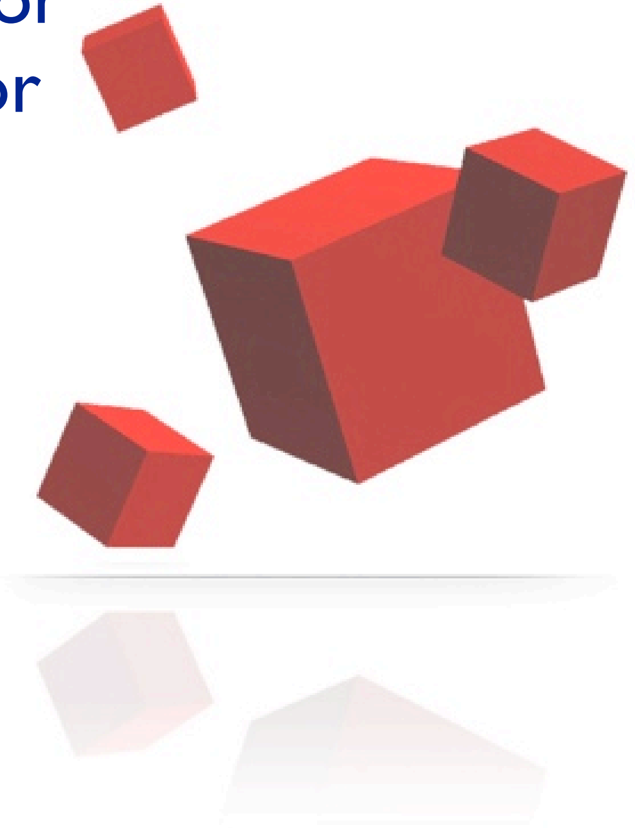
REALTALK
A Programming Language for Wireless Embedded Sensor Network

Alexandre Bergel
Siobhán Clarke

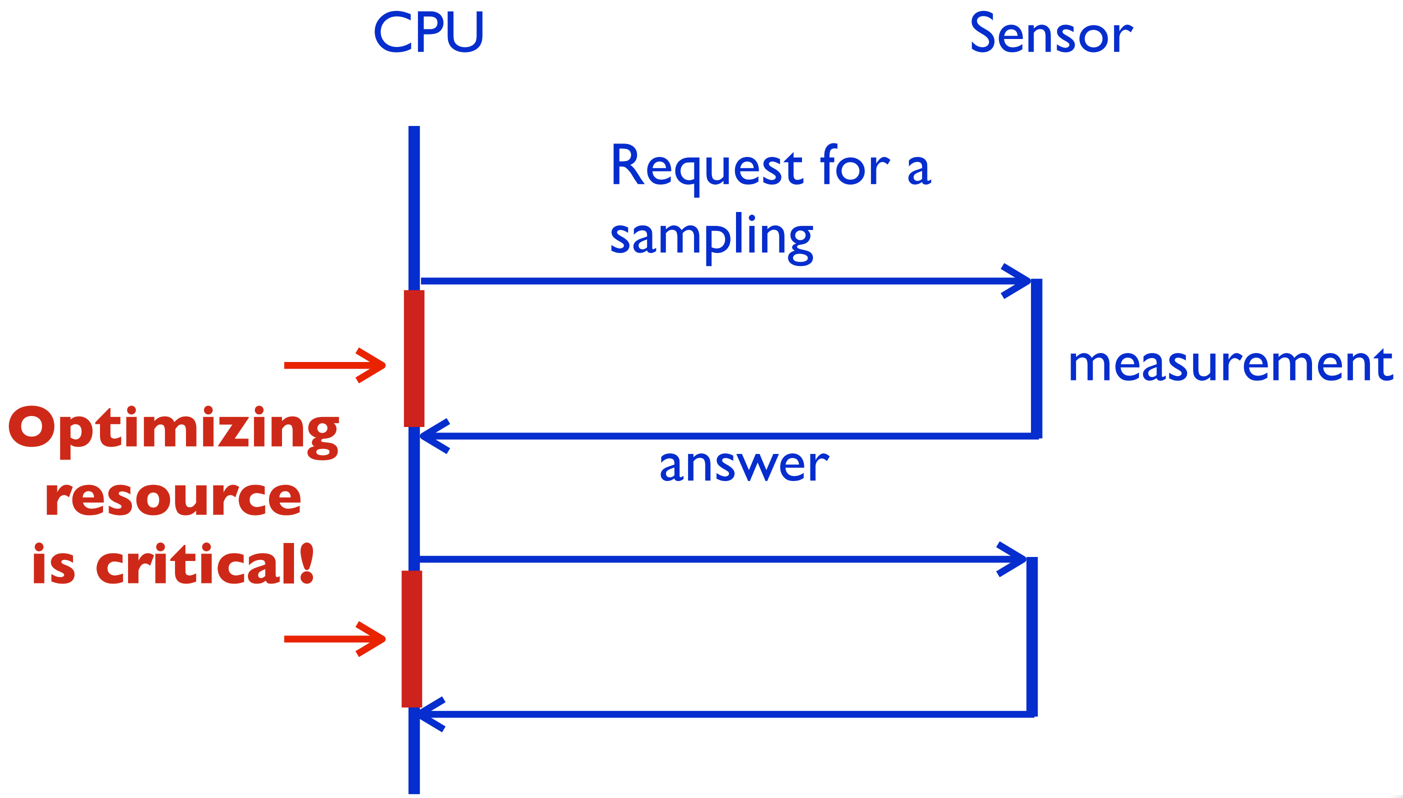
Distributed Systems Group
Trinity College Dublin



