An Approach to Manage Reconfiguration in Fault-Tolerant Distributed Systems

Stefano Porcarelli[1], Marco Castaldi[2], Felicita Di Giandomenico[1], Andrea Bondavalli[3], Paola Inverardi[2]

[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

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

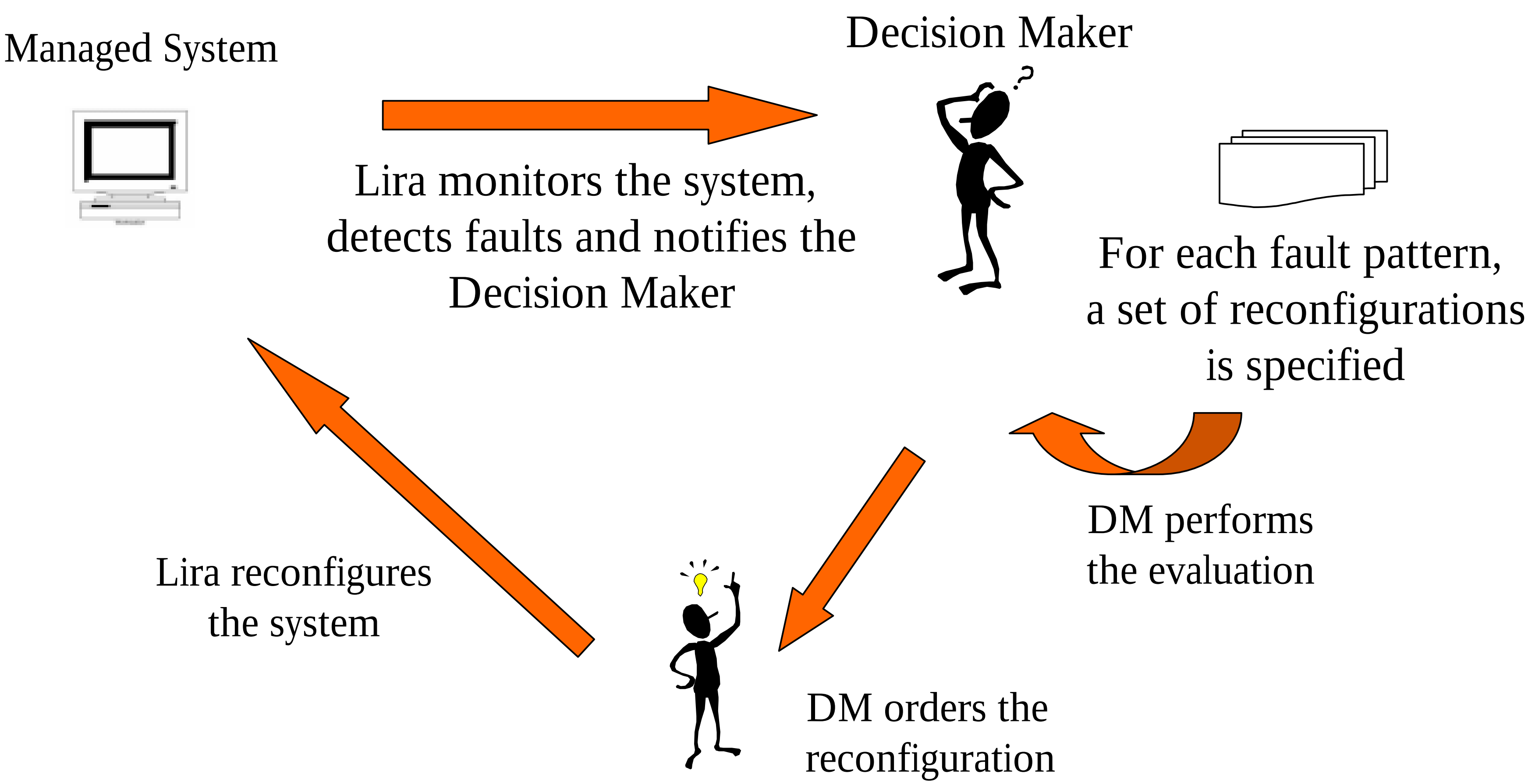
- **Dynamic:** the reconfiguration must be performed while the system is running, without service interruption
- **Automatic:** the reconfiguration may be triggered as a reaction for a specified event, issued by a human administrator or an automatic Decision Maker
- **Distributed**: the reconfiguration 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**

