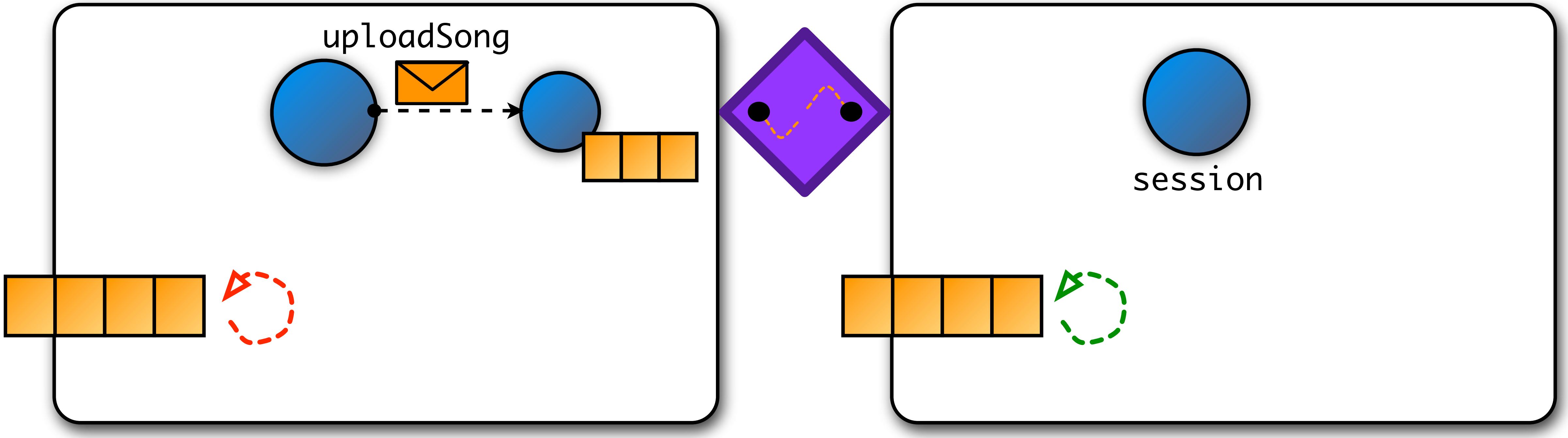
Far References



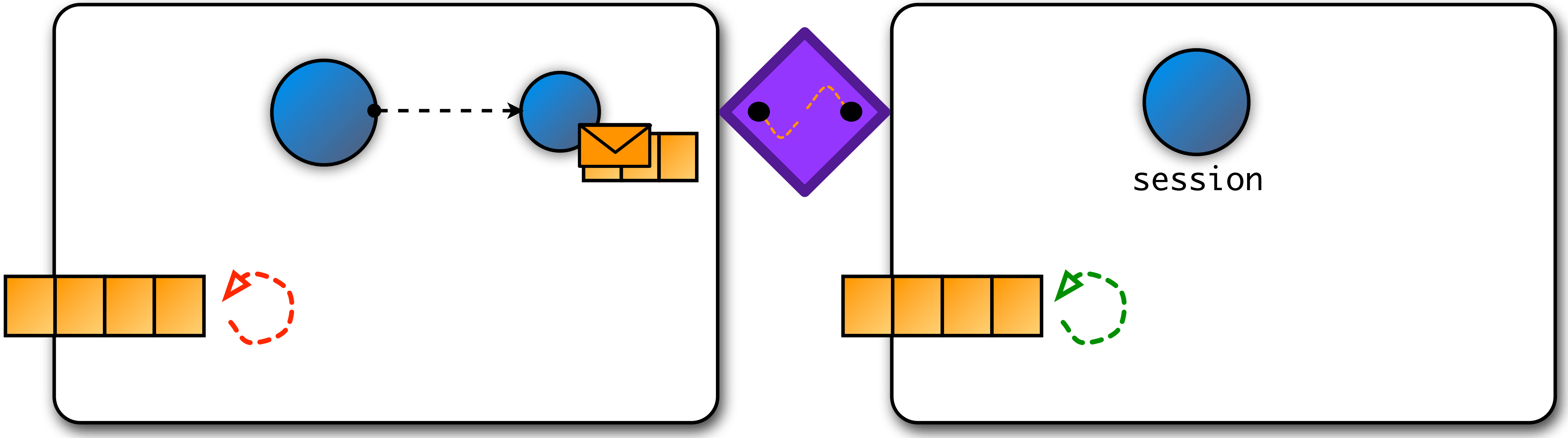
Far References



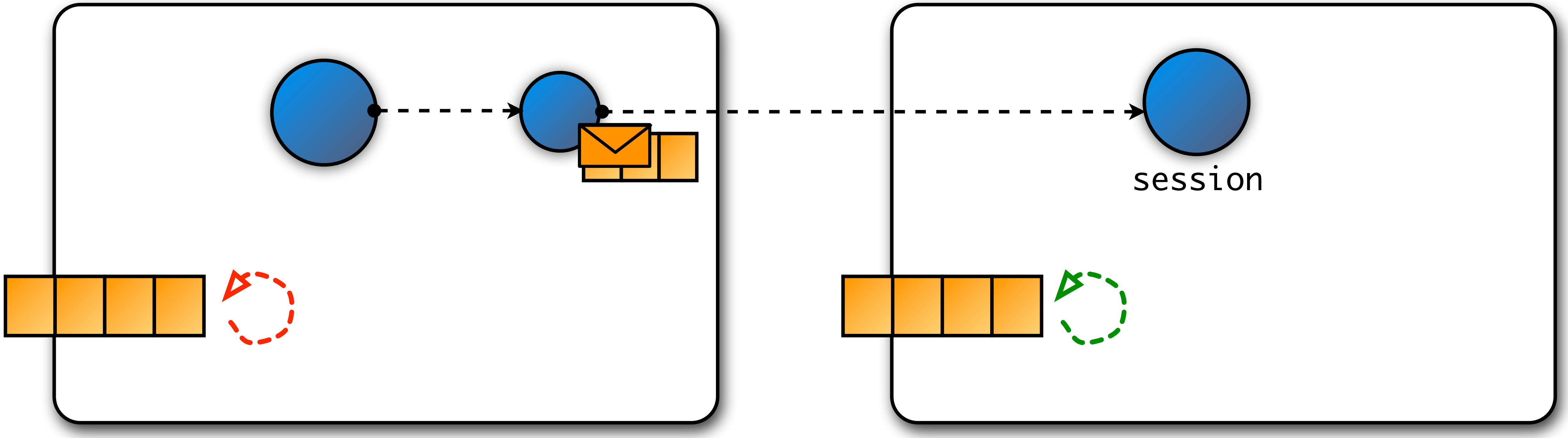
Far References



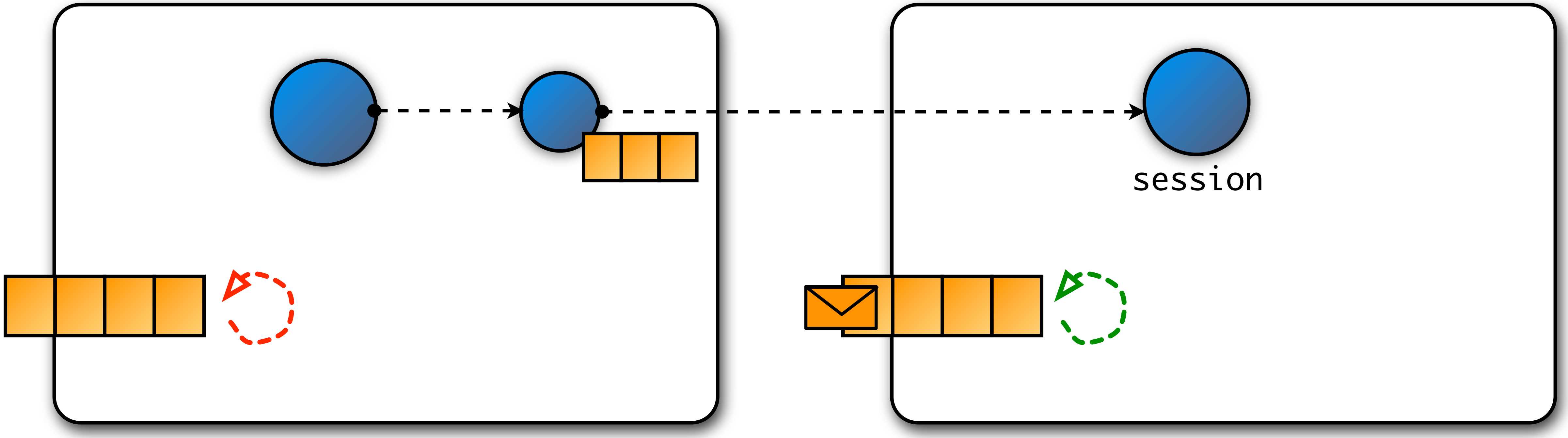
Far References



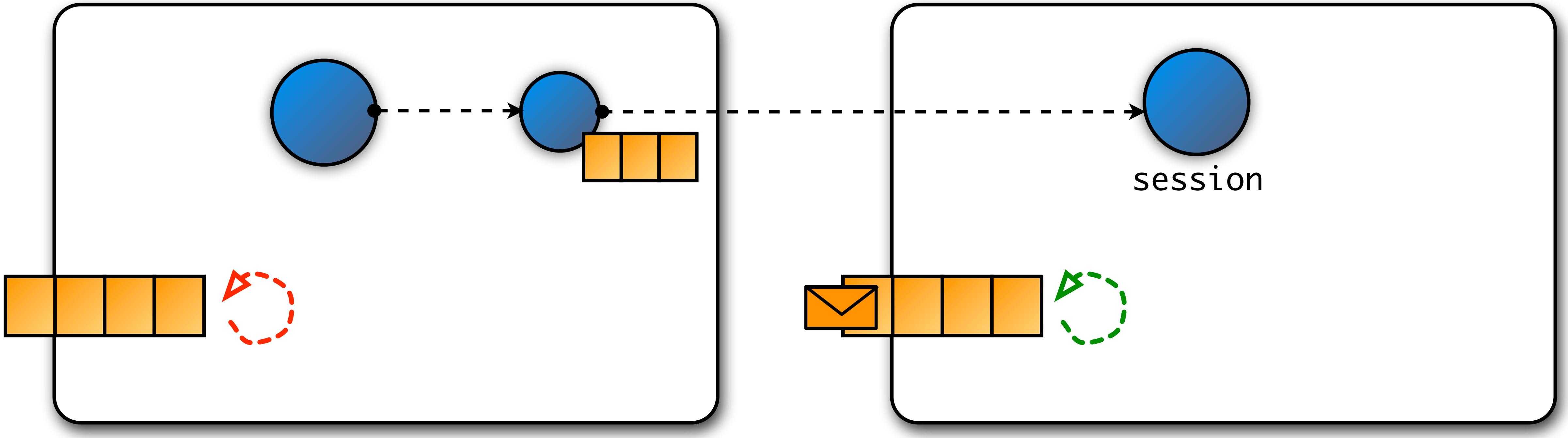
Far References



