EcoCast: Interactive, Object-Oriented Macroprogramming for Networks of Ultra-Compact Wireless Sensor Nodes

Speaker: Dan Avramescu
Presentation outline

1. Introduction
2. System Overview
   - Scripting
   - Compiling and linking
   - Reprogramming and execution
3. Evaluation
4. Conclusion
Introduction

- Macroprogramming for WSN

- Issues
  - Remote Firmware Update
  - Interactive Scripting

- Solution: EcoCast: lightweight, interactive, object-oriented macroprogramming framework

Challenge the future
System overview

- EcoCast
System overview

• Execution Flow:
  • Node handle creation
  • Command processing
    • Language wrapping – type marshalling and demarshalling
    • Code swapping and dynamic compilation
  • Message Issuing
Scripting

- EcoCast uses Python
  - Clean syntax
  - Can run in interactive mode and batch mode

- Interactive access through the Esh shell
# Scripting EcoCast Command List

<table>
<thead>
<tr>
<th>Command</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>listAll()</td>
<td>list IDs of found nodes</td>
</tr>
<tr>
<td>getNodes()</td>
<td>obtain a list of node instances</td>
</tr>
<tr>
<td>ecNode</td>
<td>open connection to one node</td>
</tr>
<tr>
<td>ecGroup()</td>
<td>open group connection(s) to node(s)</td>
</tr>
<tr>
<td>ecMap()</td>
<td>map function for node(s)</td>
</tr>
<tr>
<td>ecFilter()</td>
<td>filter function for node(s)</td>
</tr>
<tr>
<td>ecReduce()</td>
<td>reduce function for node(s)</td>
</tr>
<tr>
<td>&amp; jobs</td>
<td>run a command as a background job</td>
</tr>
<tr>
<td>scope...end</td>
<td>set symbol scope to node or nodes</td>
</tr>
<tr>
<td>extern</td>
<td>enable access global variable in symbol scope of node or nodes</td>
</tr>
</tbody>
</table>

## Python
- map(f, A)
- filter(f, L)
- reduce(f, L, [Init])

## EcoCast
- ecMap(f, GH)
- ecFilter(func, GH, f)
- ecReduce(func, GH, f)
Scripting

- Code Example – temperature reading

```python
>>> listAll()
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
>>> GH = ecGroup([1,2,4,5,7,9]) # group based on IDs
>>> GH
ecGroup([1, 2, 4, 5, 7, 9])
>>> ecMap(readTemp, GH)
>>> ecFilter(lambda x: x < 2000, GH, readADC)
[ecNode(1), ecNode(2), ecNode(4), ecNode(9)]
>>> ecReduce(lambda x, y: x + y, GH, readTemp)
177.5
```
Compilation And Linking

• Compilation from Python
  • Python code is translated into C
  • Dynamical type parameters issue solved with naming conventions
• Incremental linking and Code Swapping
  • Assigned addresses are kept
  • Independent pieces of code are overlaid
• Version Control and Patching Scripts
  • Nodes with the same version maintain the same entry (group strategy)
  • Patches are generated with diffed binaries
Reprogramming and execution

- Nodes have two software components: Store and Execute
- Group reprogramming of nodes
Reprogramming and execution

- Group execution of functions on nodes
Multi-hop Networking

- Phase 1: Topology discovery
- Phase 2: Normal transfer

### Discovery Phase

<table>
<thead>
<tr>
<th>Code</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Host Broadcast Request</td>
</tr>
<tr>
<td>3</td>
<td>Node Broadcast Request</td>
</tr>
<tr>
<td>4</td>
<td>Node Broadcast Ack</td>
</tr>
<tr>
<td>5</td>
<td>Host Broadcast Ack</td>
</tr>
</tbody>
</table>

### Transfer Phase

<table>
<thead>
<tr>
<th>Code</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Host Data Request</td>
</tr>
<tr>
<td>1</td>
<td>Node Data Ack</td>
</tr>
</tbody>
</table>
Evaluation

- One host PC, Ethernet base station and 10 “clean” nodes
- Reprogramming Latency

(a) Reprogramming time of ADC.