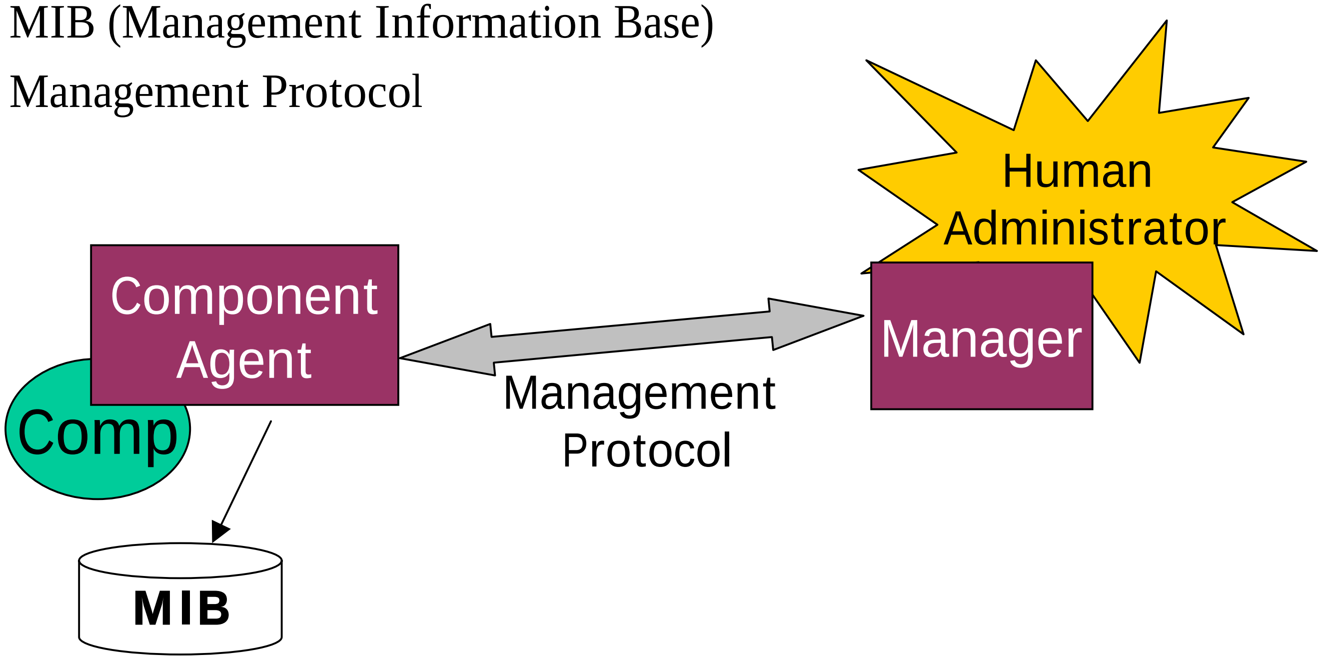


Our Approach to (Fault) Reconfiguration

- 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

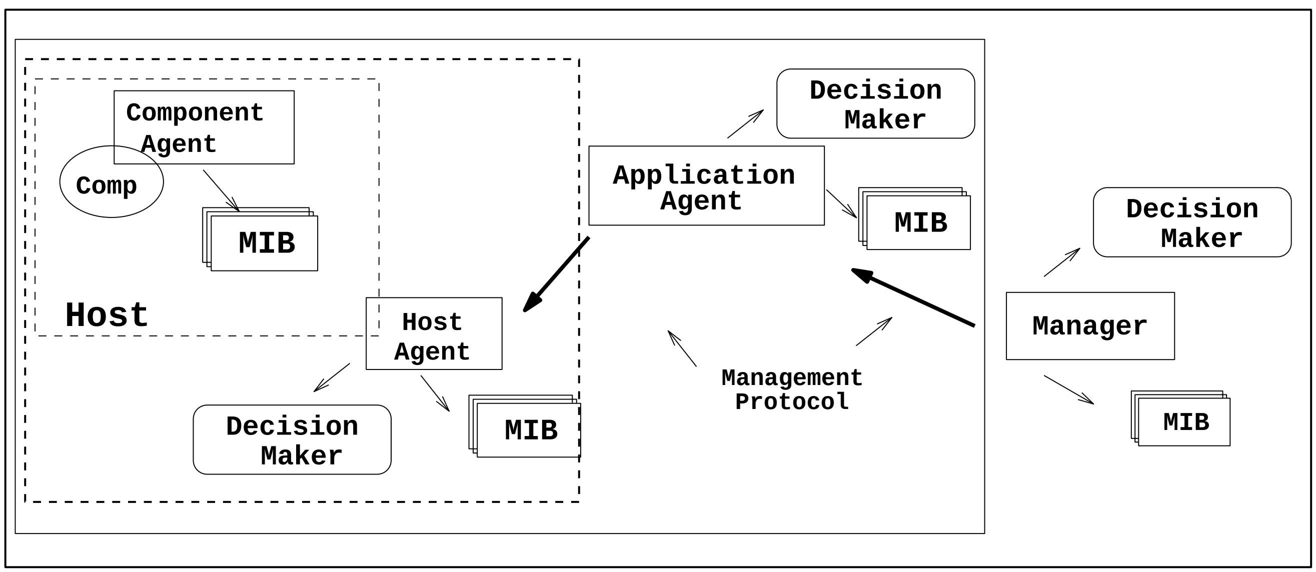
Lira Architecture

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



