Applying Self-Organizing Coordination to Emergent Tuple Organization in Distributed Networks

Matteo Casadei  Mirko Viroli

Alma Mater Studiorum – Università di Bologna
{m.casadei,mirko.viroli}@unibo.it

SASO 2008
Venice, Italy
October 23, 2008
The Need for a New Coordination Paradigm

- Today’s software systems show a high degree of
  - dynamism
  - unpredictability
  - context dependence
  - distribution

- As a consequence, traditional coordination models become unfeasible
  - mechanisms based on *determinism* and *full predictability* are no longer adequate
The Need for a New Coordination Paradigm

- Today’s software systems show a high degree of:
  - distribution
  - dynamism
  - unpredictability
- As a consequence, traditional coordination models become unfeasible:
  - mechanisms based on determinism and full predictability are no longer adequate

Self-organization as an inspiration to promote a brand new coordination paradigm!
Self-Organizing Coordination

- **Self-organization** has so far been adopted by few coordination models
- Self-organization as an inspiring metaphor for conceiving a new model of coordination based on:
  - *local interactions* and *unsupervised control*
- Traditional coordination paradigms deal with obtaining . . .
  - *best efficiency, optimality, predictable control* of system’s interaction
- . . . self-organizing coordination promotes a shifts of the focus to
  - *robustness to failures, adaptivity to unpredictable situations, sub-optimal performances*
Scenario and Goal of the Paper

Objectives

- Tuple-based coordination on networks of distributed tuple spaces
- Exploit self-organization principles to provide a solution to tuple organization
- Tuple organization treated by considering separately the issues of tuple clustering and tuple sorting
- The devised solutions take inspiration from corpse clustering and larval sorting respectively
- Test the solutions on grid-like networks
A Framework for Self-organising Coordination

A Definition

- Self-organizing coordination refers to the management of system interactions by self-organizing properties, namely, where interactions are local, and *global desired effects of coordination appear by emergence*.
- Moreover, self-organizing coordination is based on *coordination media spread over a topological environment* enacting *probabilistic coordination rules*. 
Objectives

- Adopt self-organization in order to provide tuple spaces with rules able to adaptively and emergently specific patterns of tuple distribution.

- Tuple organization and spatial tuple-distribution patterns need to be provided by adopting emergent and adaptive processes.
  - [Emergence]: the area where patterns are generated is uncertain.
  - [Adaptiveness]: patterns should be reached independently of initial conditions and ongoing perturbations.

- In particular, this work focuses on tuple clustering and tuple sorting.
Solution Schema

- A network of tuple spaces connected according to a specific topology.
- Each tuple space is provided with the same set of coordination rules.
- Coordination rules promote interaction among neighboring tuple spaces only so as to move and reposition tuples (according to the pattern to be achieved).
- Coordination rules are probabilistic so as to allow a high degree of adaptiveness to unpredictable state changes.
Tuple Organization III

Relevance

- Distribution of tuples in the right place is essential for easing tuple retrieval (i.e., coordination) by agents
  - Agents cannot look for needed tuples in a complete random way because . . .
  - . . . tuple spaces are distributed so that looking for tuples may imply expensive network operations

- A complete lack of a strategy for searching tuples would result in observing several tuple spaces before finding the required tuple

- It is important to provide agents with some kind of awareness of tuple position

- **Tuple organization** as a way to improve efficiency of coordinated agents
  - Specific spatial tuple-distribution patterns in the tuple-space network forming the coordination system for software agents
Tuple Clustering

Goal

- By taking inspiration from corpse clustering and . . .
- . . . considering one kind of tuples . . .
- . . . find a set of self-organizing rules able to promote emergent and complete tuple aggregation in localized area of the network
Each tuple is tagged a meta-data qualifying a tuple as moving or still:

- a moving tuple is one recently moved
- a still tuple is one not recently moved

All the tuples are initially considered to be moving

As an additional meta-data, each tuple $t$ is tagged with $\text{conc}(t)$

- $\text{conc}(t)$ is a value representing the tuple concentration in the last tuple space visited by $t$

$\text{conc}(t)$ is worth to ease the storage of tuples in tuple spaces already aggregating a great deal of tuples
Tuple Space Protocol

1. A rd operation is performed on local space $L$, yielding a tuple $t$;
2. If $t$ is still
   - $t$ gets moving by a probability $P_{mobile}$
     - If $t$ gets moving: $\text{conc}(t) = \text{conc}(L)$
3. Else ($t$ is moving)
   - A random number $r$ between 0 and 1 is drawn
   - If $r < e^{-\left(\text{conc}(t)/\text{conc}(L)\right)}$
     - $t$ gets still
   - Else
     - A neighboring tuple space $R$ is randomly chosen
     - $t$ is moved to $R$
Simulations