(b) Ratio of different steps of latency in reprogramming 10 nodes
Evaluation

• Round-trip command execution latency

Response time of serial execution and parallel execution of EcoCast

(a) TEMP

(b) ADC
Evaluation

- Comparison of memory footprint

<table>
<thead>
<tr>
<th>Runtime System</th>
<th>Program memory</th>
<th>Data memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>SensorWare</td>
<td>≤ 180KB</td>
<td>≤ 64KB</td>
</tr>
<tr>
<td>Maté</td>
<td>16KB</td>
<td>849B</td>
</tr>
<tr>
<td>LiteOS</td>
<td>30KB</td>
<td>≈ 1.6KB</td>
</tr>
<tr>
<td>Mantis</td>
<td>14KB</td>
<td>≤ 5KB</td>
</tr>
<tr>
<td>Marionette</td>
<td>≤ 4KB</td>
<td>153B</td>
</tr>
<tr>
<td>EcoCast</td>
<td>3.94 KB</td>
<td>215B</td>
</tr>
</tbody>
</table>
Evaluation

- Multi-hop latency

Latency of finding a path for each node
Conclusions

+ EcoCast adopts Phyton to enable interactive programming for WSN
+ Tool for beginner and expert programmers
+ Nodes are integrated as objects with the rest of application code
+ Suitable for resource-constrained platforms

- No prioritization of the most frequently used segments of code
- Adequate up to modest-sized multi-hop networks
Thank you for your attention
• Scripting Environment

- High level class library
- Esh shell
- Low-level class library
def PEG():
    # Request all nodes to do one minute sensing
    # n = nodes that have average readings > THRESHOLD
    n = ecFilter(lambda x: x > THRESHOLD, GH, readADC1Min)
    # Request nodes to do ACTION
    ecMap(ACTION, n)
    GH = ecGroup(getNodes())
    ecMap(setSamplesPerSec, GH, [20] * len(GH))
    task = repeated(1, PEG)  # obj to spawn a new thread / sec
    task.start()
    time.sleep(60)
    task.stop()
Naming conventions for dynamic types

<table>
<thead>
<tr>
<th>Prefix</th>
<th>C_</th>
<th>UC_</th>
<th>I_</th>
<th>UI_</th>
<th>F_</th>
</tr>
</thead>
<tbody>
<tr>
<td>C Type</td>
<td>char</td>
<td>unsigned char</td>
<td>int</td>
<td>unsigned int</td>
<td>float</td>
</tr>
</tbody>
</table>
A network discovery example
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Size</th>
<th>Iterative command</th>
<th>Macroprogramming statements</th>
</tr>
</thead>
</table>
| TEMP    | Read the output of temperature sensor and return to host.                   | 202  bytes | \( g = \text{getNodes}([0,1,2,3,4,5,6,7,8,9]) \)
for \( n \) in \( g \):
\( n.\text{readTemp}() \) | \( g = \text{ecGroup}([0,1,2,3,4,5,6,7,8,9]) \)
\( \text{ecMap(readTemp,g)} \) |
| ADC     | Reads the output value for the digitalized output of triaxial accelerometer return to host. | 212 bytes | for \( n \) in \( g \):
\( n.\text{readADC}() \) | \( \text{ecMap(readADC,g)} \) |
| ADC_MIN | Samples the output value of triaxial accelerometer in 25Hz for one minute and return the average of the outputs. | 372 bytes | No iterative version | \( \text{ecMap(readADCLMin,g)} \) |
| LIGHTING | Control the switch of the light(on/off)                                    | 12   bytes | for \( n \) in \( g \):
\( n.\text{turnOn}() \)
\( \text{time.sleep(5)} \)
for \( n \) in \( g \):
\( n.\text{turnOff}() \) | \( \text{ecMap(turnOn, g)} \)
\( \text{time.sleep(5)} \)
\( \text{ecMap(turnOff, g)} \) |
Long time execution test
ADC\textsubscript{1}min

(a) Response time
(b) Ratio of different phases
25

Challenge the future

(a) Light control       (b) Door monitoring

The latency of adding functions