Telecooperation Office (TecO)

Overview

- Founded 11 years ago
- cooperation between CEC(SAP) and University of Karlsruhe
- Application-oriented research
- 5 RAs, 15-20 students (undergrads, master)
- 100% third-party funded
  - industry, e.g. SAP, Microsoft, Daimler, Nokia, Phillips, Infineon
  - Public grants: EC projects, national grants
- 1999: Focus on **Ubiquitous Computing**
  - Integration of ID- and Computer-Technologies in objects (smart items)
  - E.g. RFID, Sensors, sensor networks, location systems
About this Tutorial

Vision
- Smart Items become part of a next generation of the Internet-of-Things.

Objectives
- Understanding of Smart Item approach
- Technical skills and know-how for Smart Items

Approach
- Teach theory and do hands-on exercises
Outline

▪ Theory
  ■ Relevance and Motivation
  ■ Smart Items (SI)
  ■ Cost functions
  ■ Use Case: Collaborative Business Items
  ■ Results

▪ Hands-on
  ■ Java VM on Particle platform
  ■ Infrastructure
  ■ Communication
  ■ Information Processing
  ■ Excerises
Relevance

Control of complex information flows between processes in the real world and computer-based information systems.
TecO Research Focus

Motivation
- Strongest connection between the information and real world by re-location of computation: processors interwoven in processes

Concept
- Embodied virtuality, i.e. embedding of information systems into objects
- Information Appliances
  → functional decomposition of general purpose computers

Realization
- Smart Items

Source: ubiq.com
Definition: Smart Items (SI)

**Smart Items** = items + embedded wireless sensor systems
- Items = goods, tools, equipment, ...

**Smart Item can**
- *autonomously* recognize, process, decide, communicate and act

More precise:
- Continuously and direct acquisition of environment via sensing
- Recognition of own state (="Smartness")
- Local processing, decision making and reaction (=proactivity)
- Communication to other Smart Items (=collaboration)
- Affect the environment through actuators
Related concepts

- „Internet of Things“ / Transponder technology
  - Communicates ID, seldom sensor data
  - Transponder → central reader / Back-end (n:1 relation)
  - Regular information processing, “smart” back-end (“real-world awareness”)

- **Smart Item = Item + Computer in the real world**
  - Sensors,
  - Bi-directional, wireless communication
  - Memory, logic (programs, functions)
  - No permanent back-end / data sink connectivity required
  
  → new capabilities: autonomous, proactive, context aware, collaborative
Characteristics of SI Systems

- miniaturized, wireless sensor network systems
- Rich capabilities & features
  - HW: small, embedded, many components, modular, constrained
  - SW: manifold relations for information processing, freely programmable computers
- Location of information processing
  - On item
  - Between items and other computer systems
Relocation of Computation: Smart Item Examples

**DigiClip (IWSAWC 2004)**
- Couple paper-based documents to computer-based document management systems
- App: Active physical (paper-based) documents

**eSeal (Pervasive 2004)**
- Transform electronic contracts onto physical items
- App: Trustworthy, continuous self-supervision of goods during transport

**Collaborative Business Items (2006)**
- Collaborative detection of hazardous situations of chemical goods
- App: safety provision for worker, environment

http://www.cobis-online.de
Central vs. Collaborative Event detection

- Central processing

  ![Diagram](image1.png)

- Collaborative processing

  ![Diagram](image2.png)

Event: yes
Central vs. Collaborative Cost Model

Linear cost model
- Includes parameters such as
  - Communication period → communication costs
  - Execution time → execution costs
  - Event rate → detection frequency
  - Sampling period → sampling costs
  - Sensor node energy budget → energy units
  - Packet processing time → comm. stack delay
- Real-time requirement: event rate < communication rate (!)

