Resource Discovery with Evolving Tuples

Drew Stovall and Christine Julien
The University of Texas at Austin
{dstovall, c.julien}@mail.utexas.edu

Presented at:
Engineering of Software Services for Pervasive Environments (ESSPE '07)
(at ESEC/FSE 2007 - Dubrovnik, Croatia, September 4, 2007)
Overview

- Target environment
- Framework support for pervasive services
- Existing tuple models
- Evolving tuples
- Discovery - a sample pervasive service
- Discovery with evolving tuples
- Open Questions
Target Environment

- Heterogeneous connections
  - 802.11, Zigbee, BlueTooth, proprietary

- Heterogeneous capabilities
  - Processor(s), Memory, Non-volatile storage, Sensors

- Heterogeneous OS and software
  - Linux, TinyOS, proprietary

- Power provided by batteries
Target Environment

- We target the lowest capability nodes and networks
  - Sensor class hosts
    - Slow processors
    - Small memory
    - Small nonvolatile storage (flash)
  - Ad-hoc networks
    - Datagram based
    - With or without ACK
    - Max payload ~10’s of bytes
    - Frequent connectivity changes
Frameworks for Pervasive Services

- Software engineering and services rely on frameworks to provide or support certain characteristics
  - Autonomous Action
  - Fully Distributed
  - Open
  - Localized
  - Best Effort
  - Context-Aware
  - Compatible
  - Implementable

- Tuples can provide a framework for these services
Tuples

- **Original Tuple Model** (Gelernter & Bernstein – 1982)
  - **Tuple** = name + ordered list of values
    < name, value, value, ... >
    <“ping”, 10, 5, 3>
  - **Pattern** = name + ordered list of predicates
    <“ping”, 10, i:integer, j:integer>
  - **Tuple Space** = bag of tuples
    - Add / Peek / Remove operations

- Tight design-time coupling between producers and consumers
Tuples

- Dictionary-like tuples (ELights / Lights)
  - **Tuple** = unordered set of name/value pairs
    - < (name, value), (name, value), ... >
    - < (msg_type=“ping”), (destination=10), ..., (ttl=3) >
  - **Pattern** = unordered set of name/predicates
    - < (msg_type,“ping”), (ttl, “?integer”)>

- Values are static in transit
- Producers rely on consumer behavior
Tuples

- Want general mechanism to process context data
  - Send
  - Collect
  - Aggregate
  - Combine

- Want behavior that can be redeployed or updated
Evolving Tuples

- Embed some of the service behavior into the network messages
- Builds on dictionary-based approach
- Adds the *formula* element to each field
- Adds the *evolution context*
Evolving Tuples

- The *formula* element
  
  \[
  < \text{(name, value, formula)}, \text{(name, value, formula)}, \ldots > \\
  < \text{(oneHourAgo, 12, context\[hour\] – 1)} >
  \]

- Used to automatically update the values of a tuple field

<table>
<thead>
<tr>
<th>Operators</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ - * /</td>
<td>Arithmetic</td>
</tr>
<tr>
<td>&lt; ≤ &gt; ≥ = !=</td>
<td>Comparison</td>
</tr>
<tr>
<td>! &amp;&amp;</td>
<td></td>
</tr>
<tr>
<td>if (x,y,z)</td>
<td>Conditional</td>
</tr>
<tr>
<td>name</td>
<td></td>
</tr>
<tr>
<td>context[\text{name}]</td>
<td>Lookup</td>
</tr>
</tbody>
</table>
Evolving Tuples

- The evolution context is a dictionary of environmental values
- Provides access to host and application provided values

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>hour</td>
<td>15</td>
</tr>
<tr>
<td>month</td>
<td>september</td>
</tr>
<tr>
<td>year</td>
<td>2007</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Evolving Tuples

\[
\text{context[hour]} = 15
\]

\[
< \text{oneHourAgo}, 12, \text{context[hour]} - 1 > \quad < \text{oneHourAgo}, 14, \text{context[hour]} - 1 >
\]
Standard Deployment

Host

“application” Tuple Space

Host Application

“outbound” Tuple Space

“inbound” Tuple Space

“Receive()” Process

Node Info
Evolving Tuples

- Dynamic values and behavior enables evolving tuples to be used for a variety of pervasive services
  - Discovery
  - Routing
  - Data dissemination
  - Data collection and aggregation
  - Remote Procedure Call
  - Instant messaging
Discovery

- Finding implementations of interfaces
- One of the initial services required in pervasive environments
Discovery by Evolution

- Resource description as evolution context

![Diagram showing a printer with attributes: resource-type "printer", color true, paper-size "letter". Arrows pointing to a box labeled "Evolution".]
Discovery by Evolution: An Example

- Looking for a “printer”
  - context[resource-type]="printer"

- Addresses
  - -1 = Broadcast
  - Null = No address, drop
Discover by Evolution: An Example

<table>
<thead>
<tr>
<th>source</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>latency</td>
<td>0</td>
</tr>
<tr>
<td>match</td>
<td>null</td>
</tr>
<tr>
<td>destination</td>
<td>-1</td>
</tr>
</tbody>
</table>

if ( match != null,  
    if( match = true, source, null ),  
    if( latency < 0.1, -1, null )  
)
Discovery Example

Client

Host 1

Host 2

Drew Stovall, ESSPE 2007
Univ. of Texas at Austin
4-September-2007
Open Questions

- Performance vs. Capability trade-off
- Prototyping protocols and applications with evolving tuples?
- What is the “right” coordination model for Pervasive Services?
  - What is the Lowest Common Denominator amongst services and hosts?
  - Including sensor-class nodes
- Can (and how) do we collaborate between administrative domains?
Resource Discovery with Evolving Tuples

Drew Stovall and Christine Julien
The University of Texas at Austin
{dstovall, c.julien}@mail.utexas.edu
http://mpc.ece.utexas.edu/evolving-tuples