Resource-Controlled Remote Execution to Enhance Wireless Network Applications

Travis Newhouse, Joseph Pasquale

{newhouse,pasquale}@cs.ucsd.edu
Wireless Applications Can Benefit From Wired Computational Resources

Goal:

- Enhance wireless applications without requiring additional resources on mobile device

Benefits:

- Reduce effects of Internet’s “best effort” design
- Transform data “designed for the desktop” to suit mobile device platforms
- Supplement mobile device hardware resources (e.g. processor, memory, battery life)
The Extension Model for Remote Execution

Extension is code that implements application-specific functionality to extend control of an endpoint.

Extension is loaded on demand by the remote node.

Non-extended endpoint need not be modified.
Java Active Extensions System

A user-level system for remote execution

Supports processor quality of service

User-level implementation supports incremental deployment

System design supports scalability of hardware resources
Resource-Controlled Remote Execution

4 phases:
- Discovery – locate an extension system in the network
- Resource allocation – request processor resources for execution
- Deployment – load extension(s)
- Execution – extension runs and communicates with endpoint
Quality of Service

Resource requests are made in the form: \( \langle q u a n t a , p e r i o d \rangle \)

Extension systems advertise quantum length

Endpoints specify the period over which resources are guaranteed
Endpoint-Extension Communication

Message passing mechanism provided by extension system
Intended for bootstrapping application-specific communication
Messages are arbitrary, application-formatted sequences of bytes
Message delivery follows in-order and at-most-once semantics
Extension System Architecture

- **Endpoint**:
  - allocate
  - deallocate
  - load

- **Manager**:
  - authorize
  - resource policy module

- **Extension Server**
  - Extension

Manager enforces locally-defined policy for resource sharing
- Responds to endpoints’ resource requests by creating extension servers
- Schedules processor resources among extension servers

Extension Server provides execution environment
- Hosts the execution of one or more extensions
- Executes extensions with pre-allocated share of processor resources
Each extension server implemented as a separate JVM
User-level scheduler operates on UNIX-like operating system
Groups of processes scheduled by the user-level scheduler
Kernel makes fine-grained scheduling decisions
## Cost of Basic Operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Mean (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discover manager using Jini</td>
<td>770 ± 1</td>
</tr>
<tr>
<td>Discover manager using socket</td>
<td>542 ± 1</td>
</tr>
<tr>
<td>Allocate extension server</td>
<td>764 ± 4</td>
</tr>
<tr>
<td>Load extension</td>
<td>315 ± 3</td>
</tr>
</tbody>
</table>

Mean operation time with a 99% confidence interval over 1000 trials
Test machine is dual 600 MHz Pentium III, 1 GB memory, Solaris 8
1000 trials performed over a local 100 Mbps Ethernet
Sending a message is comparable to a retrieving a local Web object
Overhead and Accuracy of User-Level Scheduler

Travis Newhouse, UCSD
Conclusions

The extension model for remote execution can enhance wireless network applications

Java Active Extensions system provides remote execution with processor quality of service

The system architecture supports scalable hardware resources

The user-level implementation supports ease of deployment and abstracts hardware details