- Parameters derived empirically for wireless sensor network and standard desktop PC

Investigated criteria
- Energy consumption
- Network load
- Message loss recovery
Central vs. Collaborative Energy Consumption

- Small advantage for collaborative processing due to communication at event detection
Central vs. Collaborative Network Load

- Assumption: constant communication period, no losses, no bursts
- What happens at irregular traffic? → next slide
Central vs. Collaborative Network Load (irregular traffic)

- Irregular traffic $\rightarrow$ short comm. periods for central to handle event bursts
- Load metric moves on a cost surface for irregular traffic
- Bursts account for non-linear behavior
Central vs. Collaborative Message Loss Recovery

- Assumed loss rate: 30%
- Fast recovery for collaborative due to low network delay and lower communication rate
Central vs. Collaborative Results and Discussion

Results
- Energy consumption: comparable
- Network load: advantage collaborative approach
- Loss recovery: advantage collaborative approach

Critiques and discussion
- Simplicity
  - Cost model is only linear
  - Still too few parameters
- Coverage
  - There is more than energy, network load, loss recovery
  - Business aspects missing 😞
Outline

**Theory**
- Motivation and Relevance
- Smart Items (SI)
- Cost functions
- Use Case: Collaborative Business Items
- Results

**Hands-on**
- Java VM on Particle platform
- Infrastructure
- Communication
- Information Processing
- Excerises
Relocation of Business Logic: CoBIs

- Re-locate business logic from information systems to business items
- Collaborative execution directly between the items → Smart Items

TecO & CoBIs: http://www.cobis-online.de
Relocation of Business Logic: CoBIs

- Supported Business Processes
- Real-time Data (U1)
- Relocated Process Tasks (U2)
- Process Control (U3)

Service Proxy Layer:
- Service₁
- Service₂
- Service₃
- Service₄

Physically Embedded System
CoBIs Example

Collaborative detection of

- Storage limit (< 100 drums)
- Hazardous combinations
- Check for correct storage place (storage A, not storage B)

→ Alert locally, no permanent back-end required

Smart Item technology

- Wireless sensor node on items
- Items monitor *continuously a common state*
- Warnings in real-time (!)

Source: BP p.l.c.

TecO & CoBIs: http://www.cobis-online.de
CoBIs – Storage Limit (Example)

Collaborative Interaction

- Storage limit set to 3
- Nodes communicate with each other, count neighbours
- If count exceeds 3, then raise alert

Features

- Distributed, no central authority
- Alert is checked by communication partners
CoBIs – Implementation

- Site: BP chemical plant
- Deployment of 20+ nodes
- Basic middleware integration support (WLAN, UPnP)
- Integration with SAP EH&S solution

TecO & CoBIs: http://www.cobis-online.de
CoBIs Usage in Hull, UK
Network Layout

- 3 Locations, different types of chemical drums
- Wireless connection to on-site network (WIFI COTS router)
- UPnP Gateway for service-oriented binding to SAP application server (SOA)
- External ftp server for logging
Investigation

Recap
- Collaboration keeps processing within the network
- Investigate message load at gateway / application
# Results

- Communication load
- All messages, incl. logging

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Expected value</th>
<th>Actual value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average message load on the application server level</td>
<td>20 msg/min</td>
<td>7 msg/min</td>
</tr>
<tr>
<td>Peak message load on the application server level (all drums put to same location)</td>
<td>400-500 msg/min</td>
<td>212 msg/min</td>
</tr>
<tr>
<td>Average message load from the sensor network</td>
<td>N/A</td>
<td>187 msg/min</td>
</tr>
<tr>
<td>Comparison: messages on gateway level vs. messages on application server level</td>
<td>80% of the messages are filtered out</td>
<td>96% filtered out</td>
</tr>
<tr>
<td>Propagation of new rules/ configurations</td>
<td>20 s per drum</td>
<td>23.3 s per drum (210 s for nine drums)</td>
</tr>
</tbody>
</table>

**Configuration**

<table>
<thead>
<tr>
<th>Service name</th>
<th>Sampling rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hazardous Goods Service</td>
<td>1 Hz</td>
</tr>
<tr>
<td>Sensor Service</td>
<td>Temperature: 0.05 Hz - Voltage: 0.4 Hz</td>
</tr>
<tr>
<td>Location Service</td>
<td>IR Location Sensor: 40 Hz</td>
</tr>
</tbody>
</table>

~3% application relevant, e.g. alert, monitoring msg.
Results (cont.)

