An Approach to Manage Reconfiguration in Fault-Tolerant Distributed Systems

Stefano Porcarelli¹, Marco Castaldi², Felicita Di Giandomenico¹, Andrea Bondavalli³, Paola Inverardi²

¹ Italian National Research Council, ISTI Dept, Italy
   stefano.porcarelli@guest.cnuce.cnr.it, digiandomenico@iei.pi.cnr.it
² University of L’Aquila, Dip. Informatica, Italy
   {castaldi, inverard}@di.univaq.it
³ University of Florence, Dip. Sistemi e Informatica, Italy
   a.bondavalli@dsi.uni.it

May 3rd, WADS 2003
Motivations

- Large distributed systems live for several years
- Environmental events and component’s faults may affect workload and functionalities of the system
- High availability and reliability of critical systems

System reconfiguration to react to faults, to manage system’s life and to provide dependability properties
System Reconfigurations

- **Dynamic**: the reconfiguration must be performed while the system is running, without service interruption

- **Automatic**: the reconfiguration may be triggered as a reaction for a specified event, issued by a human administrator or an automatic Decision Maker

- **Distributed**: the reconfiguration is performed on distributed systems

In particular, we address:

- **Component Reconfiguration**: any change of the component parameters (component re-parametrization)

- **Application Reconfiguration**: any architecture’s modification in terms of topology, component’s number and location
Our Approach to (Fault) Reconfiguration

- We propose to use Lira, an infrastructure created to perform dynamic reconfiguration, enriched with a model-based Decision Maker.

Managed System

- Lira monitors the system, detects faults and notifies the Decision Maker.

Decision Maker

- For each fault pattern, a set of reconfigurations is specified.
- DM performs the evaluation.
- DM orders the reconfiguration.
- Lira reconfigures the system.
Our Approach to (Fault) Reconfiguration

• The decision making capability is decomposed in a hierarchical fashion:
  – Favoring fault-tolerance by distribution of control
  – Avoiding heavy computation and coordination activity whenever faults can be managed at local level
  – Facilitating the construction and on-line solution of analytical models
  – Favoring scalability
Lira Architecture

- **Lira Management Infrastructure**
  - Light-weight Infrastructure for Reconfiguring Applications
  - Lira is based on:
    - Agents
    - MIB (Management Information Base)
    - Management Protocol
Enriched Lira Architecture

- **Lira uses a different agent for each hierarchical level:**
  - Component, Host, Application, Manager agent
- **Each agent is enriched with a decision maker**
  - Decision making capabilities depend on the hierarchical level of the agent
**Decision Maker**

- **Model-Based Decision Maker**
  - The dynamic topology of the system and the number of managed faults demand for statistical decisions capabilities.
  - Combinatorial and Petri net like models (for complex relationships among components) help to take the most appropriate decision.
  - The possible reconfiguration options are pre-planned: models allow deciding each time which is the most appropriate one.

The component’s state is modeled by using three states:

- **Up**
- **Degraded**
- **Down**
A Case Study

- Distributed computing where peer-to-peer clients on the network are communicating
- Path redundancy is used to prevent service’s interruption

<table>
<thead>
<tr>
<th>Path</th>
<th>Route</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a-N₁-c-N₃-f</td>
</tr>
<tr>
<td>2</td>
<td>a-N₁-c-N₃-d-N₂-e-N₄-g</td>
</tr>
<tr>
<td>3</td>
<td>b-N₂-e-N₄-g</td>
</tr>
<tr>
<td>4</td>
<td>b-N₂-d-N₃-f</td>
</tr>
</tbody>
</table>
A Case Study (cont)

- **Component agent**
  - HEALTH
  - CONNECTED_NODE
  - Function to connect different nodes
  - Functions to control the node

- **Host agent**
  - HEALTH
  - CONNECTED_HOST
  - Functions to install and activate nodes

- **Application Agent**
  - AVAILABLE
  - ACTIVE_NODES
  - ACTIVE_HOSTS
  - Functions provided by the Host agents

- **Manager Agent**
  - ACTIVE_HOSTS
  - Functions provided by the Application agents
An Example

- Let suppose that node $N_3$ starts to work in degraded manner
- The associated agent $A_3$ notifies at the upper level $AA_1$
- The agent $AA_1$ checks the path availability on the controlled network
- Three different reconfiguration options are possible:
  - Continuing to work in degraded manner
  - Temporarily bypassing node $N_3$ and waiting for its restart
  - Activate a new node for substituting $N_3$
An Example

• Three different reconfiguration options are possible:
  - Continuing to work in degraded manner
  - Temporarily bypassing node $N_3$ and waiting for its restart
  - Activate a new node for substituting $N_3$

• The best reconfiguration consists in restarting $N_3$

<table>
<thead>
<tr>
<th>Link or component status</th>
<th>Failure Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Up state</td>
<td>$10^3$</td>
</tr>
<tr>
<td>Degraded state</td>
<td>$10^2$</td>
</tr>
<tr>
<td>Restarted and new</td>
<td>$5 \times 10^3$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Policy Options</th>
<th>$P_F$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Working in degraded manner</td>
<td>$1.73848 \times 10^8$</td>
</tr>
<tr>
<td>Restart node $N_3$</td>
<td>$5.19695 \times 10^9$</td>
</tr>
<tr>
<td>Set-up a new path</td>
<td>$4.77510 \times 10^{-8}$</td>
</tr>
</tbody>
</table>
Conclusions

• An architecture for dependability provision has been proposed. It is based on:
  – Lira
  – Model-based Decision Maker

• We concentrate on system reconfiguration as consequence of faults (both sw and hw)

• Hierarchical approach
Future Work

• Lira infrastructure has to be fault-tolerant itself

• Development of Petri net based decision maker (combinatorial models are not able to handle complex scenarios)
  – Dependencies among components
  – Account for Time
  – Repairing of components

• Development of a prototype
  – Experimental measurements