

Code Mobility

Konstantin Selyunin

e1228206@student.tuwien.ac.at

Igor Pelesić

igor.pelesic@gmail.com

Miljenko Jakovljević

micky686@gmail.com

December 5, 2012

Outline

- 1 Introduction
 - Code mobility overview
 - Level of abstraction
 - Requirements
- 2 System architecture
 - General overview
 - Agents
 - Platform
 - Scheduler
 - Execution Layer
 - Communication Protocol
- 3 Project management
- 4 Tools

Code mobility overview

Concept of code mobility

Our goal:

- Design code mobility system on ESE Board
- Hardware drivers & mobile agents & communication
- Master project management skills

Concept of code mobility

Mobile agent

Strong and *weak* code mobility

Layered architecture

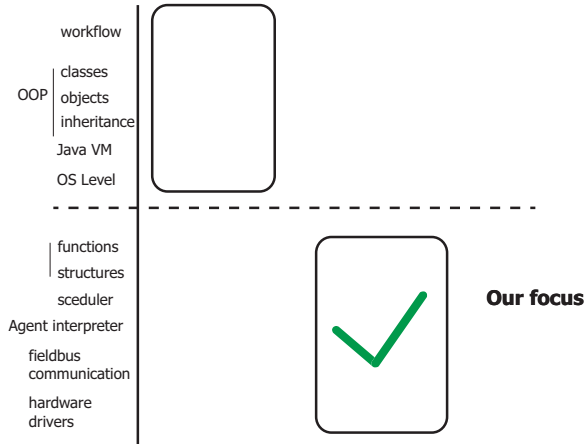
Advantages of code mobility

Move code close to resources

Enable client customization of remote resources

Performance gains

Level of abstraction



Requirements

- Agents:
 - simple language
 - support mobility and message exchange
- Platform:
 - execute agents concurrently
 - provide hardware services to agents
- Communication:
 - transfer agents & state *strong mobility*
 - transfer messages between platforms
 - cross board communication via Zigbee

General overview

3 layered architecture:

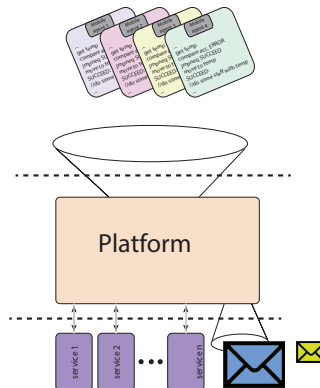
- Agent level
- Platform level
- communication & drivers



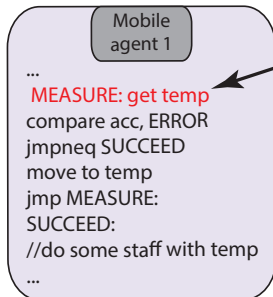
General overview

3 layered architecture:

- Agent level
- Platform level
- communication & drivers



Agents



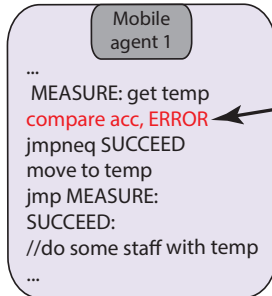
Get temperature value

Platform can provide this service?

yes: do stuff

no: move agent to another platform

Agents



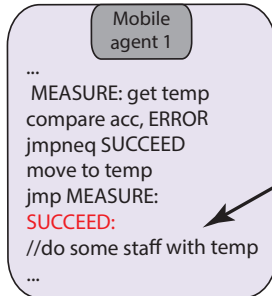
Get temperature value

Platform can provide this service?

yes: do staff

no: move agent to another platform

Agents



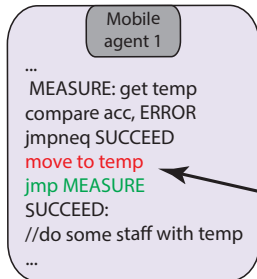
Get temperature value

Platform can provide this service?

