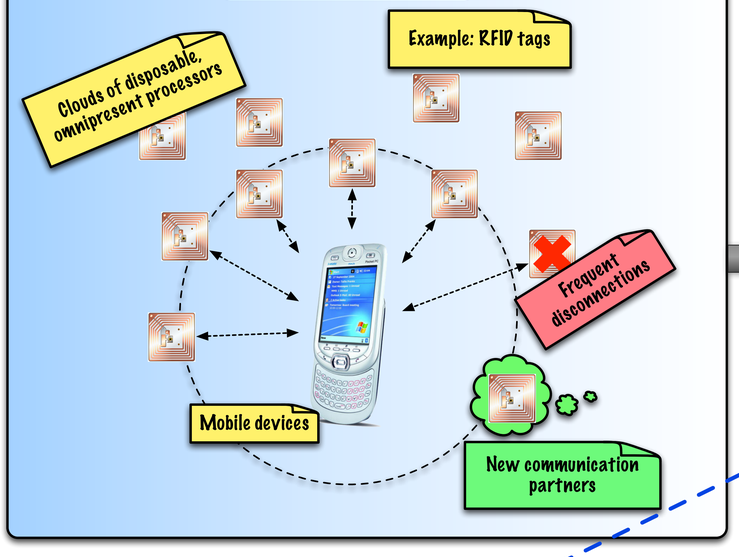
Group Communication Abstractions for Distributed Reactive Systems

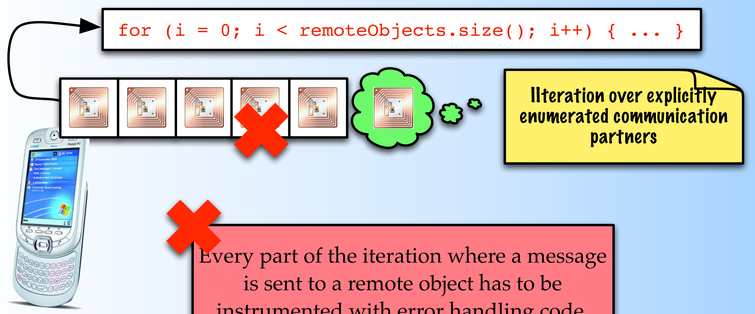
Andoni Lombide Carreton Stijn Mostinckx Wolfgang De Meuter

Programming Technology Laboratory
Vrije Universiteit Brussel, Belgium

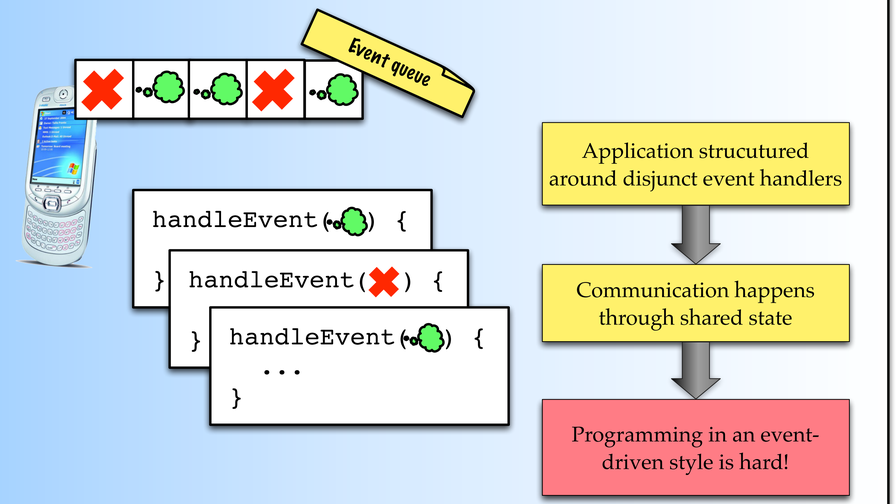
Context



Classic Distributed Systems



Event Driven Systems



AmbientTalk

- Object oriented, distributed and concurrent
- Asynchronous message sending
- Explicit references to remote objects
- Futures and future observers

Example: Smart shopping cart equipped with RFID reader

Programmer is responsible of writing event handlers to process the results and to deal with disconnections. Polling has to be done manually to keep the results up to date.

```
def prices := Vector.new();

rfidTagsInCart.each: { |tag|
  when: tag<-getPrice() becomes: { |price|
    prices.add(price);
} catch: TimeOutExc using: { |exc|
    system.println("Could not read tag!");
}}
```

Reactive Programming in Ambienttalk

```
>system.println(seconds)
>>7684034598
>>7684034599
>>7684034600
```

Time-varying values called behaviors

- Object oriented, distributed and concurrent
- Asynchronous message sending
- Reactive collections of remote objects
- Futures and future observers

In our example however...

```
def updatePrices(tags) {
  prices := Vector.new();
  tags.map: { |tag|
    when: tag<-getPrice() becomes: { |price|
      prices.add(price);
}}};

updatePrices(rfidTagsInCartBehavior);
```

prices is not a behavior!

There is still a gap between the reactive local application and the results computed in parallel by the outside world.

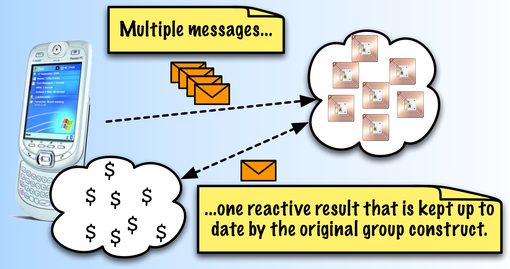
rfidTagsInCartBehavior reflects the state of the network and updatePrices will be reevaluated each time the behavior changes.

...but we still have to use event handlers to process the results of asynchronous invocations.

Reactive Group Communication

```
def pricesBehavior :=
  rfidTagsInCartBehavior.collect: <-getPrice();
```

Use a higher order message to deliver the argument message and aggregate the results in a behavior.



- Object oriented, distributed and concurrent
- Asynchronous message sending
- Broadcasting of messages to reactive groups of remote objects
- Sustained communication
- Reactive aggregation of results of asynchronous invocations

Further Reading

[1] Andoni Lombide Carreton, Stijn Mostinckx, Wolfgang De Meuter, "Group Communication Abstractions for Distributed Reactive Systems", Vrije Universiteit Brussel, 2008

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