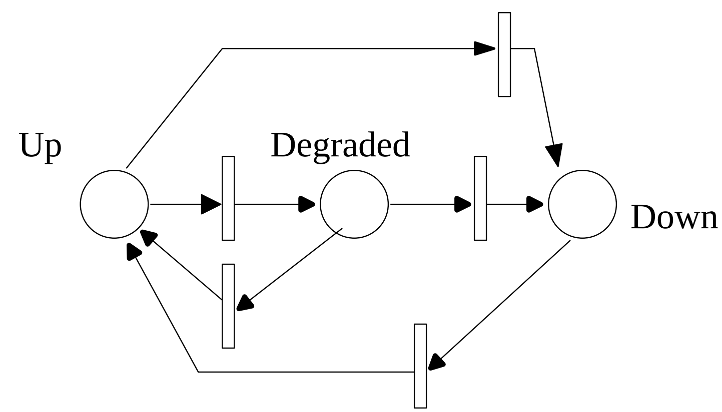
Enriched Lira Architecture

- **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

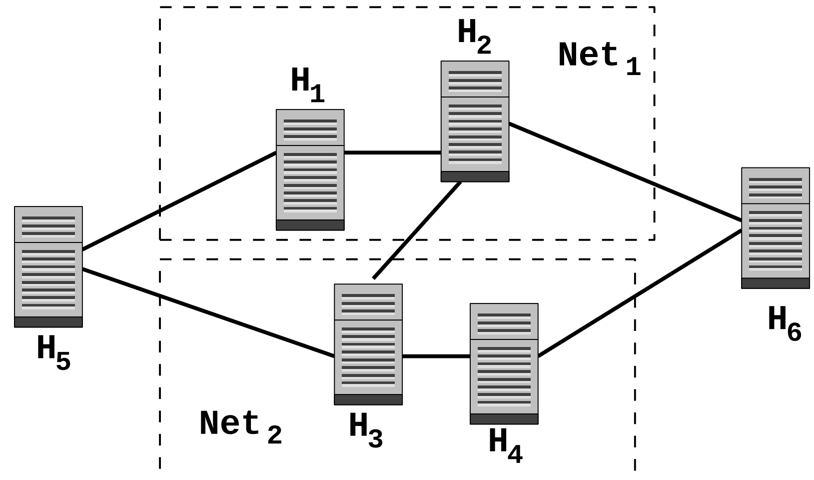


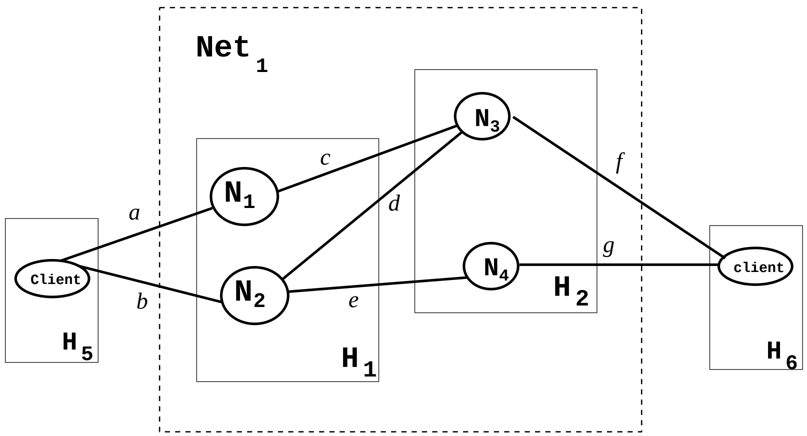
The component's state is modeled by using three states :

