REP:
A Communication Mechanism for Pervasive Computing

Janet Davis
May 25, 2001
Pervasive computing

- Computation everywhere
- Focus on tasks, not technology
- Dynamic, ad-hoc distributed system
- Challenge: build adaptive applications
- Communication is key
Problem statement

We need an appropriate high-level communication mechanism for pervasive computing.
**RPC**

- Common communication model
- Communication looks like procedure call
- Easy to understand and use
RPC?

- Transparent
- Fixed interface
- Only one of many possible models
Remote event passing (REP)

- Asynchronous, typed message passing

- Expose communication
  - encourage adaptation
  - provide mobility transparency
REP continued

- Support dynamic composition and evolution
  - Uniform event-handling interface
    
    ```java
    public void handle(Event e)
    ```
  - Passive, semi-structured data
- Provide useful, but general, primitives
  - Natural when reacting to environment
  - General enough to implement other protocols
Naming

Specific  RemoteReference

NamedResource

General  DiscoveredResource
Operations

- export
- send
- resolve
Implementation

- one.world service
- Relies on structured I/O
- Caches connections
Latency

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>RMI</th>
<th>REP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty call</td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td>Simple event</td>
<td>3.0</td>
<td>3.8</td>
</tr>
<tr>
<td>Large event</td>
<td>3.4</td>
<td>4.2</td>
</tr>
<tr>
<td>Complex event</td>
<td>3.8</td>
<td>5.5</td>
</tr>
</tbody>
</table>

Round-trip latency in milliseconds
## Throughput

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>RMI</th>
<th>REP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty call</td>
<td>4330</td>
<td></td>
</tr>
<tr>
<td>Event without return</td>
<td>833</td>
<td>881</td>
</tr>
<tr>
<td>Event with return</td>
<td>609</td>
<td>751</td>
</tr>
</tbody>
</table>

Server throughput in calls or events per second
Effectiveness and usability

- User study comparing REP and RMI/Jini
- Our experiences
- Topics:
  - transparency
  - communication model
  - composition and interface evolution
  - programming with events
Transparency

- RMI/Jini
  - getting started was easy, but
  - considered failures after the fact
- REP
  - considered failures from the beginning
- Late binding is useful
Communication model

- Events aren’t hard to understand
- Used even with RMI/Jini
- General enough to implement other semantics
  - reliable delivery
  - request/response interactions
  - forwarding/short-circuiting
Composition and interface evolution

- Uniform interface simplified development
- Experiment did not address interface evolution
Programming with events

- Easy with simple control flow
- Helps manage concurrency
- Unfamiliar programming model
  - complex control flow is difficult
- Need design patterns
  - e.g., logic/operation
Future work

- Further evaluation by building applications
- Internode debugging
- Lightweight REP
- Virtualization
Conclusion

- REP…
  - encourages early consideration of distribution
  - provides mobility transparency
  - supports composition (and interface evolution?)
  - is useful, but general
Acknowledgements

- The *one.world* team:
  Robert Grimm, Eric Lemar,
  Adam MacBeth, Steve Swanson

- My advisors:
  Tom Anderson and Steve Gribble