Far References



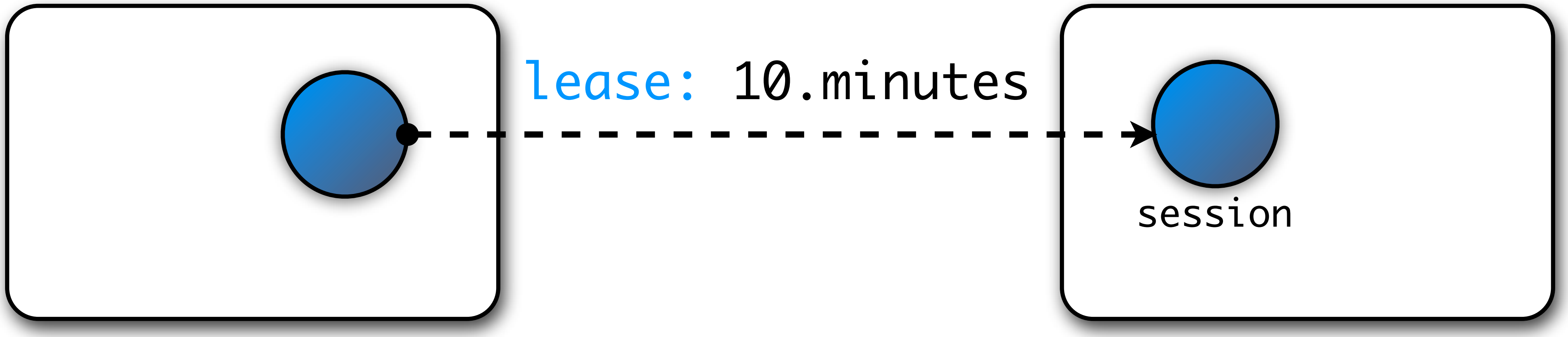
Far References

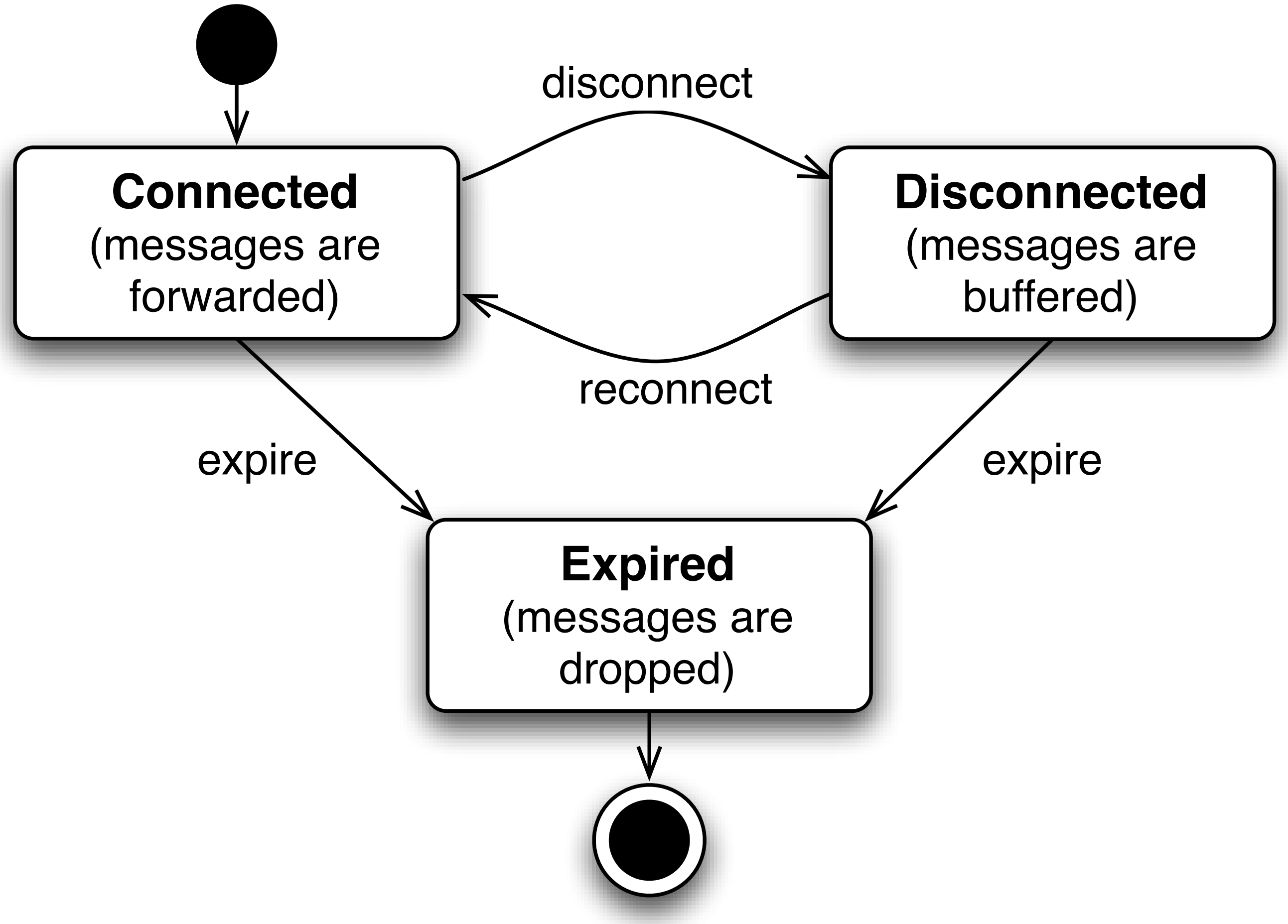


✖

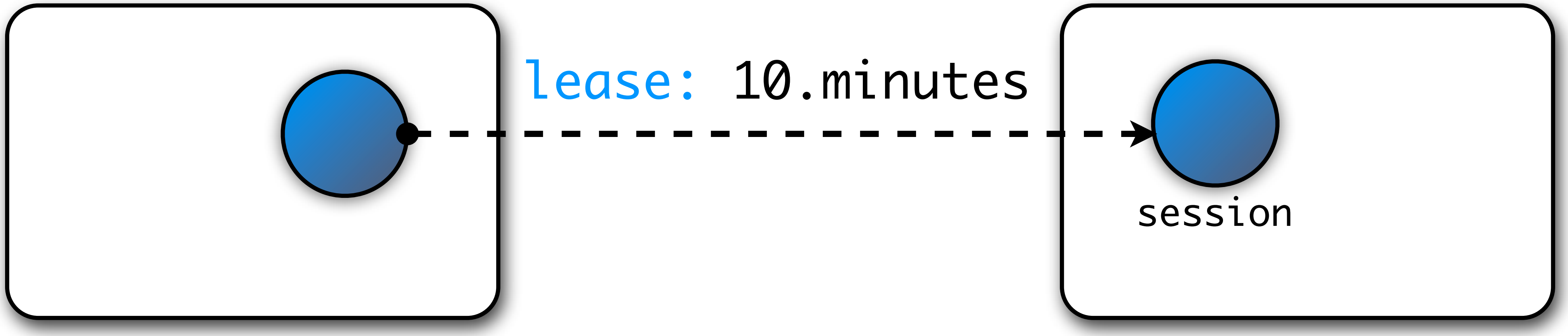
```
when: session<-uploadSong(s)@Due(timeout) becomes: { |ack|
  // continue exchange
} catch: TimeoutException using: { |e|
  // stop exchange
}
```