- We performed a series of early simulation on different grid-like topology networks:
  - regular-grid topology
  - torus topology
  - grid topology with diagonal links
- the choice of such a topology was mainly driven by the need of finding a topology providing a uniform setting for testing collective sort
- NetLogo adopted as a simulation framework
Test Instance

- 100-tuple-space torus-topology network:
- 2500 tuples initially distributed in a uniform way
- $P_{mobile} = 0.2$
Simulation: Evolution I
Simulation: Evolution II
Simulation: Final State

Complete Clustering!
Tuple Sorting

Goal

- Generalization of *tuple sorting*
- Main inspiration taken from *larval sorting*
- Tuples belonging to different kinds
- ...find a set of self-organizing rules able to promote emergent *tuple aggregation* and *segregation* based on tuple kind
Tuple Sorting: Ingredients

- Same meta-data for tagging each tuple as for tuple clustering

- \textit{conc} is generalized with regards to tuple kind
  - let $t_m$ be a tuple of kind $m$
  - \textit{conc}(t_m) represents now the concentration of tuples of kind $m$ in the last tuple space visited by $t_m$
Tuple Sorting: Repulsion I

Repulsion

- In addition to the previous mechanisms we need a way to promote segregation of tuples belonging to different kinds.
- If a tuple space contains tuples of different kinds, tuples belonging to kind with low concentration are expected to be likely to be moved away from the space.
- Such a Decision is still made only by relying on local knowledge but traditional Linda primitive (rd) is no longer adequate.

Repulsion Criterion

- How may a tuple space decide to move a tuple \( t \) away?
- It needs to be possible for the tuple space to recognize that the kind of \( t \) is aggregating more elsewhere . . .
- We need a primitive to (probabilistically) perceive kind concentrations.
### Uniform Read Primitive (urd)

- *rd* primitive as proposed in the original **LINDA** model is good enough for our purpose.

- However, all **LINDA** implementations adopt a deterministic *rd* primitive (e.g. based on FIFO policy).

- We need a truly non-deterministic implementation of *rd*, which returns a tuple by adopting equiprobability.
  - we named such a primitive *urd* just to make it clear the contrast with the abstract version proposed in **LINDA**.
Tuple Space Protocol

1. A urd operation is performed on local space \( L \), yielding a tuple \( t_k \) of kind \( k \).

2. Another urd operation is performed on \( L \) yielding a tuple \( t_m \) of a kind \( m \neq k \) (if existing).

3. If \( t_m \) is still
   - Let \( \text{mob}(m, L) = 1 - \frac{\text{conc}_m(L)}{\text{conc}(L)} \)
   - \( P_{\text{mobile}} = \begin{cases} \text{mob}(m, L) & \text{if } \text{mob}(m, L) > 0 \\ C \text{ (constant)} & \text{otherwise} \end{cases} \)
   - \( t_m \) gets moving by probability \( P_{\text{mobile}} \)
     - If \( t_m \) gets moving:
       - \( \text{conc}_m(t_m) = \text{conc}_m(L) \)

4. Else (\( t_m \) is moving)
   - A random number \( r \) between 0 and 1 is drawn
   - If \( r < e^{-\frac{\text{conc}(t_m)}{\text{conc}_m(L)}} \)
     - \( t_m \) gets still
   - Else
     - A neighboring tuple space \( R \) is chosen
     - \( t_m \) is moved to \( R \)
Test Instance

- 100-tuple-space torus-topology network
- 4 different kinds of tuple
- 2500 tuples per kind initially distributed in a uniform way
- $C = 0.2$
Simulation: Evolution I
Simulation: Evolution II
Simulation: Final State

Complete Sorting!
Devised strategies for tuple sorting and clustering inspired from self-organizing principles

Early simulations show that clustering and sorting emerges in torus topology networks

Tests on different topologies and network sizes

First evidence of behavior invariance as the problem scales up
Future Work

- Perform a comprehensive set of experiments to fully validate the presented results
- Adopt stochastic model-checking techniques for a full validation
- Simulations and tests on further topologies
- Propose a concrete case study to apply the proposed strategies
- Start working on a parametric version, that is, able to produce several patterns
Applying Self-Organizing Coordination to Emergent Tuple Organization in Distributed Networks

Matteo Casadei    Mirko Viroli

Alma Mater Studiorum – Università di Bologna
{m.casadei,mirko.viroli}@unibo.it

SASO 2008
Venice, Italy
October 23, 2008