DACIA: A Mobile Component Framework for Building Adaptive Distributed Applications

Radu Litiu and Atul Prakash
University of Michigan, EECS
Outline

- Motivation and Goals
- DACIA Architecture
- Performance Measurements
- Related Work
- Conclusions and Future Work
Need for Adaptation - SPARC Collaboratory

An application is a graph of connected components.

Possible changes:

- Execute the computation on the client machine
- Store computed images instead of raw data
- Add/remove modules
Why Reconfiguration?

- Environment: Pentium II 200, Ultra SPARC 1
- Raw data size / computed image size = 1/2
- Compute time = 5 msec/kB (fast machine)
  15 msec/kB (slow machine)

Adaptability and reconfiguration can be useful
Mobility

application
move
Design Goals

- Adapt to variability
- Runtime reconfiguration
- Application and user mobility
- Persistent connectivity between components
- Low communication overhead
- Ease of use
DACIA* Architecture

Engine (mechanism)
- Communicate between hosts
- Manage connections between components
- Relocate components
- Reconfigure the application

Monitor (policy)
- Monitor performance
- Make reconfiguration decisions
- Implement application-specific reconfiguration policies
PROC - Processing and Routing Component

- Communication through ports
- Key goal: low communication costs

Synchronous communication

Asynchronous communication

Message queue

Data
Communication Performance

- Environment: Pentium II 200, Ultra SPARC 1, 10 Mbps LAN
- Latencies (µsec) for inter-PROC communication and raw TCP

<table>
<thead>
<tr>
<th>message size (bytes)</th>
<th>local PROCs synchronous</th>
<th>local PROCs asynchronous</th>
<th>local procedure call</th>
<th>local TCP</th>
<th>Remote PROCs</th>
<th>Remote TCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>6.6</td>
<td>44</td>
<td>6.4</td>
<td>370</td>
<td>2040</td>
<td>990</td>
</tr>
<tr>
<td>1000</td>
<td>6.6</td>
<td>44</td>
<td>6.4</td>
<td>400</td>
<td>3900</td>
<td>2600</td>
</tr>
</tbody>
</table>

- Throughput (message size = 1-5 kB)
  - DACIA: 4.78-5.33 Mbps
  - TCP (Java): 5.35-6.61 Mbps
Component Mobility

- Transfer the PROC's state
  - data
  - message queue
- Implicit/explicit state capture
- Movement at well-defined times

- Cost of PROC movement - 121 msec (size = 788 bytes)
  - Java serialization cost
- Component mobility more effective for long-term environment changes
Connectivity

- Multiplex virtual connections between PROCs
  - Low cost to establish connections
  - Hide temporary network failures
- Persistent connectivity between moving PROCs
  - Messages buffered or forwarded
  - Dissemination of PROC location information
Dynamic Application Reconfiguration

- Change connections between components
- Change components’ location
- Load new components

An adaptive application: multi-party communication
Reconfiguration Mechanisms

- Specialized monitors

- Command-line interface:
  - `connect [hostname] [portnumber]`
  - `connectProcs [sourceProcID] [sourcePortNo] [destProcID] [destPortNo]`
  - `disconnectProcs [sourceProcID] [sourcePortNo]`
  - `move [procID] [hostname]`
  - `start [procID]`
  - `startMonitor`
Related Work

- Distributed component architectures: CORBA, Globus, Darwin, Scout
- Code mobility & mobile agents: Telescript, Obliq, Sumatra, Tacoma, Aglets, FarGo
- Mobile environments: Rover, Daedalus/Barwan, GloMop
- Adaptive systems: Odyssey, Conductor
Conclusions

- DACIA - a framework for building adaptive distributed applications
- Dynamic reconfiguration can improve the performance of the application
- Low-cost connectivity
- Application and user mobility
- Persistent connectivity between mobile components
Current and Future Work

- Policies and algorithms for application reconfiguration
- Formalism for specifying components and composition rules
- Deployment and experimentation
- Security infrastructure

http://www.eecs.umich.edu/~radu