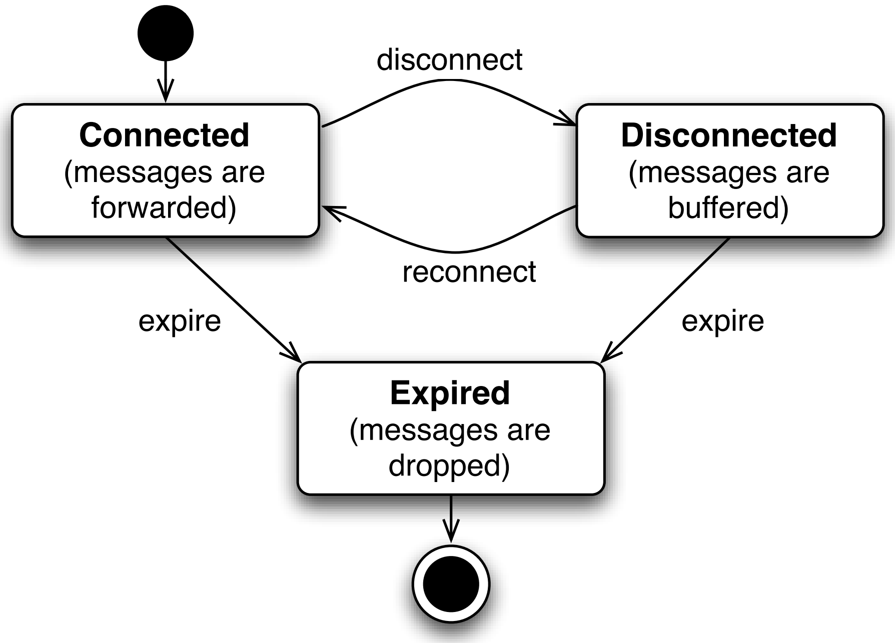
Leasing


lease: 10.minutes
session

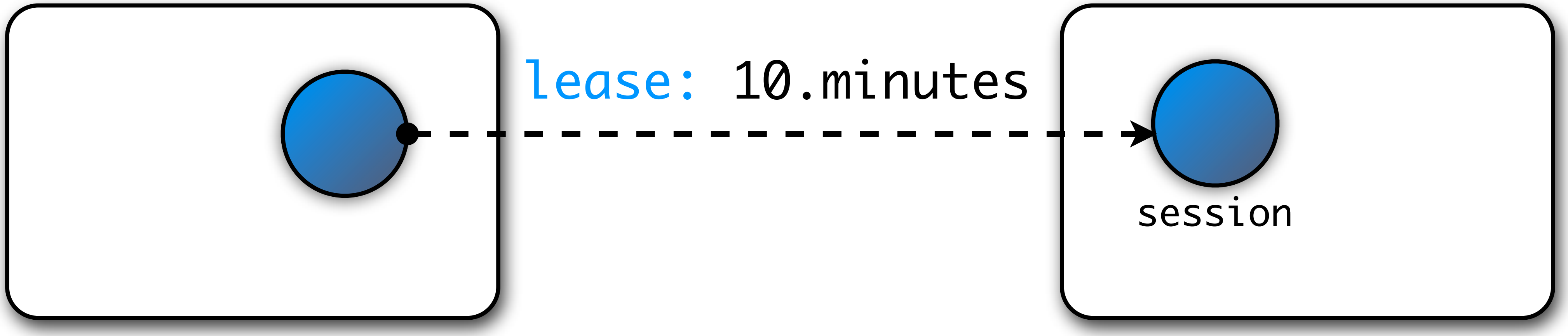

disconnect
Connected (messages are forwarded)
Disconnected (messages are buffered)
reconnect
expire
expire
Expired (messages are dropped)

Reacting to failures





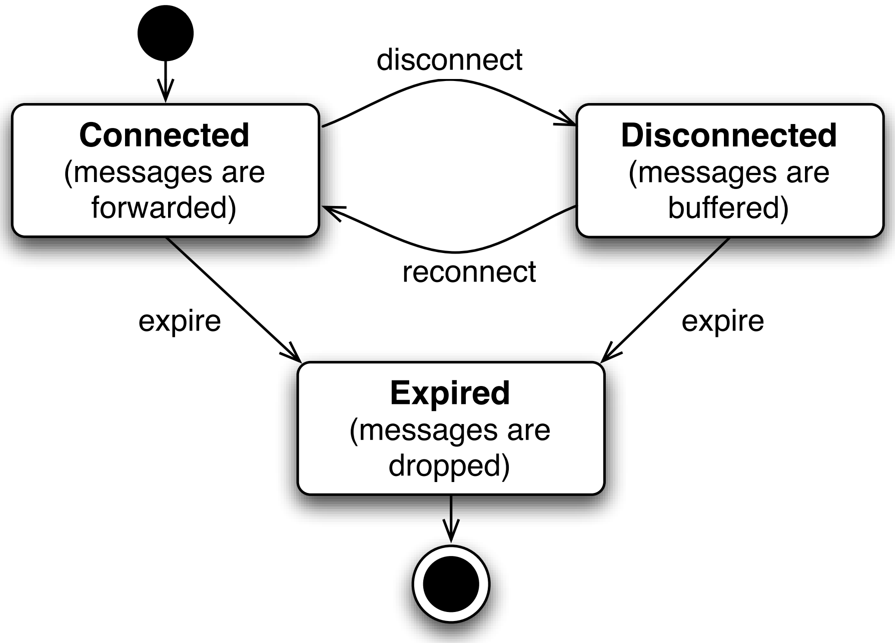
Reacting to failures



lease: 10.minutes

session

```
when: session disconnected: {
  // pause transmission
}
```



Connected
(messages are forwarded)

Disconnected
(messages are buffered)

Expired
(messages are dropped)