- **Events**: Hazardous goods detection, e.g. storage limit, incompatibles goods, environmental alerts
- Detection time: < 1second

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Actual value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of nodes</td>
<td>&gt;=20</td>
</tr>
<tr>
<td>Number of services/proxies</td>
<td>&gt;=80</td>
</tr>
<tr>
<td>Avg. Gateway event load prior filtering/service transformation</td>
<td>10 msg/s</td>
</tr>
<tr>
<td>Avg. Gateway event load after filtering/service transformation</td>
<td>2 msg/s</td>
</tr>
<tr>
<td>Processing time for transformation</td>
<td>&lt;100ms</td>
</tr>
<tr>
<td>Processing time for (non-blocking) invocation</td>
<td>&lt;10ms</td>
</tr>
</tbody>
</table>

State updates, Further reduction by relaxing detection time
Experiences

Sensor nodes deployment
- Autonomous operation for 1 month
- Fast, accurate, robust detection in critical situations

Infrastructure
- 802.11b Wifi COTS routers
- High message loss due to humid weather, proximity to cooling towers
- Effects on UDP traffic
  - UPnP discovery
  - DHCP service
- Long discovery delays and a delayed notification of events for back-end system
Conclusion

**Smart Items**

- Promising concept for sensor networks in advanced industrial applications
- Integrated with modern business information systems

**Benefits of Collaboration**

- Scaling
  - Processing is delegated away from the back-end
- Real-time action
  - Re-located processing shortens notification paths
  - Enables robust immediate action
- Increased degree of freedom
  - Mobility with no permanent back-end connection
Outline

- **Theory**
  - Motivation and Relevance
  - Smart Items (SI)
  - Cost functions
  - Use Case
  - Results

- **Hands-on**
  - Particle platform
  - JavaVM
  - Infrastructure, Communication
  - Information Processing
  - Excerises
Tutorial Smart Item Hardware
Particle Computer

- Communication:
  - AwareCon: 868 MHz, TDMA protocol

- Sensing:
  - Infrared (IR) receiver for location system
  - Acceleration, shock, temperature, light

- Power Supply: 2x AAA batteries
Java on Particle Computers

- Subset of Java VM on Particle computer

Java Source

---

Standard javac compiler

Java ByteCode

---

Strip down, optimization, versioning

Java ByteCode for Particles

---

Wireless Transfer

Versioning control, Selective updates, Mass programming

Java Virtual Machine

Particle Computer

TecO

Smart Items

32
Particle VM

**Features**
- All java language features except for reflection and exception handling
- Automatic garbage collection
- Object de-/serialization
- Usage of 32bit arithmetic operation on low-end (16bit and 8bit) microcontrollers
- Guarantees type safeness
- Java Native Interface (JNI) for time critical or performance critical execution, e.g. sensor sampling, communication
- Executes byte code generated by standard Sun's javac compiler

**Novel features**
- Automatic version control for java classes
- Version’ed upload mechanism for over-the-air programming
- Mass programming in field
ParticleVM
Runtime Library

**Functionality**
- Provides basic platform support
- Java classes for
  - Object de-/serialization
  - Binding to hardware
  - Communication and message passing
  - Sensor abstraction interfaces

<table>
<thead>
<tr>
<th>Table: Evaluation of runtime library</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size of all .class files</td>
</tr>
<tr>
<td>Size of transformed .pclass files</td>
</tr>
<tr>
<td># classes</td>
</tr>
<tr>
<td>Factor of code size reduction by transformation</td>
</tr>
<tr>
<td>Memory consumption (int. Flash)</td>
</tr>
</tbody>
</table>
### Particle VM Memory consumption

