Ruling Agent Motion in Structure Environments

Marco Cremonini, Andrea Omicini

LIA – DEIS – Università di Bologna - Italy

{mcremonini, aomicini}@deis.unibo.it

Franco Zambonelli

DSI - Università di Modena e Reggio Emilia - Italy

franco.zambonelli@unimo.it

HPCN Europe 2000 - Amsterdam 8-10 May 2000

Outline

- **Structured Environments: The Virtual Enterprise Case**
- **Dynamic Knowledge Acquisition**
- **TuCSoN Topology and Coordination Model**
- **Dynamic Exploration Constraining**
- **Security Issues**

Mobile Agents

Basic Characteristics:

- ***Agency***: software entities autonomously acting to achieve a goal.
- ***Proactivity***: ability to plan the operational steps according with a goal
- ***Reactivity***: ability to sense the environment status and respond to its changes
- ***Mobility***: the ability of a system to allow the migration of both the code AND the execution state of a mobile agent

Motivation

- Save bandwidth/computational resources, Flexibility, System Extensibility

Application Areas

- Cooperative applications, Auctions, Mobile Computing, Information Retrieval

Virtual Enterprises

- components are ***independent entities*** (companies, branches, groups) that federate to develop or support a given project/service/system.

- a virtual enterprise appears to external entities (users, clients, etc.) as an ***homogeneous and unique system***.