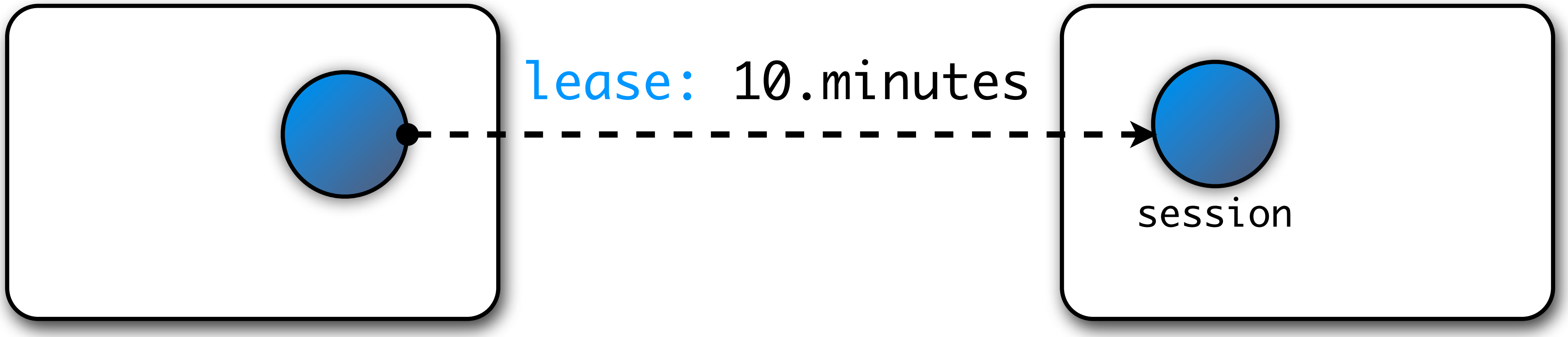
disconnect

reconnect

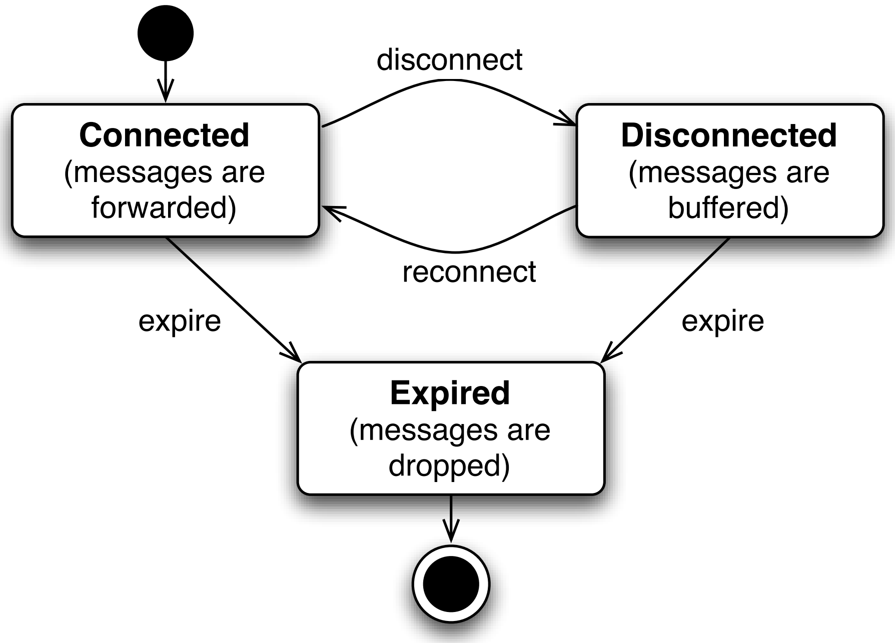
expire

expire

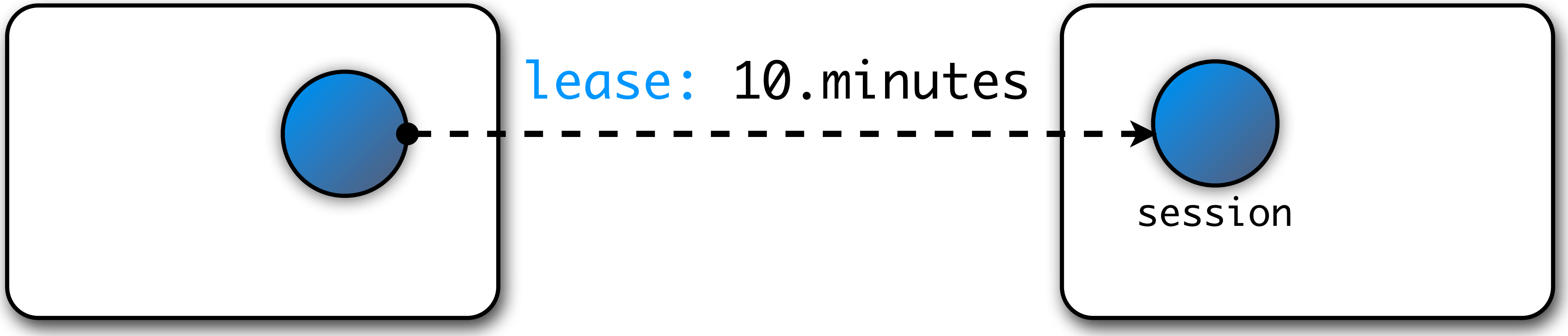
Reacting to failures



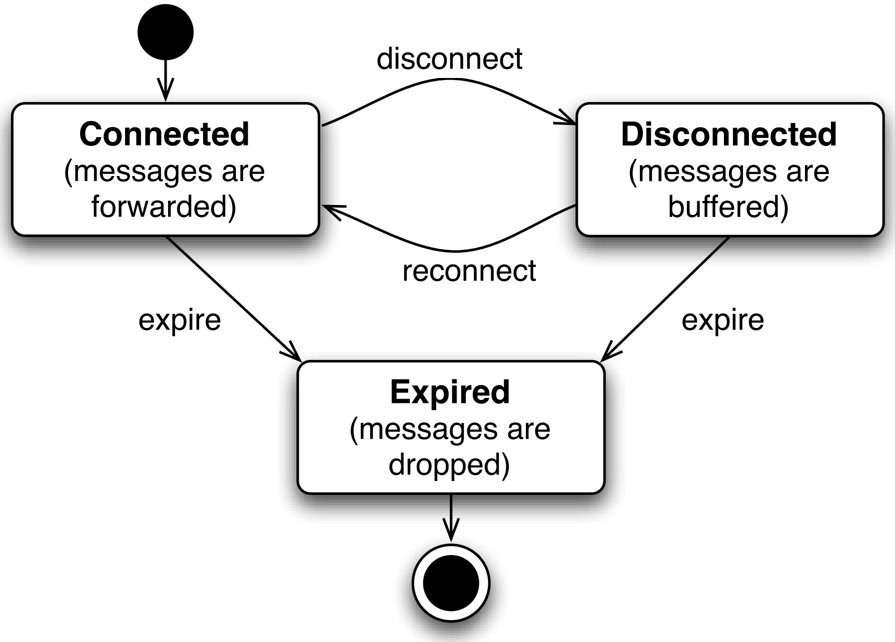
```
when: session disconnected: {
  // pause transmission
}
when: session reconnected: {
  // resume transmission
}
```



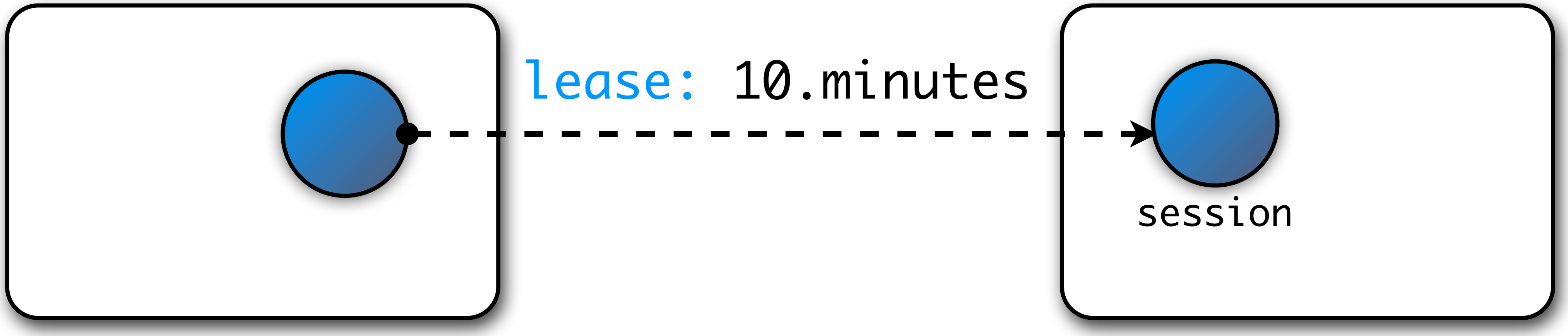
Reacting to failures



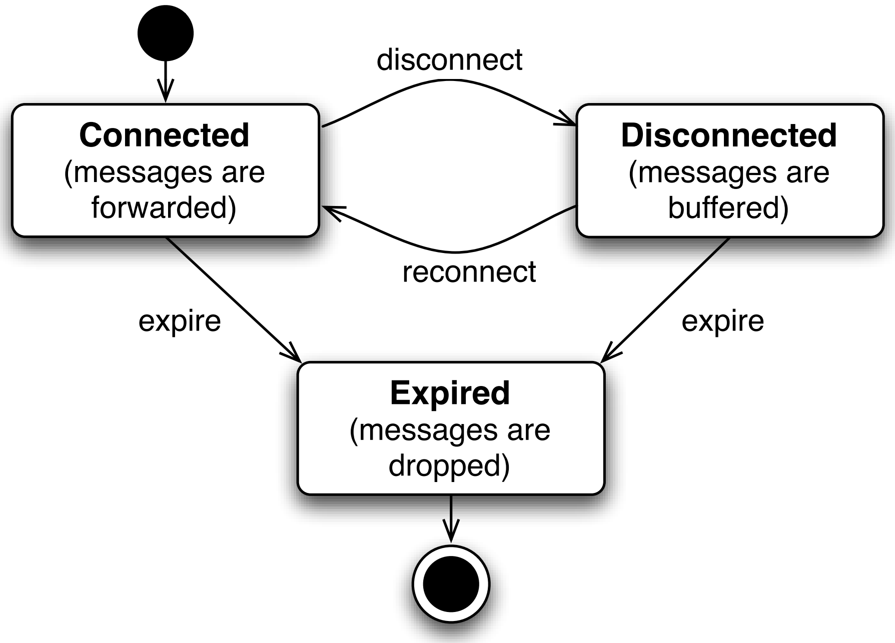
```
when: session disconnected: {
  // pause transmission
}
when: session reconnected: {
  // resume transmission
}
when: session expired: {
  // stop transmission
}
```



Reacting to failures



```
when: session disconnected: {
  // pause transmission
}
when: session reconnected: {
  // resume transmission
}
when: session expired: {
  // stop transmission
}
```



```
when: session expired: {
  // clean up resources
}
```

Event Notifications

when: type discovered: { |obj| ... }	whenever: type discovered: { |obj| ... }
when: obj disconnected: { ... }	whenever: obj disconnected: { ... }
when: obj reconnected: { ... }	whenever: obj reconnected: { ... }
when: obj expired: { ... }	
when: 5.minutes elapsed: { ... }	whenever: 5.minutes elapsed: { ... }
when: future becomes: { |result| ... }	

Event Notifications

Discovery

```
when: type discovered: { |obj| ... }          whenever: type discovered: { |obj| ... }

when: obj disconnected: { ... }               whenever: obj disconnected: { ... }

when: obj reconnected: { ... }                whenever: obj reconnected: { ... }

when: obj expired: { ... }

when: 5.minutes elapsed: { ... }              whenever: 5.minutes elapsed: { ... }

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

Event Notifications

```
when: type discovered: { |obj| ... }          whenever: type discovered: { |obj| ... }
```

Failure Handling

```
when: obj disconnected: { ... }               whenever: obj disconnected: { ... }

when: obj reconnected: { ... }                whenever: obj reconnected: { ... }

when: obj expired: { ... }
```

```
when: 5.minutes elapsed: { ... }              whenever: 5.minutes elapsed: { ... }

