ROVERS: Pervasive Computing Platform for Heterogeneous Sensor-Actuator Networks

J. Domaszewicz, M. Rój, A. Pruszkowski, M. Golański, K. Kacperski

Institute of Telecommunications
Warsaw University of Technology
Presentation outline

- Concept of open pervasive computing platform
- ROVERS programming model
- Summary
Concept of Open Pervasive Computing Platform
Heterogeneous sensor-actuator network

CPU, memory,
node-specific sensor and actuators resources
Open pervasive computing platform

- Nodes form a quasi-static, ad-hoc, peer-to-peer network
- The network is heterogeneous in terms of nodes’ sensor and actuator resources
- Each node contributes its unique resources to the platform
- Available resources not known at the development time
- Support for „opportunistic pervasive computing” (discover and take the best advantage of whatever resources are available)
- Ability to inject new applications (active sensor network)

- **Implementability on constrained nodes (of the TinyOS class)!**
Our goal

- To create a middleware transforming a heterogeneous sensor-actuator network into an open pervasive computing platform

![Diagram showing ROVERS Applications, ROVERS Middleware, and Environment](image)
ROVERS Programming Model
**Micro-agent**

- A ROVERS application consists of **micro-agents**
- Micro-agents do (a) context acquisition, (b) context synthesis, (c) decision making, and (d) effecting change through actuators
- They are organized into a tree-like hierarchy (one **boss** and a number of **subordinates**)
- A micro-agent has a type (a nesC-like interface towards its boss)
- Internally, a micro-agent is a collection of event handlers
The logical structure of a ROVERS application

- micro-agent
- communications channel
Building the application tree (1)

- Building the application tree starts from the root and progresses towards leaves

- A boss hires its subordinates, which, in turn, can hire theirs

- Hiring is a ROVERS primitive that may yield a new instance (instances) of the subordinate

- Once the instance is created, ROVERS establishes a communications channel between the boss and the subordinate
Building the application tree (2)

Injecting the application
Building the application tree (3)
Building the application tree (4)
Building the application tree (5)
Building the application tree (6)
**ROVERS Virtual Machine – RVM**

- Micro-agent execution environment
- Generates events, interprets instructions
- Available at every node
- Supports *node-specific* event set and instruction set

![Diagram of ROVERS Virtual Machine (RVM)]
RVM – events and instructions (1)

- Remainder: the network is heterogeneous in terms of nodes’ resources (sensors and actuators)
- Remainder: a micro-agent is a collection of event handlers

Events, instructions

- Generic
  - At every node
- Non-generic
  - At some nodes
  - Represent node-specific resources
RVM – events and instructions (2)

- ROVERS exposes node-specific resources by means of non-generic RVM events and instructions

- In our approach, non-generic events and instructions are **domain-oriented** and **ontology-driven**
RVM – events and instructions (3)

<table>
<thead>
<tr>
<th>Instructions (samples)</th>
<th>Events (samples)</th>
</tr>
</thead>
<tbody>
<tr>
<td>add</td>
<td>OnceEvent</td>
</tr>
<tr>
<td>push</td>
<td>TimerEvent</td>
</tr>
<tr>
<td>call</td>
<td>CommandEvent</td>
</tr>
<tr>
<td>hireGeneric</td>
<td>ReportEvent</td>
</tr>
<tr>
<td>hireNonGeneric</td>
<td></td>
</tr>
<tr>
<td>command</td>
<td></td>
</tr>
<tr>
<td>report</td>
<td></td>
</tr>
<tr>
<td>switchOnDisplay</td>
<td>RefrigeratorOpenEvent</td>
</tr>
<tr>
<td>switchOnPDADisplay</td>
<td>LightSwitchedOffEvent</td>
</tr>
<tr>
<td>switchOffPDADisplay</td>
<td>CeilingLampInKitchenEvent</td>
</tr>
<tr>
<td>isOnDeskLamp</td>
<td>AlarmActivatedEvent</td>
</tr>
<tr>
<td>switchOffHomeItemInKitchen</td>
<td>WasherCycleOverEvent</td>
</tr>
<tr>
<td>turnDownCeilingLampInHall</td>
<td>DoorBellRingingEvent</td>
</tr>
</tbody>
</table>

4th International Workshop on Mobile and Distributed Computing (MDC’06)
Generic and non-generic micro-agents

- A micro-agent is **generic** if all its events and instructions are generic (i.e., if it does not use any sensors or actuators)

- Otherwise, a micro-agent is **non-generic**

- A generic micro-agent can run on any node

- A non-generic micro-agent can run only if a node’s RVM supports all its non-generic events and instructions (i.e., if the node has all the needed sensors and actuators)
Hiring and mobility of generic micro-agents

- Once a generic micro-agent is hired, an instance is created on the node where the boss resides
- After that, the instance can be moved by ROVERS at any time to any node
- Strong mobility
- Generic micro-agents are moved to
  - reduce application-generated traffic
  - offload overloaded nodes
- The programmer does not need to think of nodes when developing generic micro-agents
Hiring and mobility of non-generic micro-agents (1)

- A node is a **host** for a non-generic micro-agent if the node’s RVM supports all the non-generic events and instructions used by the micro-agent.
Hiring and mobility of non-generic micro-agents (2)

- Two modes:
  - **anycast** (instantiate on a single host)
  - **broadcast** (instantiate on all available hosts)
- Looking for hosts is done by ROVERS
- An instance is created on a host (or multiple hosts)
- Weak mobility (a running instance does not move)
- Non-generic micro-agent mobility amounts to service discovery
Micro-agent mobility (1)
Micro-agent mobility (2)
Micro-agent mobility (3)
Micro-agent mobility (4)
Micro-agent mobility (5)
Micro-agent mobility (6)
Micro-agent mobility (7)
Micro-agent mobility (8)
Micro-agent mobility (9)
Micro-agent mobility (10)

4th International Workshop on Mobile and Distributed Computing (MDC’06)
Micro-agent mobility (11)
Micro-agent mobility (12)
Micro-agent mobility (13)
Micro-agent mobility (14)
Summary
ROVERS at a glance

- Resource discovery through non-generic micro-agent mobility
- Traffic reduction through generic micro-agent mobility
- Code mobility transparent to the programmer
- Inter-agent communications (communications channel with application-derived addressing)
- Orderly representation of sensor and actuator resources (domain-oriented, ontology-driven)
- Support for component-based software development (micro-agents)
Ongoing and future work

- Algorithms for ontology-driven programming artifacts
- Attribute-based addressing (non-generic micro-agent mobility)
- Anycast in sensor networks
- A protocol for generic micro-agent mobility
- Effective mobile code in sensor networks (energy!)
- Ongoing implementation: Motes/TinyOS/nesC/802.15.4 (but the programming model not tied to any implementation platform)
Thank you!