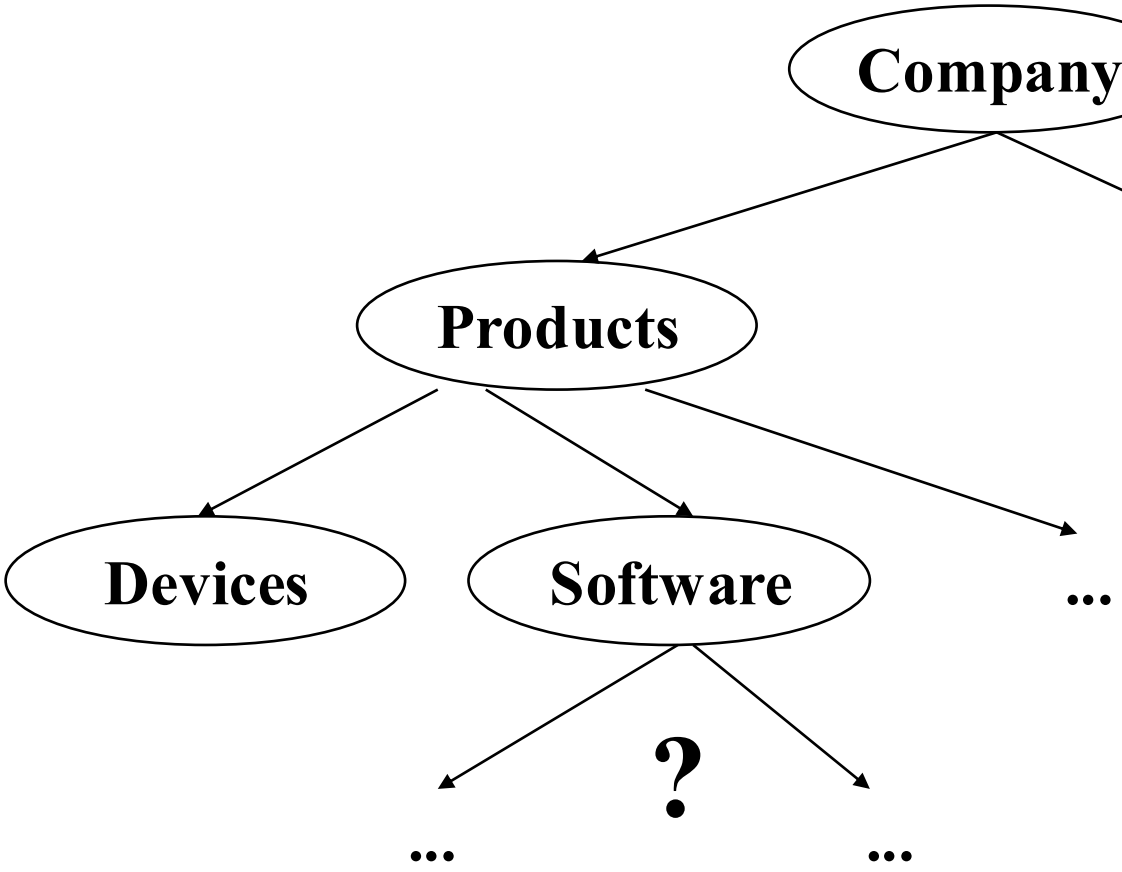
Assumptions

- both the resource availability and the organisation topology should be considered, in general, *a-priori unknown* or only *partially known*.
- destinations of agent motion within an organisation should be considered *a-priori unpredictable*.

Scenario:

- ***dynamic knowledge acquisition -*** agents should be supported and coordinated by the organisation infrastructure to acquire the knowledge about the resources availability and location.
- ***dynamic exploration constraining*** - agents should not be allowed to freely roam an organisation subnetwork. The exploration should be governed according with authorization policies.

Dynamic Knowledge Acquisition



Decentralised management

Distributed information

Dynamic re-configuration