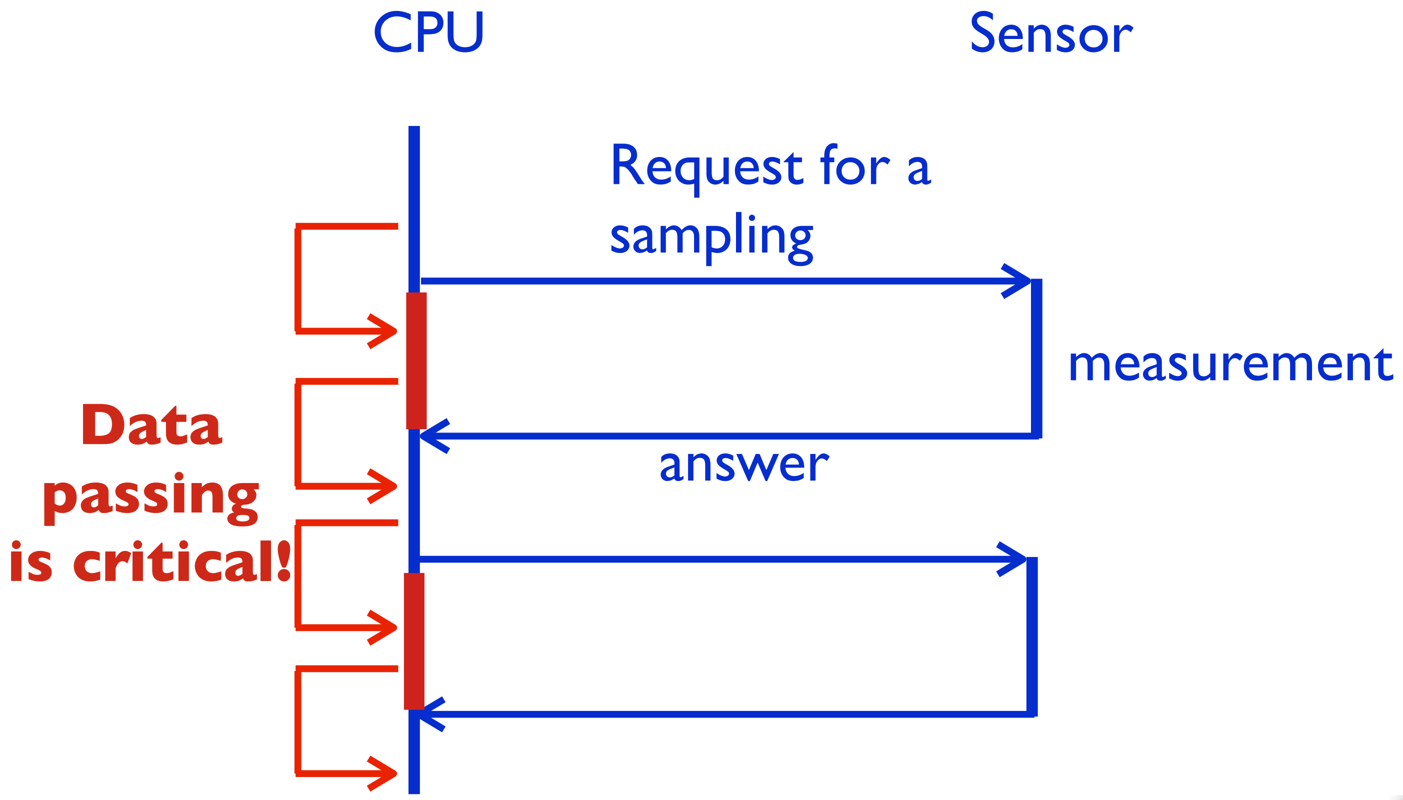
Programming wireless sensor devices is difficult

- Embedded sensor systems are difficult to program.
- Power consumption, memory management, hardware abstraction:
  - component maintenance: the C programming language prevents software component from being easily maintained.
  - sensor asynchronism: sampling is achieved by emitting and receiving events.

Sensor sampling creates disruption in the app



Sensor sampling creates disruption in the app



Realtalk: An Object-based language

Goal: making embedded programming easier and providing advanced hardware abstraction.

