DESAL$^\alpha$: An Implementation of the Dynamic Embedded Sensor-Actuator Language


Presented by: Sally K. Wahba

Dependable Systems Research Group, School of Computing

CLEMSON UNIVERSITY

August 6, 2008
Presentation Roadmap

1. Introduction
2. DESAL$^\alpha$ Features
3. Language Design
4. Runtime Architecture
5. Application Examples
6. Resource Requirements
7. Conclusion and Future Work
8. Questions
Introduction

**What are wireless sensor networks?**
Collections of devices that monitor their environments and communicate to cooperatively solve problems

**What are they used for?**
Introduction (cont’d)

Why are they interesting?

- Enable a new class of applications
- Resource-constrained
  - Computation
  - Memory
  - Storage
  - Power
- Unique programming language and operating system
  - NeSC
  - TinyOS
Introduction (cont’d)

What are (some of) the problems?

- Mismatch between target users and tools
- Significant knowledge of:
  - Low-level programming primitives
  - Operating system components
  - Network protocols

Observation

A simpler language is needed
DESAL$^\alpha$ Features

Main Features
- Rule-based programming
- Synchronized action scheduling
- Neighborhood management
- Distributed state sharing
Abbreviated DESAL\(^\alpha\) Grammar

1. \( \langle \text{program} \rangle \rightarrow \text{component} \ \langle \text{cid} \rangle \ \langle \text{subcmpnts} \rangle \)
2. \( \langle \text{subcmpnt} \rangle \rightarrow \langle \text{vars} \rangle \ \langle \text{bindings} \rangle \ \langle \text{body} \rangle \)
3. \( \langle \text{var} \rangle \rightarrow \ [\text{unshared} | \text{shared}] \ \langle \text{dec} \rangle \)
4. \( \langle \text{dec} \rangle \rightarrow \langle \text{type} \rangle \ \langle \text{id} \rangle \)
5. \( \langle \text{binding} \rangle \rightarrow \text{binding} \ (\langle \text{sbinding} \rangle | \langle \text{mbinding} \rangle) \)
6. \( \langle \text{sbinding} \rangle \rightarrow \langle \text{dec} \rangle \leftarrow \langle \text{nid} \rangle \ . \langle \text{cid} \rangle \ . \langle \text{vid} \rangle \)
7. \( \langle \text{mbinding} \rangle \rightarrow \langle \text{dec} \rangle \leftarrow \ "\*\" . \langle \text{cid} \rangle \ . \langle \text{vid} \rangle"["\langle \text{num} \rangle"] \)

Productions Summary

- Program declaration
- Variable declaration
- Subcomponents
- Singleton binding
- Multi-binding
## Abbreviated DESAL$^\alpha$ Grammar

<table>
<thead>
<tr>
<th>Production</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. $\langle body \rangle \rightarrow \textbf{every} \ \langle num \rangle \ \langle tunit \rangle \ \textbf{after}$</td>
</tr>
<tr>
<td>2. $\langle num \rangle \ \langle tunit \rangle \ \langle gactions \rangle$</td>
</tr>
<tr>
<td>3. $\langle tunit \rangle \rightarrow \textbf{ms}</td>
</tr>
<tr>
<td>4. $\langle gaction \rangle \rightarrow \langle guard \rangle : \langle stmtlst \rangle$</td>
</tr>
<tr>
<td>5. $\langle guard \rangle \rightarrow \langle boolexpr \rangle$</td>
</tr>
<tr>
<td>6. $\langle forstmt \rangle \rightarrow \textbf{foreach} \ \langle vid \rangle \ \textbf{in} \ \langle bid \rangle \ { \langle stmtlst \rangle }$</td>
</tr>
</tbody>
</table>

### Productions Summary

- Body construct
- Time units
- Guarded actions
- For loop construct
Abbreviated DESAL\(^\alpha\) Grammar

1. \[\langle \text{bndfunc} \rangle \rightarrow \text{bound}(\langle \text{bid} \rangle) \mid \text{src}(\langle \text{bid} \rangle) \mid \text{age}(\langle \text{bid} \rangle)\]
2. \[\langle \text{sensfunc} \rangle \rightarrow \$\text{temp}() \mid \$\text{humid}() \mid \$\text{tsr}() \mid \$\text{par}() \mid \]
   \[\$\text{adc0}() \mid \$\text{adc1}() \mid \$\text{volt}()\]
3. \[\langle \text{actfunc} \rangle \rightarrow \$\text{redOn}() \mid \$\text{redOff}() \mid \$\text{redToggle}() \mid \]
   \[//..analogous for blue, yellow.. \mid \]
   \[\$\text{gio0On}() \mid \$\text{gio0Off}() \mid \]
   \[\$\text{gio1On}() \mid \$\text{gio1Off}()\]

Productions Summary

- Binding functions
- Sensor functions
- Actuation functions
Runtime Architecture (cont’d)

Application Architecture

- TinyOS compilation target
- TinyThread
- Statements and expressions syntaxes
- Configurable architecture through a range of parameters
## Application Examples

### Blink

```
component Blink
    every 3s after 0s
        true: $redOn()
    every 3s after 1s
        true: $redOff()
```
Application Examples (cont’d)

MaxTemp

```plaintext
component MaxTemp
  shared  uint16 max = 0
  binding uint16 nmax <- *.MaxTemp.max[20]
  every 1s after 0s
    $temp() > max: max = $temp()
      [ ]
    true:
      foreach n in nmax
        { if (n > max) { max = n } }
  every 30s after 0s
    true: max = 0
```
Resource Requirements

Shared Variable Overhead

<table>
<thead>
<tr>
<th>Bytes</th>
<th>Empty Body</th>
<th>bool, uint[8/16] (1)</th>
<th>bool, uint[8,16] (10)</th>
<th>bool, uint[8/16] (100)</th>
<th>uint32 (1)</th>
<th>uint32 (10)</th>
<th>uint32 (100)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2000</td>
<td>4000</td>
<td>6000</td>
<td>8000</td>
<td>10000</td>
<td>12000</td>
<td>14000</td>
</tr>
</tbody>
</table>

Chart showing resource requirements for ROM and RAM with different data types.
Resource Requirements (cont’d)

Singleton Bindings Overhead

<table>
<thead>
<tr>
<th>Bytes</th>
<th>ROM</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty Body</td>
<td>bool, uint[8/16] (1)</td>
<td>bool, uint[8/16] (10)</td>
</tr>
<tr>
<td></td>
<td>uint32 (1)</td>
<td>uint32 (10)</td>
</tr>
</tbody>
</table>
Resource Requirements (cont’d)

Multi-Bindings Overhead

<table>
<thead>
<tr>
<th>Bytes</th>
<th>ROM</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty Body</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bool, uint[8/16] (1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bool/uint[8/16] (10)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>uint32 (1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>uint32 (10)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Conclusion and Future Work

Conclusion
- Mismatch between target users and tools
- The need for DESAL$^\alpha$
- A simple yet powerful language

Future Work
- Composite bindings
- DESAL$^\alpha$ extensions
- Multi-hop bindings
DESAL$^\alpha$: An Implementation of the Dynamic Embedded Sensor-Actuator Language


Presented by: Sally K. Wahba

Dependable Systems Research Group, School of Computing

Clemson University