Exceptions handling

TuCSoN

TOPOLOGICAL ABSTRACTIONS

place - execution environment located in a network node

gateway - a place enhanced with facilities for providing information to agents and control their actions.

domain - a group composed by a supervisor gateway and other places or gateways

COORDINATION MODEL

- an interaction space spread over a collection of Internet nodes
- built upon a multiplicity of independent tuple-based communication abstractions called **tuple centres**
- associates tuple centres to each Internet node
- tuple centres enable agent interactions - **`tc@node?Op(`*`tuple`*`)`**

The TuCSoN Coordination Space

Node --> application-dependent tuple centre

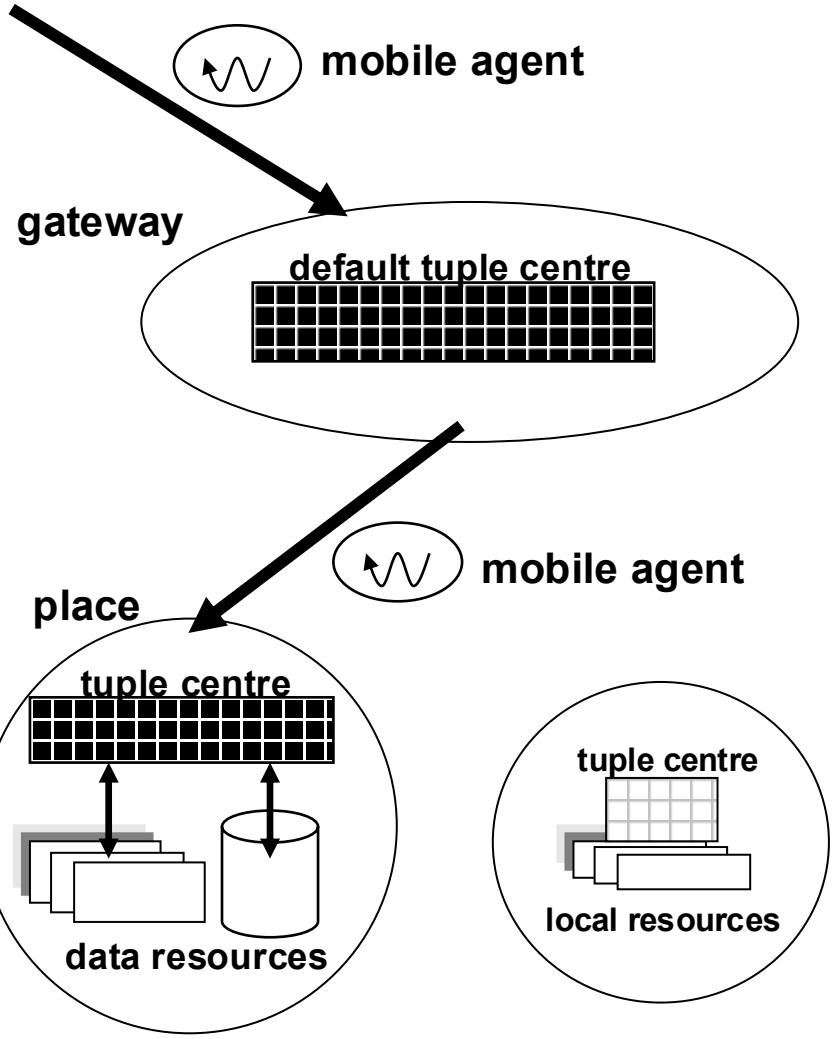
Gateway --> Node + **default** tuple centre