- **Low memory consumption**
- executable on low-end microcontrollers (cost efficient)

<table>
<thead>
<tr>
<th>RAM (3.5 KB)</th>
<th>ParticleVM (1.5 KB)</th>
<th>VM-Heap (1.5 KB)</th>
<th>ParticleOS (0.5KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program Flash (128KB)</td>
<td>Particle VM (60KB)</td>
<td>ParticleOS (45KB)</td>
<td>Free (23KB)</td>
</tr>
<tr>
<td>External Flash (max. 512KB)</td>
<td>Storage for Application Java classes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 512KB for user java application
- May allow even complex java programs
ParticleVM
Over-the-Air programming

Version control
- Automatic versioning within class files
- Version control on PC and Particle sensor node (!)
- Transfer only new classes or updated classes
  → small code updates
  → very fast

Over-the-Air programming
- RF is inherently broadcast → mass programming
- *This tutorial:* programming via addressed device

Measured results on Particle sensor nodes
- Class update: 2-4 seconds
- Runtime library (20 classes): 50-60 seconds
ParticleVM
Execution Speed

- Avg. interpretation overhead: 3000 % (factor 30)

- Flash overhead: slowdown caused by access to flash for new byte code instructions
- >50% slowdown due to flash overhead (and not Java)
Outline

- **Theory**
  - Motivation and Relevance
  - Smart Items (SI)
  - Cost functions
  - Use Case
  - Results

- **Hands-on**
  - Particle platform
  - JavaVM
    - Infrastructure, communication
  - Internals: Java run-time
  - Excerises
Infrastructure: Communication

SVN Server
Code deployment

Your workstations

XBRidge

ByteCode

Smart Items
Infrastructure: Location System

Infrared (IR) Cell-of-Origin location system

- IR beacon device illuminates a footprint
- Modulates an identification number (ID)
- Sensor device receives ID
- Location resolved using look-up table ID→location
- Accuracy: illumination area
Communication: AwareCon Stack

- TDMA communication protocol for Particle
- Time frame collaboratively established
- Fully distributed, no master needed
- 1 slot = 13ms
  - ~ 8ms communication phase
  - ~ 5ms application phase → **Java processing**

![Diagram of AwareCon Timeslot](image-url)
Java Processing

- AwareCon preempts every 13ms
- MainLoop drives Java application
- Java App: runs to completion, no yield
Java MainLoop

Mainloop
- Drives application
  - Slotted: as fast as possible (>= 13ms (slot time) )
  - Per second: each 77 slots
  - Delay by application (!)
- Processes packets
  - System packets, e.g. loading new classes
  - App. specific packets

Your Java App

JavaVM MainLoop

Sensor Sampling Packet Assembl.

Slotted, PerSecond

Packet Proc.

no yield

Time
Java Application

Application
- Several classes
- public static void main(..)
  - Hooks in objects for being run by MainLoop

Example:
public static void main(...)
{
    Scheduler.ExecuteSlotted(new Blink());
}

class Blink implements ExecuteSlotted
{
    ...
    public void onSlot() {...}
}
Consequences for Application Runtime Design

- Application does not yield
- MainLoop might get delayed (sometimes forever)
- AwareCon stack will always run

- while(true) loops:
  - Delays MainLoop forever
  - Will prevent code update
    (code update is part of JavaVM)

→ Restart Smart Item
  - At bootup, auto-check for new code
Communication Protocol
ConCom

Characteristics in wireless sensor networks
- Small packets ~30-70 bytes, small headers
- 1 CRC for all layers
- No standard addressing
  - Device address
  - Application specific addressing, i.e. written (text) identifier within the message

ConCom
- Communication data format for Particle Computers
- Device + application specific addressing
- Flat hierarchy → sentence