yes: do staff

no: move agent to another platform

Agents



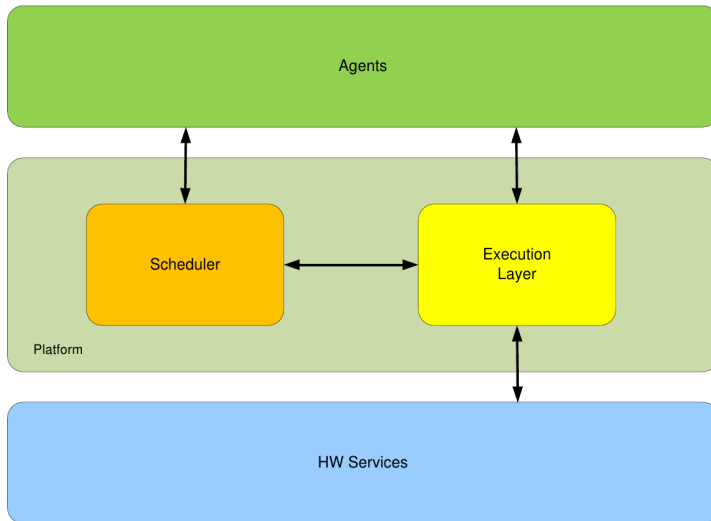
Get temperature value

Platform can provide this service?

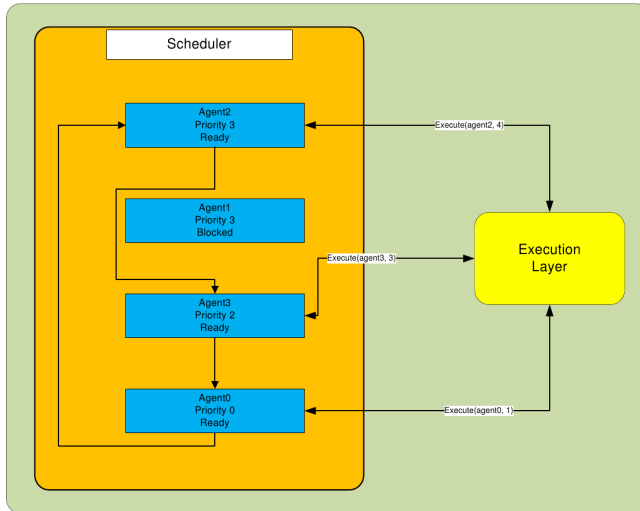
yes: do staff

no: move agent to another platform

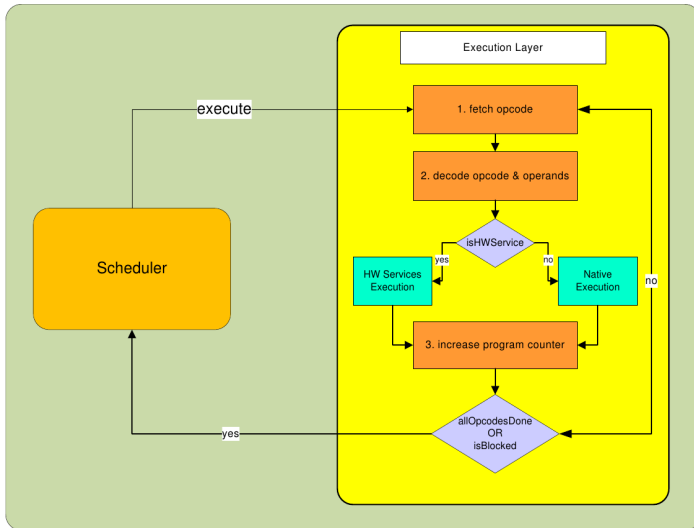
Platform



Scheduler

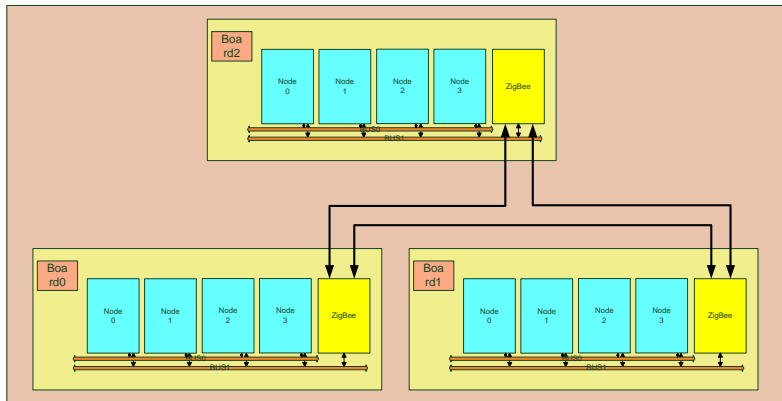


Execution Layer



Communication Protocol

Network Infrastructure



Protocol Design

Application Requirements

Local and remote communication

Sending agent code

Sending application data

Protocol Design cont.