- 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

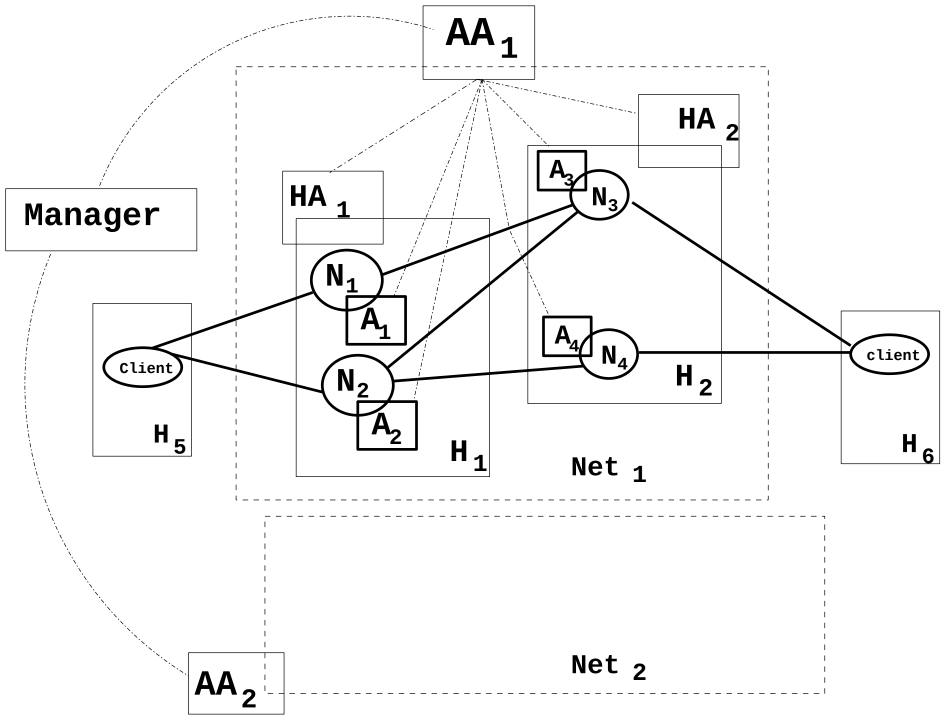




Path	Route
1	**a-N_1-c-N_3-f**
2	**a-N_1-c-N_3-d-N_2-e-N_4-g**
3	**b-N_2-e-N_4-g**
4	**b-N_2-d-N_3-f**

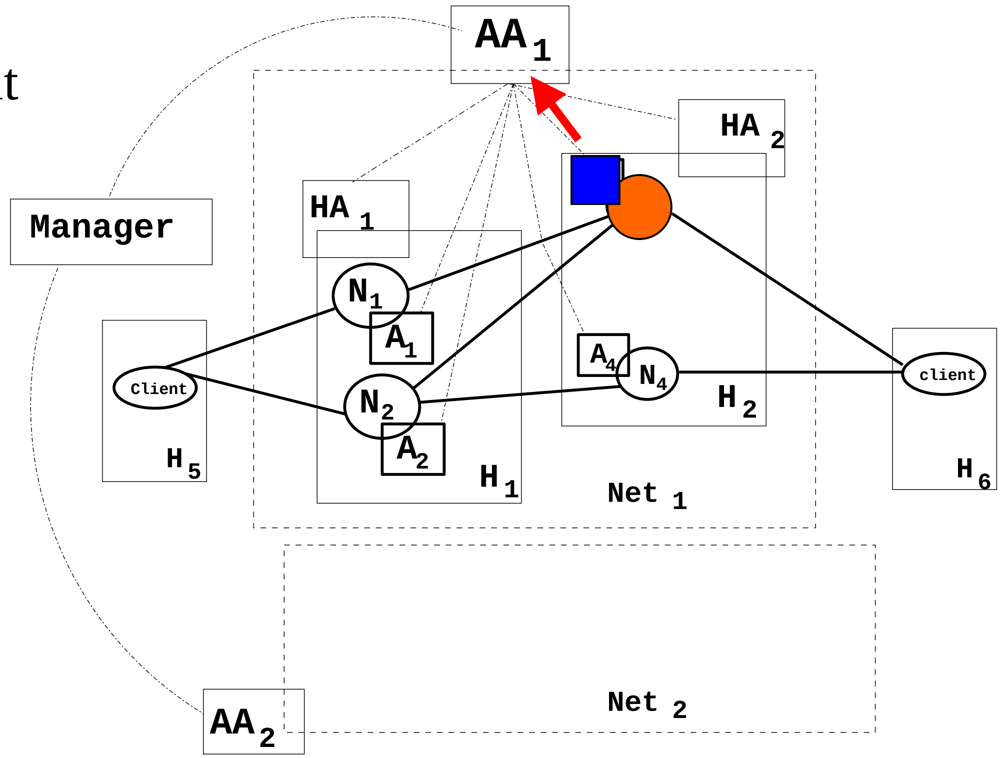
A Case Study (cont)

- **Component agent**
  - HEALTH_STATE
  - CONNECTED_NODE
  - Function to connect different nodes
  - Functions to control the node
- **Host agent**
  - HEALTH_STATE
  - CONNECTED_HOST
  - Functions to install and activate nodes
- **Application Agent**
  - AVAILABLE_PATHS
  - 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 $\mathbf{N_3}$ 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**

Link or component status	Failure Probability
Up state	10^{-3}
Degraded state	10^{-2}
Restarted and new	$5 * 10^{-3}$

Policy Options	P_F
Working in degraded manner	$1.73848 * 10^{-8}$
Restart node N_3	$5.19695 * 10^{-9}$
Set-up a new path	$4.77510 * 10^{-8}$

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