Benefits:

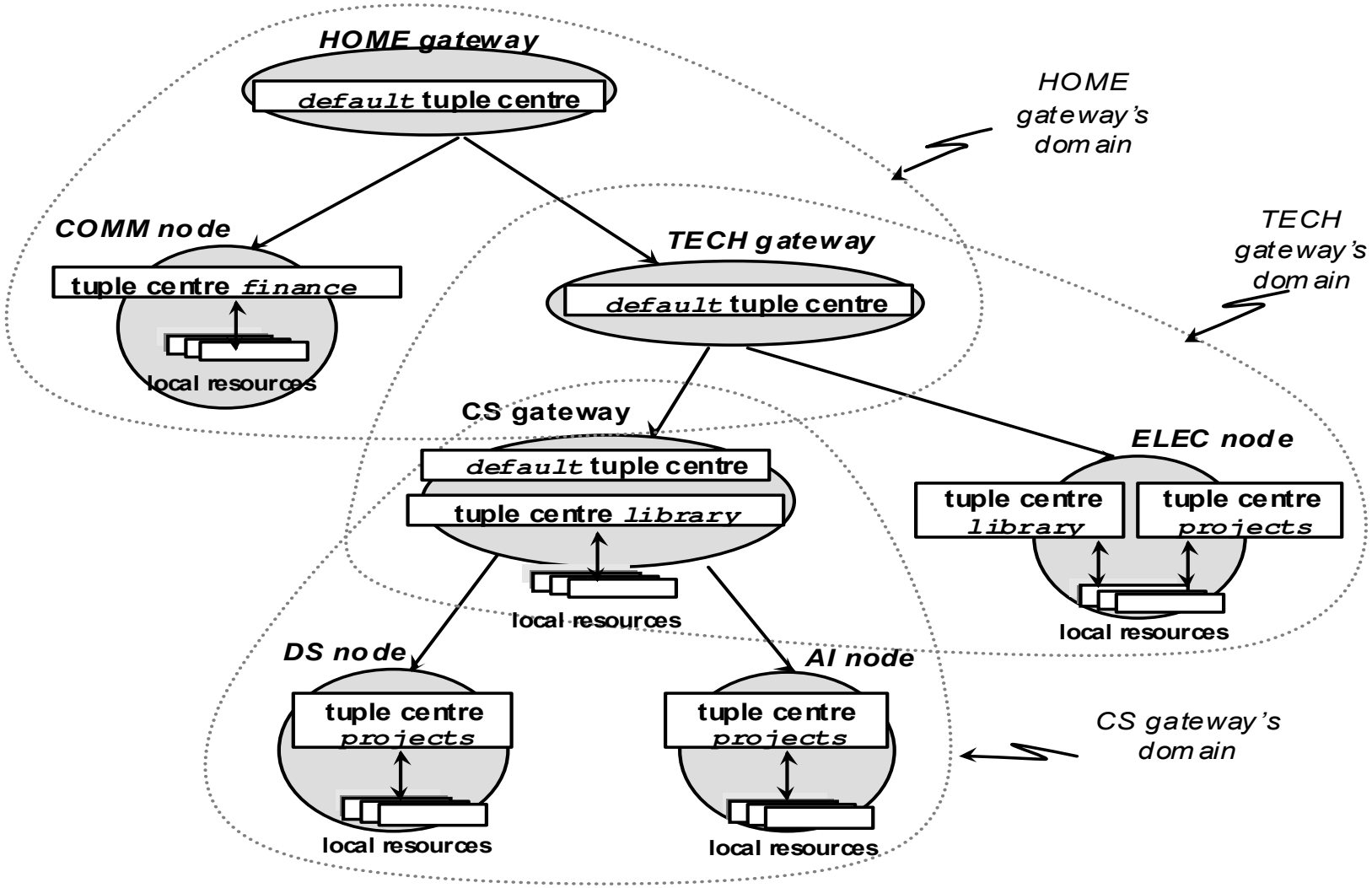
resource heterogeneity transparency

support for exploration

programmability of coordination and security policies



The Hierarchical Topology



Dynamic Exploration Constraining

LEAST PRIVILEGES PRINCIPLE

- a traditional principle of system design: the least set of permissions should be granted to each entity.
- **mobility** and **access to the distributed topology** should be subject of specific permissions
- avoid unauthorised access to nodes (save computational resources, limit exception handling)
- avoid uncontrolled roaming of mobile code

SECURITY ISSUES

- Authentication
- Access Control

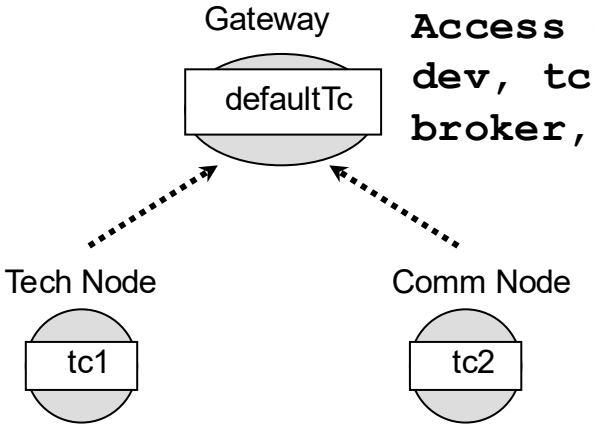
Security Policies

Access Control List

`ACLentry(Identity, Permission)`

Access Control Matrix

`ACMatrix(Identity, TupleCentre, Node, Permission)`



`Access Control Matrix`
`dev, tc1, tech, readerDB`
`broker, tc2, comm, managerDB`

`Access Control List`
`dev, readerDB`

`Access Control List`
`broker, managerDB`

Communication Pattern

defaultTc@gateway?in(*standard domain tuple template*)

where *standard domain tuple template* is:

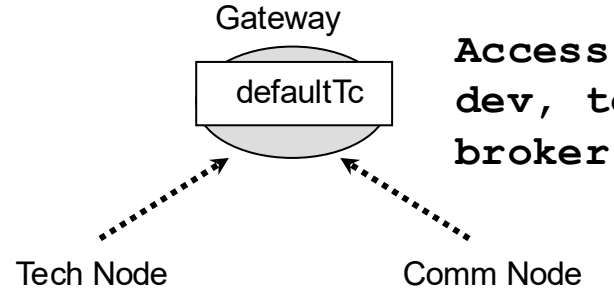
domain(Role, info(Tc, Node, Permission), InfoList))

Example - in operation

defaultTc@gateway?in(domain(broker,info(_, _, _),InfoList))

Results

domain(broker, info(_, _, _), [info(tc2, comm, managerDB)])



Access Matrix
dev, tc1, tech, readerDB
broker, tc2, comm, managerDB

Communication Pattern 2

```
defaultTc@gateway?in(
domain(Identity, info(Tc, Node, Permission), InfoList)))
```

Mode	Example
`info(_, _, _)`	`info(_, _, _)`
`info(`*`tc`*`, _, _)`	`info(library, _, _)`
`info(_,` *`node`*`, _)`	`info(_, deis.unibo.it, _)`
`info(_, _,` *`permission`*`)`	`info(_, _, manager)`

