Experience and Prospects for Various Control Strategies for Self-Replicating Multi-Agent Systems

J.-P. Briot, Z. Guessoum, S. Aknine, A. Luna-Almeida, N. Faci and M. Gatti

CReSTIC (Centre de Recherche en STIC, Université de Reims)

LIP6 (Laboratoire d'Informatique de Paris 6)

faci@leri.univ-reims.fr
Fault-Tolerant MAS

- Large-scale multi-agent systems
  - Physically distributed
  - Dynamic environment (with limited resources)

- Types of failures
  - Software (bugs, deadlocks, ...)
  - Hardware (Network links, machines, ...)

» How to avoid failures?
Fault Classifications

Based on how a failed component behaves once it has failed, faults can be classified into 4 categories: crash, omission, timing or Byzantine.

- Crash faults: the component either completely stops operating or never returns to a valid state;
- Omission faults: the component completely fails to perform its service;
- Timing faults: the component does not complete its service on time;
- Byzantine faults: these are faults of an arbitrary nature.
Replication

Existing solution: Replication strategies

- Replication of data and/or computation is an effective way to achieve fault tolerance in distributed systems.
- A replicated software component is defined as a software component that possesses a representation on two or more hosts.

Distributed applications:

- Small number of components
- Component criticality is static
- ...

The number of replicas and the replication strategy are explicitly and statically defined by the designer before run time
Agent Replication

- Multi-agent application characteristics:
  - adaptive agent,
  - large scale,
  - dynamic and adaptive organizational structures
  - ...

- Criticality (the number of replicats and the replication strategy) cannot be explicitly and statically defined by the designer before run time

  - *Automatically* and *dynamically* apply replication mechanisms *where* (to which agents) and *when* it is most needed.
Dynamic Replication

DarX: a new replication framework
- http://www-src.lip6.fr/darx/
- Large-scale distributed systems
- Replication mechanisms
  - Several replication strategies (active, passive, hybrid...)
  - Dynamic replication: change dynamically the number of replicas and the replication strategy
- Observation mechanisms
- Fault detection/recovery mechanisms
- Encapsulation of the system tasks into the replication group
  - Transparence of the replication regarding the other agents
  - Replication mechanisms are not attached to the DarX servers, they are attached to the replication groups
  - ...
Automatic Replication

Adaptive Replication Mechanism

- Which agents need to be replicated and when?
- What is the number of replicas?
- Where?
Adaptive Control of Replication

- **Hypothesis and principles**
  - Automatic mechanisms
  - Some prior inputs from the designer of the application
  - Agents can be either reactive or deliberative
  - Agents can be heterogeneous
  - Agents communicate with some ACL (FIPA, ...)

- **Agent criticality relies on Semantic-level information**
  - Roles [Selmas’03] [AAMAS’02]
  - Interdependence graph [AAMAS’04] [Selmas’05]
  - Plans
  - ...
The arcs are labeled by any information which is susceptible to enable the detection or anticipation of undesirable behaviors.
Interdependence may be defined by considering

- $\text{NbM}_{ij}$: the number of messages received by Agent$_i$ from Agent$_j$

- $\text{NbM}(\Delta t) = \text{Mop}(\text{NbM}_{1,1}(\Delta t), \text{NbM}_{1,2}(\Delta t), ..., \text{NbM}_{n,n}(\Delta t))$

  **Mop** is the aggregation operator median.

- $\Delta t$: monitoring is activated each $\Delta t$

A simple adaptation algorithm

- $w_{ij}(t0)$ initialized by the designer/user

- $w_{ij}(t + \Delta t) = w_{ij}(t) + (\text{NbM}_{ij}(\Delta t) - \text{NbM}(\Delta t)) / \text{NbM}(\Delta t)$
Multi-Agent Architecture

Agents Level

Monitor 1
Agent 1

Monitor 2
Agent 2

Monitor 3
Agent 3

Monitor 4
Agent 4

Observation Level

Host-Monitor
node-2
Adaptation algorithm

Host_i
Host_j
**Multi-Agent Architecture**

**Agent-Monitors**
- observe the domain agents
- build/update the interdependences of the associated agent
- control the domain agents
- ...

**Host-Monitors**
- aggregate information and dispatch back to agent-monitors
- manage the resources
- ...

**Domain Agents (agents of the application)**
Multi-Agent Architecture

Interdependence

Graph

Monitoring agents

Domain Agents

Environment

Observation Level

Micro Level
DimaX: A Fault-Tolerant Multi-Agent Platform

- Various services (naming service, fault detection, replication, ...)
- Agent monitors and host-monitors
- ...

![Diagram of DimaX architecture](image-url)
Experiments

- Technical details
  - Multi-agent platform: DIMA (Guessoum & Briot 99)
  - Middleware: DarX (Guessoum et al. 2003)
    - Naming localization, observation, replication, failure detection

- Example: Scheduling meetings
  - Interact with the user to receive their meeting requests and associated information (a title, a description, possible dates, participants, priority, etc.)
  - Interact with the other agents of the system to schedule meetings.
Experiments

- **Robustness**
  - 100 agents on 10 machines
  - Failure simulator: randomly stops the thread of an agent
  - Scenario
    - 50 meetings
    - Goal of the MAS: Schedule the 50 meetings
  - Rate of successful simulations
    - Number of simulations which did not fail / total number of simulations
  - 3 replication approaches
    - Random
    - Roles
    - Dependences
Experiments

- Robustness

![Graph showing rates of succeeded simulations against number of replicas for different conditions: random, roles, and dependencies.]
Conclusions and Future Work

✓ A new fault-tolerant multi-agent platform (DimaX)
  ↳ Based on DIMA and DarX
  ↳ A new approach to evaluate dynamically the criticality of agents
    • Small applications have been developed (meetings scheduling ...)

❖ Other categories of faults
  ❖ Timing, Byzantine

❖ More experiments
  ❖ To validate the proposed approach
  ❖ To better identify:
    • the potential target application domains (load balancing ...)
    • the domains for which the approach is not suited