→ Supported by Java runtime library
Communication Protocol
ConCom

- Data encoding above the network layer
- Data encoded as tuples, strictly typed
  - Type: 3 bytes, freely selectable
  - Length: 1 byte, data length
  - Data: data to transmit
- Subject = first tuple, identifies application
- Sentence = subject + [tuples]*
- Max. length: 64 bytes (!) per packet
Communication Protocol
ConCom

**Addressed Communication**
- Addressing is managed by AwareCon
- Add CAD tuple to ConCom sentence
- Address: 8 Byte ID

Example:
- ABC 1 [tuples]* CAD <your SI ID>

Your Smart Item ID: 1.1.1.1.1.2.3.4
Outline

**Theory**
- Motivation and Relevance
- Smart Items (SI)
- Cost functions
- Use Case
- Results

**Hands-on**
- Particle platform
- JavaVM
- Infrastructure, communication
- Internals: Java run-time
- Excerises
Exercises

- Preparation hardware / software
- Development process
- Exercises
  - Ping
  - Hello World
  - Sensor Reading
  - CoBIs

Together, step-by-step
Your exercise
Tutorial Smart Item Hardware
Particle Computer

- Communication:
  - AwareCon: 868 MHz, TDMA protocol
- Sensing:
  - Infrared (IR) receiver for location system
  - Acceleration, shock, temperature, light
- Power Supply: 2x AAA batteries

Java VM
PIC 18F6720

Particle Computer (Communication, Processing)
IR Location Technology
Sensors and actuators
Power Supply (AAA, lifetime ~ 1 month)
Software Requirements

Local development
- Java JDK, min. 1.4
- Eclipse Development Framework
- Subversion (SVN) → subclipse

Infrastructure
- Compiling
- Versioning
- Deployment

- Software repository
  - JDK, Eclipse, subclipse
Development Process

- All code is versioned!
- Your (versioned) code is coupled with your Smart Item via the svn directory.

Development process
1. Start with the provided code template
2. Write your code
3. Compile your code
4. Check-in your source code (login and password provided)

Infrastructure takes over
5. Auto-update to your Smart Item
Exercises

- Preparation hardware / software
- Development process

- Exercises
  - Ping
  - Hello World
  - Sensor Reading
  - CoBIs

\{ Together, step-by-step \} \quad \{ Your exercise \}
Exercise: Ping

Task
- Detect your Smart Item via „Ping“

Protocol
- Ping is supported by AwareCon through HELO/OLEH
- ConCom:
  - HELO: ACM CHE 255 255 255 255 255 255 255 255
  - OLEH: ACM CEH <workstation id>
- Workstation ID: 1.1.1.1.<ip address>

Your workstation

Your workstation

ACM CHE 255 255 255 255 255 255 255 255

ACM CEH <workstation id>
Exercise
Hello World

Task
- Send a „Hello World“ from your Smart Item

Protocol
- Write a Java program for your Smart Item
- Send a request (ConCom) to your Smart Item
- Smart Item replies with „Hello World“
- Receive „Hello World“ and print on console
Exercise
Sensor Reading

Task
- Send Sensor Information back to PC

Protocol
- Write a Java program for your Smart Item
- Read sensors, e.g. light sensor, temperature sensor, location ID
- Think of sampling speed: slotted / PerSecond

Your workstation

ABC SN1 value SN2 value ...
Exercise
Collaborative Business Items

Task
- Manage hazardous situation in a chemical plant

Group work
- Group 1: supervise storage location
- Group 2: detect incompatible goods
- Group 3: ensure storage limit (tough!!!)

Protocol for all Smart Items
- Define your chemID, e.g. water, oxid, inflammable
- Define your current state, e.g. location, chemID, storage number
- Send out periodically your chemID and state

http://www.cobis-online.de
Thank you

Telecooperation Office (TecO)
University of Karlsruhe

cdecker@teco.edu