A Possible Interaction Protocol

network topology

1 `<goto d>`

2 `<identify>`

3 `?read(subgwlist)`

`?read(nodelist:tclist)`

4 `<for p in nodelist do>`

`<goto p>`

local interaction

5 `<for n in tclist do>`

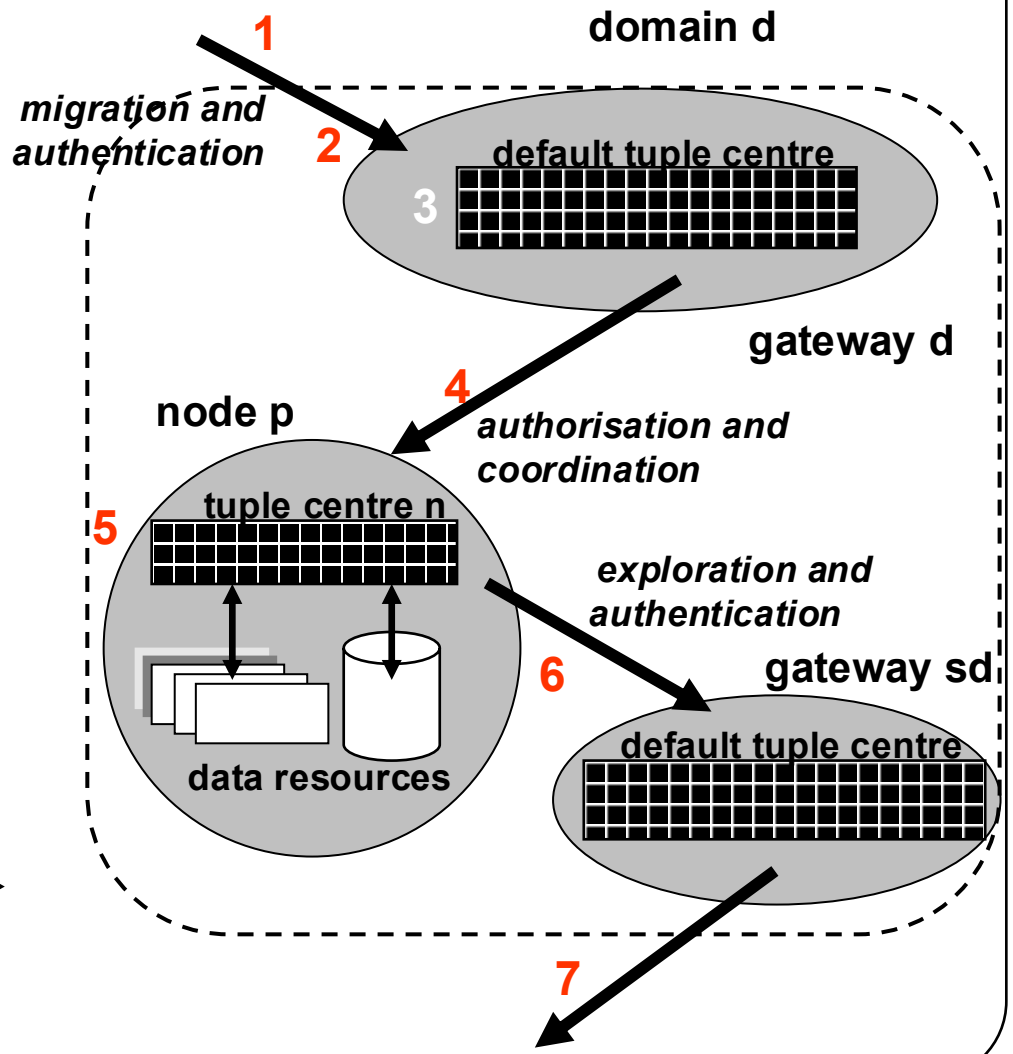
`n?op(Tuple)`

network topology

6 `<for sd in subgwlist do>`

7 `<goto sd>`

`<...>`



Conclusions

Advantages of TuCSoN:

- Agents are unaware of the information resource location and naming
- Agents dynamically acquire information by needs
- Complex environments can be modeled according with their physical and administrative structure;
- Agent motion and interaction are always ruled by a uniform communication space

In-Progress and Future Work:

- Definition of the delegation-based security model
- Integration of coordination and authorization in a Multi-Agent context
- Development of an agent-oriented infrastructure for Virtual Enterprises

TuCSoN *Home Page*

`www-lia.deis.unibo.it/Research/TuCSoN/`