Design Principles

Layered design

- Low level - CSMA/CA
- High Level - Routing

Composability with *Zigbee*

- *IEEE 802.15.4*

Fairness in network access

Acknowledgement and retry

- *Unreliable network*
- *Congestion avoidance*
- *Complexity - e.g. TCP*

Transmission Layers

Byte	MSB	LSB
0	<i>destination node</i>	<i>payload length</i>
1	<i>data</i>	
1	...	
14	<i>data</i>	
15	<i>crc</i>	

Figure: Low Level Datagram

Byte	MSB	LSB
0	<i>destination node</i>	<i>payload length</i>
1	<i>source node</i>	<i>destination board</i>
2	<i>source board</i>	<i>packet type</i>
3	<i>frame id</i>	
4	<i>packet id high</i>	
5	<i>packet id low</i>	
6	<i>empty</i>	
7	<i>data</i>	
1	...	
14	<i>data</i>	
15	<i>crc</i>	

Figure: High Level Datagram

Network Configuration

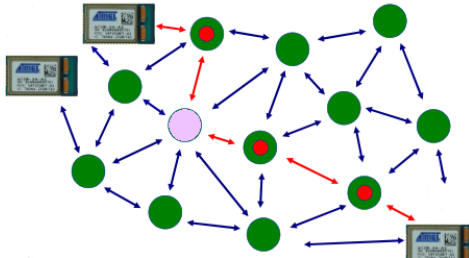


Figure: Zigbee Mesh Network

Zigbee Network Configuration

Rerouting Example

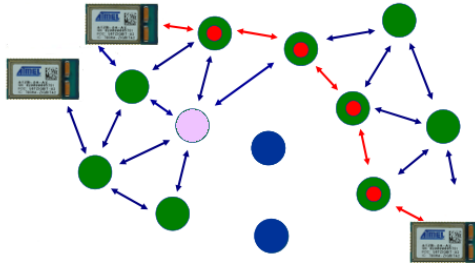


Figure: Network after rerouting

- Network Coordinator
- Failed Node
- Network Router
- Message Route

Milestones



Phase 1. Product outline and information gathering



Phase 2. Application requirements and specification



Phase 3. Implementation

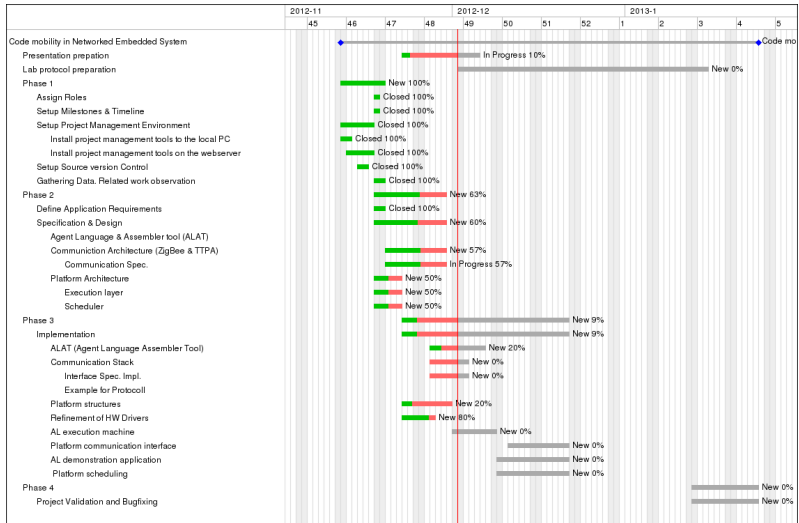


Phase 4. Validation and analysis

Workpackages

	Name	Dates	Interdependencies	Deliverables
WP1	Documentation	all	25.10.12 - 15.01.13	D1.1 Lab protocol
				D1.2 specification
				D1.3 workshop1
				D1.4 workshop2
WP2	Adaption of drivers		10.12 - 15.12	D2.1 hardware drivers
WP2	Agent language tool		6.12 - 10.12	D2.1 Agent language assembler tool
WP4	Communication	D2.1		Protocol
WP5	Platform	WP2, WP4	10.12 - 21.12	D3.1 Platform

Gantt diagram



Tools

Version control



git

Documentation & code repository



github

File sharing



amazon s3

Project management



redmine

<http://nes2012group4.herokuapp.com/>

Code generation



SCADE

Editors



Emacs

gedit