Creation of a counter application:

```
RTObject subclass: #Counter
  variableNames: ‘leds timer value’.

Counter compile: ‘start
  value := 0.
  timer invoke: #triggeredMethod every: 500.’

Counter compile: ‘triggeredMethod
  value := value + 1.
  leds display: value.’
```

Object composition and hybrid type system

Linking the counter to the leds and a timer:

```
RTObject subclass: #Counter
  variableNames: ‘leds timer value’.

Counter composeWith:
  { #leds -> Leds. #timer -> Timer }
```

Realtalk supports an *hybrid type system.*

Class specialization

Let’s emit a sound at each increment:

```
Counter subclass: #SoundCounter
  variableNames: ‘sounder’
  aliases: { #triggeredMethod -> #incAndDisplay}.

SoundCounter compile: ‘triggeredMethod
  self incAndDisplay.
  leds isRedOn
     ifTrue:  [sounder start]
     ifFalse: [sounder stop].’
```

Example of Controlled Disruption

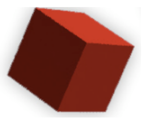
Reading two sensors and emitting their sampling:

```
AnyObject compile: ‘readingTwoSensors
    | value1 value2 |

    leds display: 1.
    value1 := lightSensor read.

    leds display: 2.
    value2 := soundSensor read.

    leds display: 3.
    radio emit: (value1 + value2).’
```

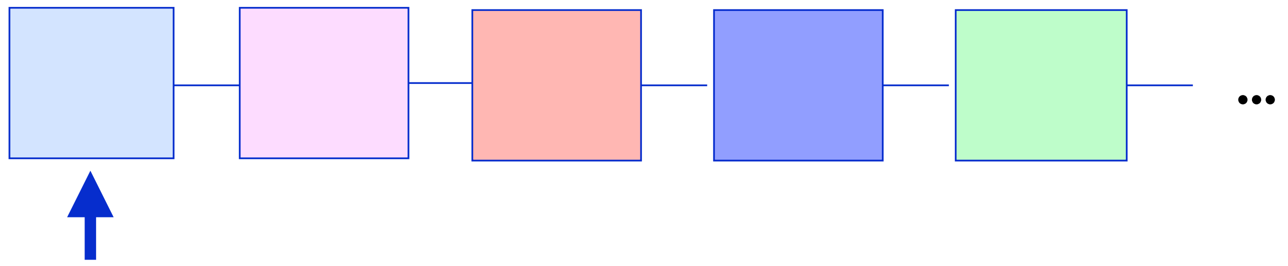
Method is cut down into pieces...

Reading two sensors and emitting their sampling:

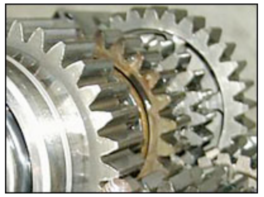
```
AnyObject compile: ‘readingTwoSensors
   | value1 value2 |
    leds display: 1.
    value1 := lightSensor read.

    leds display: 2.
    value2 := soundSensor read.

    leds display: 3.
    radio emit: (value1 + value2).’
```

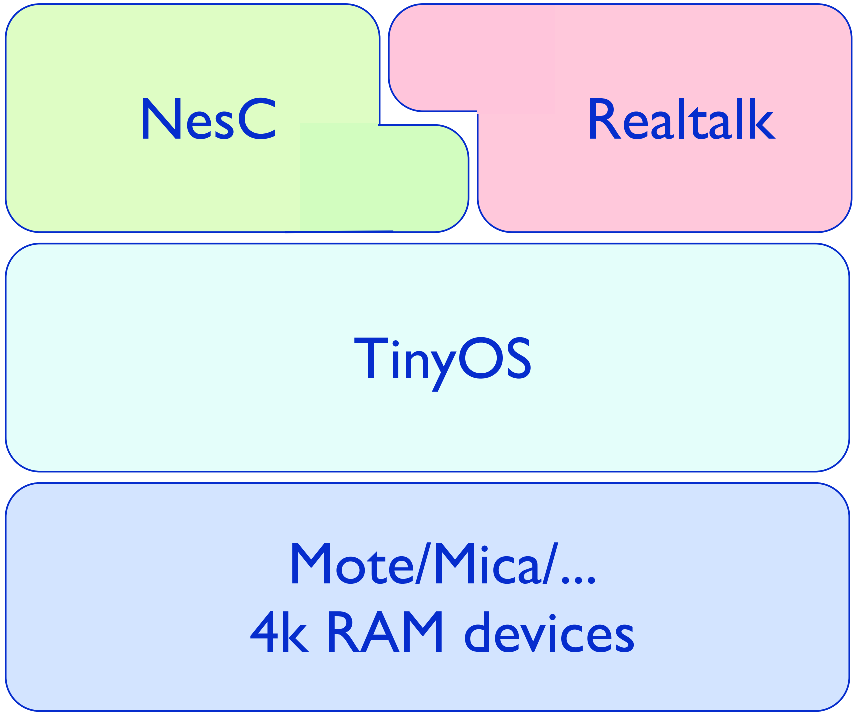


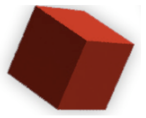
... and those pieces are inserted in a pool



Implementation based on TinyOS/NesC







Conclusion

- Realtalk is a scripting language dedicated to small sensor device with high limited resources.

- It provides advanced features regarding:
  - component interactions
  - component specialization
  - sensor programming

- On going work focuses on large software construction.