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

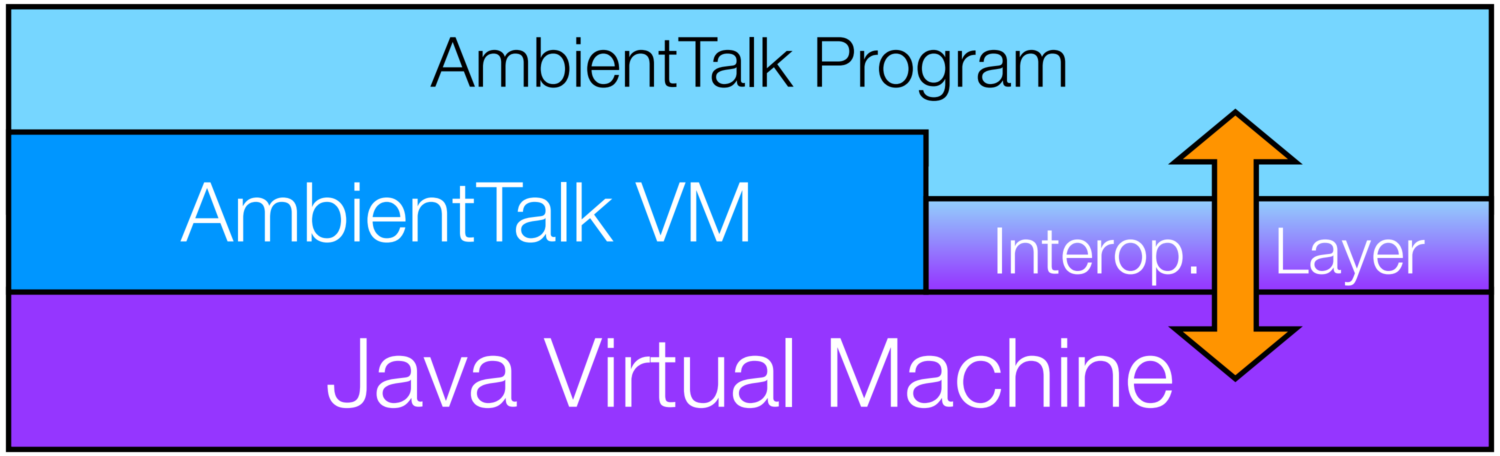
Event Notifications

```
when: type discovered: { |obj| ... }      whenever: type discovered: { |obj| ... }

when: obj disconnected: { ... }           whenever: obj disconnected: { ... }

when: obj reconnected: { ... }            whenever: obj reconnected: { ... }

when: obj expired: { ... }
```

Synchronisation

```
when: 5.minutes elapsed: { ... }          whenever: 5.minutes elapsed: { ... }

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

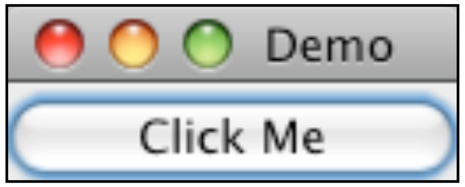
Language Interoperability



- Scripting language on top of the JVM, cfr. JRuby, Jython, Groovy, ...
- AmbientTalk can use Java libraries, Java can use AmbientTalk scripts

Batteries Included

```
def Button := jlobby.java.awt.Button;
def button := Button.new(“Click Me”);
button.addActionListener(object: {
  def actionPerformed(actionEvent) {
    println(“button clicked”);
  }
});
button.setVisible(true);
```



Reflection

- AmbientTalk code can introspect and change behavior of objects and actors

```
def makeSong(artist, title) {
  object: {
    def printArtist() {
      if: (artist == nil) then: {
        “unknown artist”;
      } else: {
        artist;
      }
    }
  }
}

def song := makeSong(“U2”, “One”);
song.printArtist();
```

Reflection

- AmbientTalk code can introspect and change behavior of objects and actors

```
def makeSong(artist, title) {
  object: {
    def printArtist() {
      if: (artist == nil) then: {
        “unknown artist”;
      } else: {
        artist;
      }
    }
  }
}

def song := makeSong(“U2”, “One”);
song.printArtist();
```

```
def mirrorOnSong := (reflect: song);

mirrorOnSong.invoke(song,
  createInvocation(`printArtist, []));

mirrorOnSong.listSlots();

mirrorOnSong.addSlot(slot);

...
```

Implementation

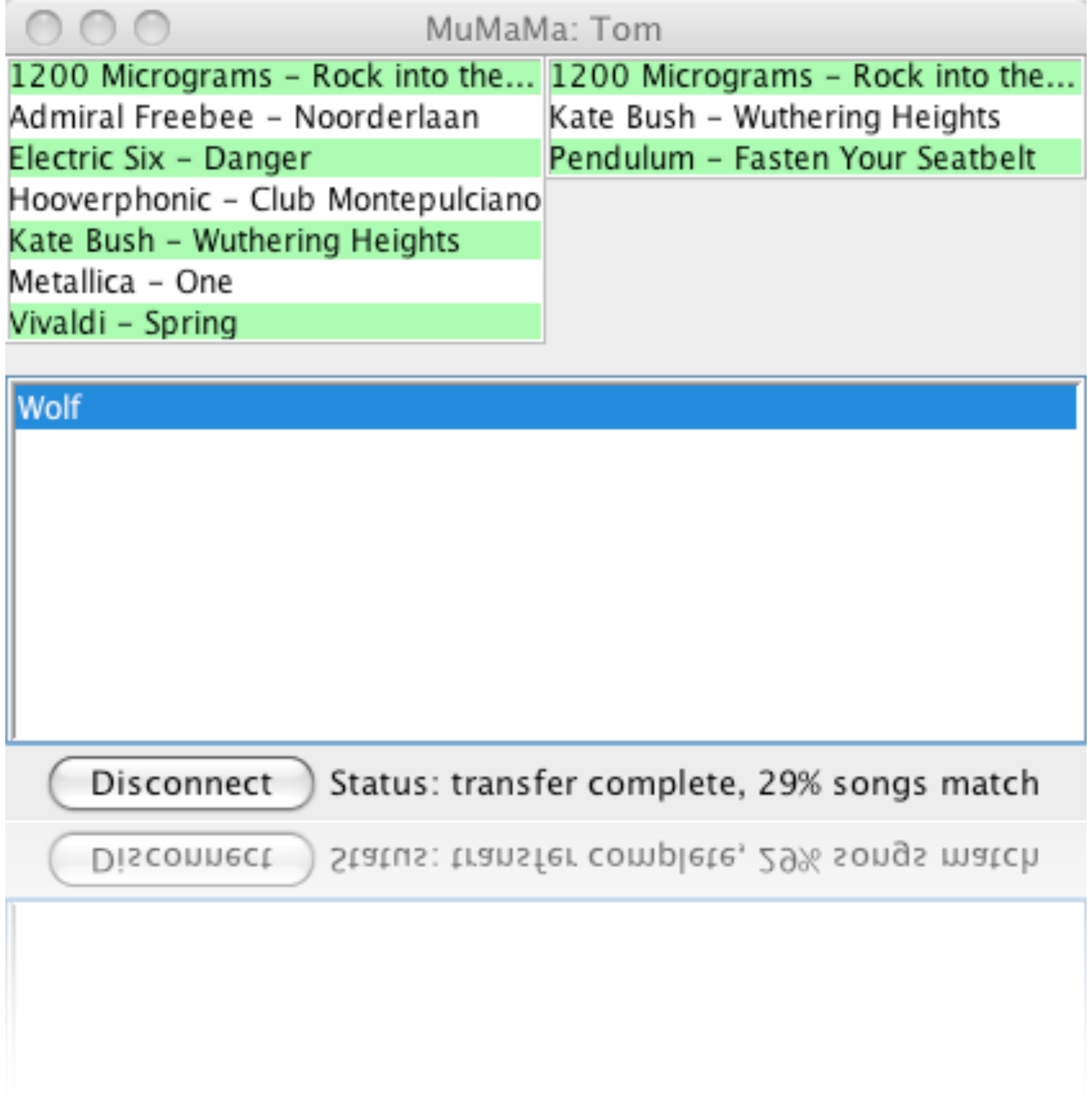
- Interactive interpreter
- ± 17.000 SLOC of Java
- UDP & TCP/IP over WLAN
- Runs on top of J2ME/CDC
  - QTek 9090 SmartPhones
  - HTC Touch Cruise SmartPhones
  - iPhone [in progress]
  - Android G1 [in progress]

Experiments

Demo Applications

Case: Musical Match Maker

- Demo



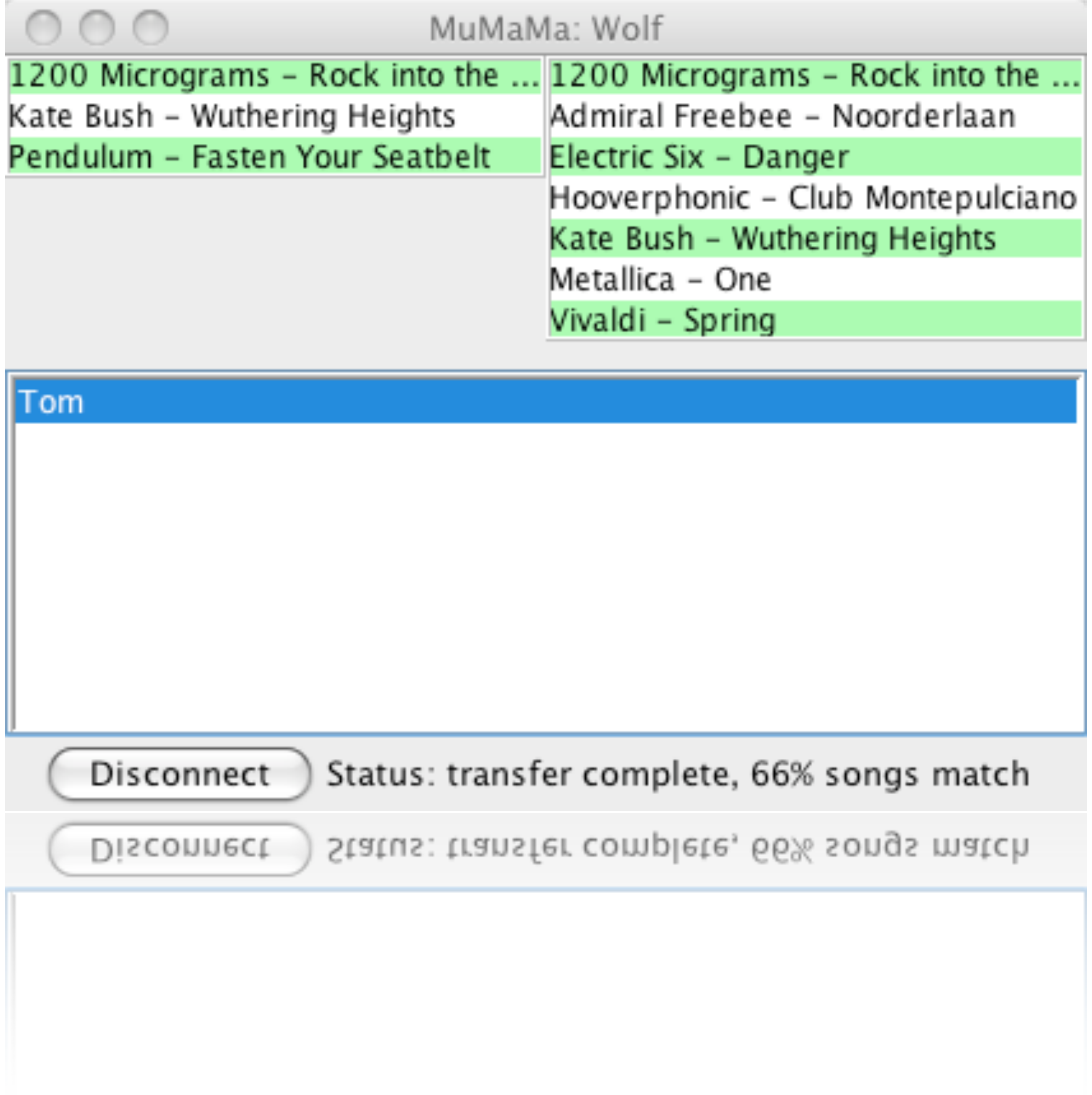
MuMaMa: Tom

1200 Micrograms - Rock into the...
Admiral Freebee - Noorderlaan
Electric Six - Danger
Hooverphonic - Club Montepulciano
Kate Bush - Wuthering Heights
Metallica - One
Vivaldi - Spring

1200 Micrograms - Rock into the...
Kate Bush - Wuthering Heights
Pendulum - Fasten Your Seatbelt

Wolf

Status: transfer complete, 29% songs match



MuMaMa: Wolf

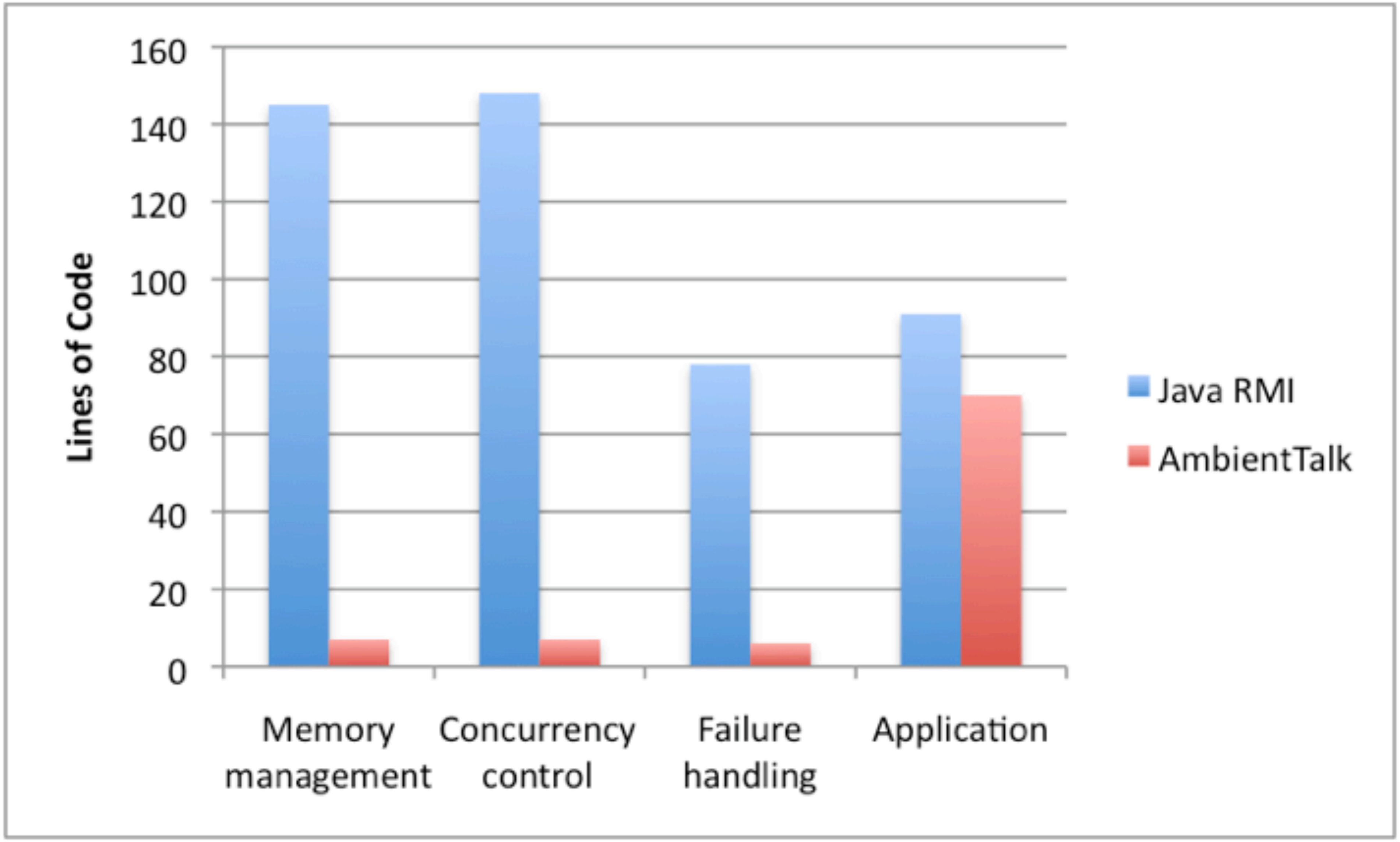
1200 Micrograms - Rock into the ...
Kate Bush - Wuthering Heights
Pendulum - Fasten Your Seatbelt

1200 Micrograms - Rock into the ...
Admiral Freebee - Noorderlaan
Electric Six - Danger
Hooverphonic - Club Montepulciano
Kate Bush - Wuthering Heights
Metallica - One
Vivaldi - Spring

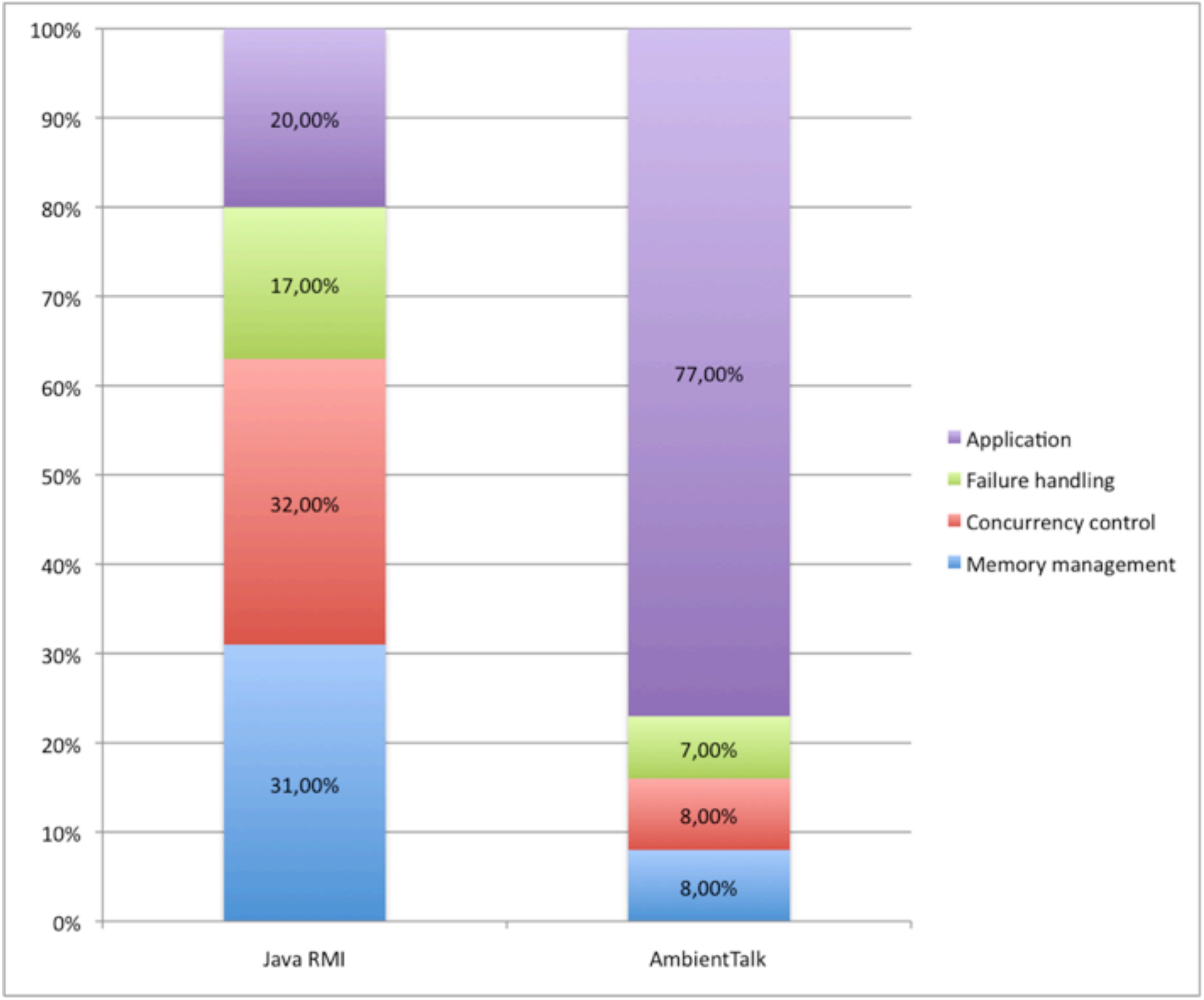
Tom

Status: transfer complete, 66% songs match

Experimental Results



Experimental Results

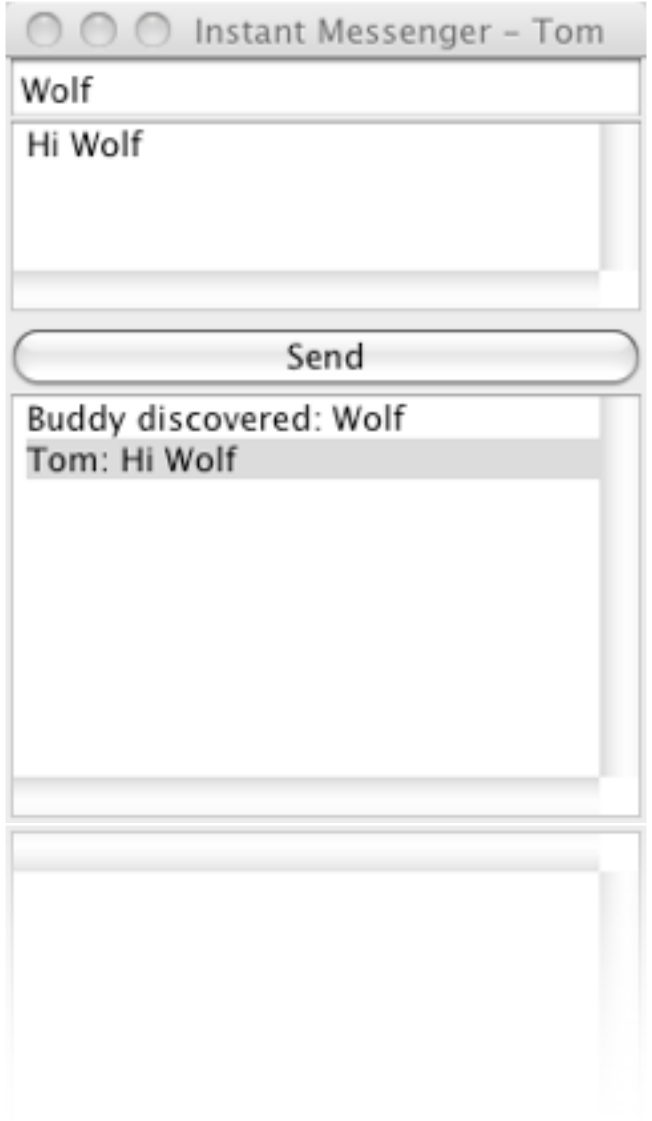


Causes

- Explicit encoding of
  - buffered, asynchronous communication using threads
  - remote messages using objects
  - timeouts using timer threads
  - event notifications (lease expiration & renewal, calls & callbacks) using listeners + event loop threads
- Java RMI does not deal with service discovery

Case: Instant Messenger

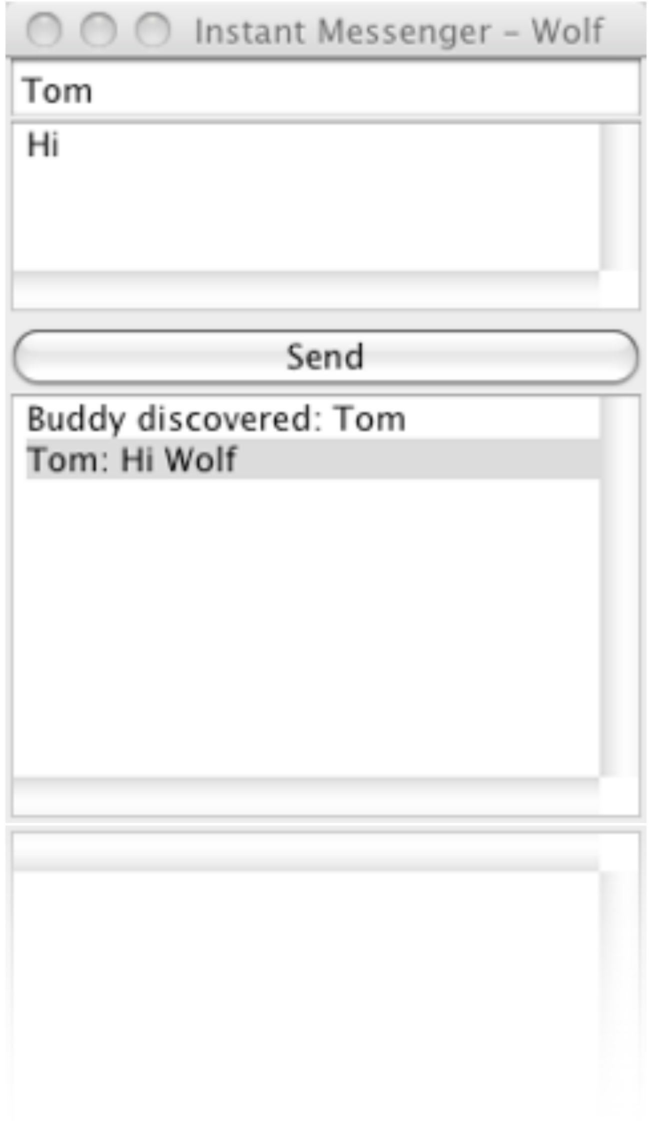
- Demo



Wolf

Hi Wolf

Buddy discovered: Wolf
Tom: Hi Wolf

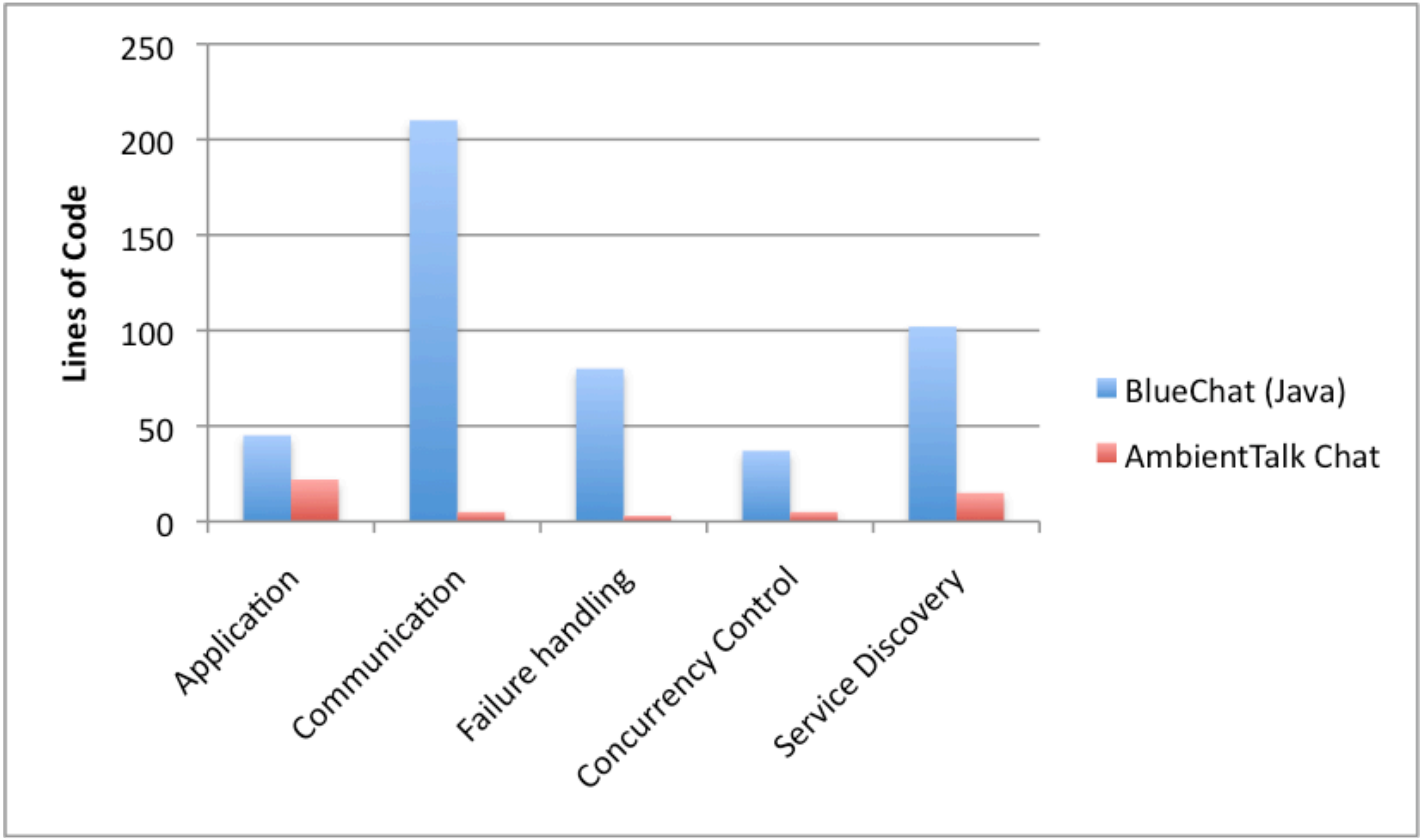


Tom

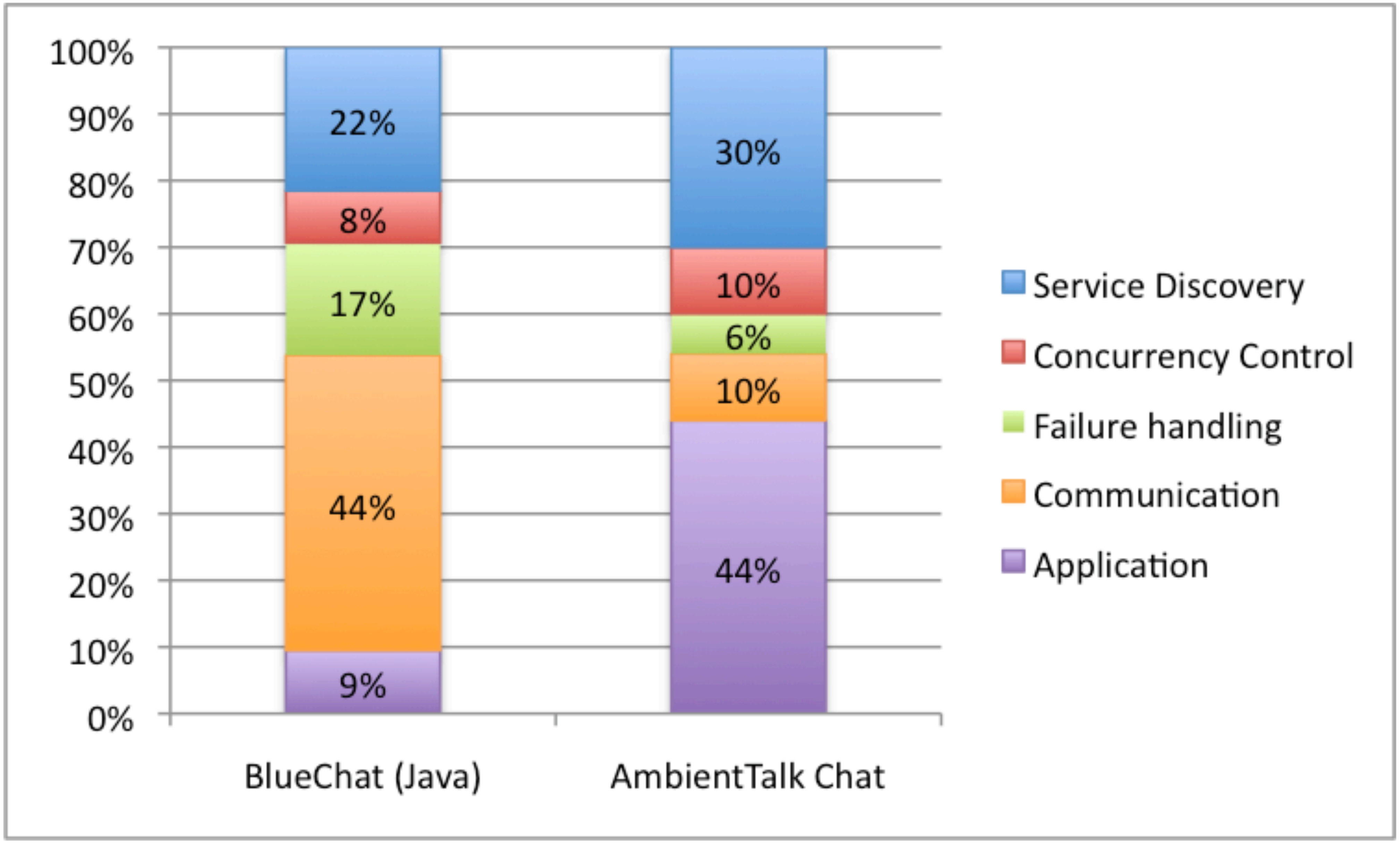
Hi

Buddy discovered: Tom
Tom: Hi Wolf

Chat: Java vs AmbientTalk (LoC)



Chat: Java vs AmbientTalk (%)



A simple application but...

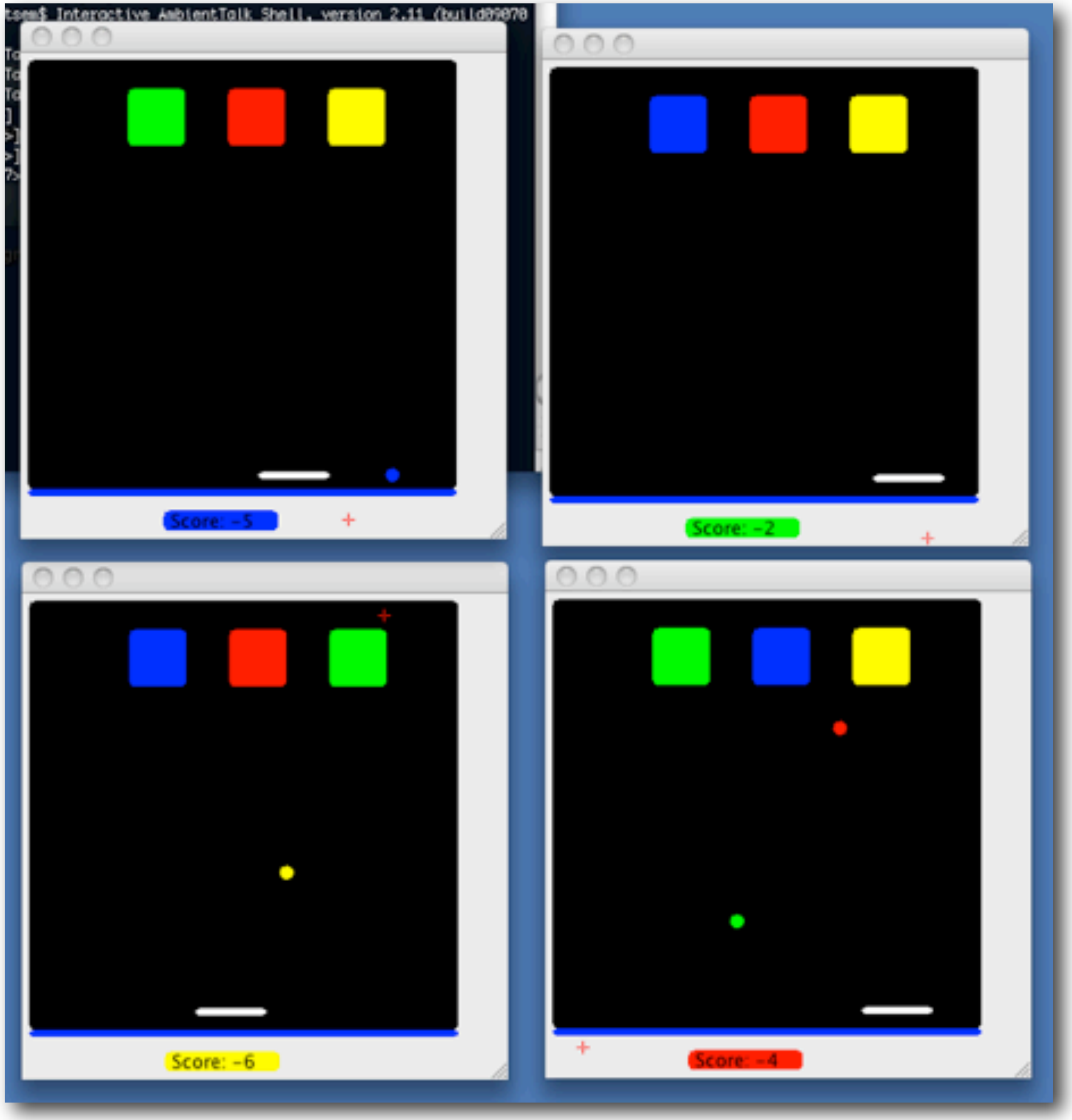
... we do not need to explicitly manage:

- threads & locks
- low-level socket connections
- stubs, skeletons
- name server or lookup service
- timeouts, leasing

AmbientMorphic

- Implementation of the Morphic UI framework from Self

- Demo: PortalPong



```
tsem$ Interactive AmbientTalk Shell, version 2.11 (build09070
```

Conclusion

- Mobile ad hoc networks:
- Ambient-oriented programming:
- AmbientTalk:
  - Scripting language on top of the JVM
  - Reactive, event-driven programs
- Applications: chat, match maker, multiplayer game

Conclusion

- Mobile ad hoc networks: Zero Infrastructure Volatile Connections
- Ambient-oriented programming:
- AmbientTalk:
  - Scripting language on top of the JVM
  - Reactive, event-driven programs
- Applications: chat, match maker, multiplayer game

Conclusion

- Mobile ad hoc networks: Zero Infrastructure Volatile Connections
- Ambient-oriented programming: Peer-to-peer Tolerate disconnections
- AmbientTalk:
  - Scripting language on top of the JVM
  - Reactive, event-driven programs
- Applications: chat, match maker, multiplayer game

Don’t program the hardware of the future with the software of the past

AMBIENTTALK

http://prog.vub.ac.be/amop