

A Self-Organizing Approach to Tuple Distribution in Large-Scale Tuple-Space Systems

Matteo Casadei¹ **Ronaldo Menezes**² **Robert Tolksdorf**³
Mirko Viroli¹

¹Alma Mater Studiorum – Università di Bologna
{m.casadei,mirko.viroli}@unibo.it

²Florida Institute of Technology - Melbourne, USA
rmenezes@cs.it.edu

³Freie Universität Berlin, Institut für Informatik
tolk@inf.fu-berlin.de

IWSOS 2007

Session IV

Lancaster - the UK

September 12, 2007



Introduction (1)

Scenario

- Coordination of active entities in open distributed scenarios by tuple-space systems
- Specific issue: tuple distribution and organization in a network of distributed tuple spaces

Why is tuple distribution and organization an important issue?

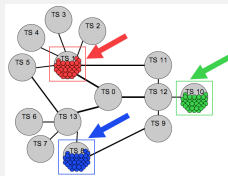
- It affects the scalability of a system ...
- ... indeed, if tuples are organized in a specific way, we can obtain a better system's scalability
- For instance, if similar tuples are kept close to each other it is easier for processes to find the specific tuple they need!



Introduction (2)

Tuple Organization: important issues

- Keeping similar tuples close to each other makes processes too dependent on aggregating nodes
- This leads to **over-clustering**: system's robustness is affected



Objectives

- Exploiting **self-organization** to build a tuple-space architecture with a scalable tuple-distribution mechanism
- Showing that this mechanism is self-organizing

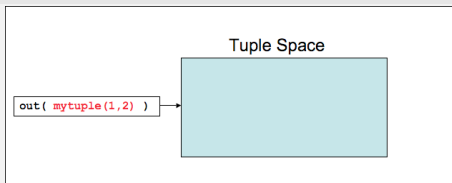


The Tuple-Space Model

Definition

- An associative memory shared by processes that need to coordinate to each other
- Data in a tuple space are called tuples and form a multiset: *a tuple space is a **multiset** of tuples*
- Processes can insert tuples by an `out` primitive and retrieve/read a tuple by an `in`/`rd` primitive

Example: `out (1)`

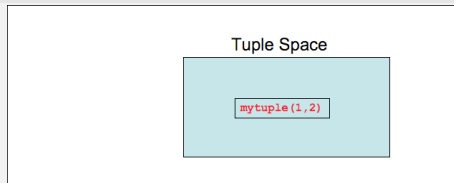


The Tuple-Space Model

Definition

- An associative memory shared by processes that need to coordinate to each other
- Data in a tuple space are called tuples and form a multiset: *a tuple space is a **multiset** of tuples*
- Processes can insert tuples by an `out` primitive and retrieve/read a tuple by an `in`/`rd` primitive

Example: out (2)

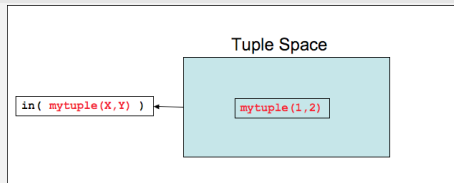


The Tuple-Space Model

Definition

- An associative memory shared by processes that need to coordinate to each other
- Data in a tuple space are called tuples and form a multiset: *a tuple space is a **multiset** of tuples*
- Processes can insert tuples by an out primitive and retrieve/read a tuple by an in/rd primitive

Example: in (1)

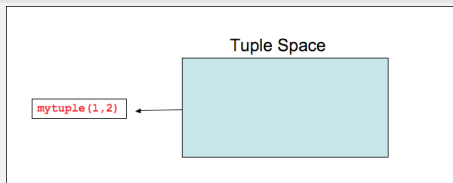


The Tuple-Space Model

Definition

- An associative memory shared by processes that need to coordinate to each other
- Data in a tuple space are called tuples and form a multiset: *a tuple space is a **multiset** of tuples*
- Processes can insert tuples by an `out` primitive and retrieve/read a tuple by an `in`/`rd` primitive

Example: `in (2)`



A Strategy to Organize Tuples

Developed by taking inspiration from ant's brood sorting

Aim of the Strategy

- Storing a tuple in a node with a high concentration of similar tuples
- Metaphor:
 - **outs** are **ants** carrying items and searching for a suitable node where to drop food
 - **tuples** are **items** of different kind carried by ants



How Tuple Distribution Works (0)

Phases

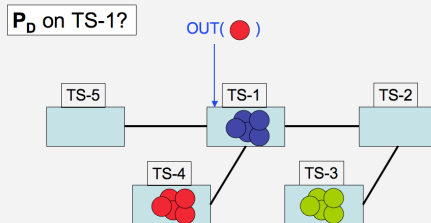
- Our tuple-distribution strategy is composed of **two phases**:
 - **Storing decision phase**: does a tuple have to be stored in the current node?
 - **Movement phase**: what is a good neighbor where to move a tuple?

Similarity Function

- How **similar** are two tuples tu and te ?
 - $\delta(tu, te) = [0, 1]$



How Tuple Distribution Works (1)

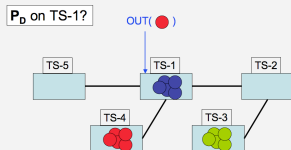


Storing Decision Phase (1)

- **Drop probability** $P_D = \left(\frac{F}{F+K} \right)^2$:
 - Concentration of tuples similar to tu : $F = \sum_{\forall t \in TS} \delta(tu, t)$
 - K is the number of hops left to an *out-ant* before storing the carried tuple tu
 - K is initially set to $Step_{MAX}$



How Tuple Distribution Works (1)



- Drop probability $P_D = \left(\frac{F}{F+K} \right)^2$:
 - Concentration of tuples similar to tu : $F = \sum_{\forall t \in TS} \delta(tu, t)$
 - K is the number of hops left to an *out-ant* before storing the carried tuple tu
 - K is initially set to $Step_{MAX}$

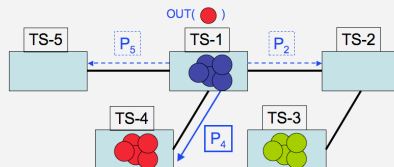
Storing Decision Phase (2)

- K implements an **aging mechanism**
 - the aging mechanism avoids to have *out-ants* wandering forever in a network without being able to store the carried tuple.
- $Step_{MAX}$ is the maximum number of tuple spaces an *out-ant* can visit
 - each time an *out-ant* is not able to store the carried tuple in the current tuple space, K is decreased by 1 and the *out-ant* moves to a neighboring tuple space according to a **movement phase**



How Tuple Distribution Works (2)

Which neighbor to move to ?



Movement Phase

- $\forall$ neighbor j , the **movement probability** P_j is determined:

$$P_j = \frac{F_j}{\sum_{i=1}^n F_i}$$

- F_j : concentration of tuples similar to tu in tuple space j
- $\sum_{i=1}^n F_i$: concentration of tuples similar to tu in the neighborhood of the current tuple space



Simulations

Network Topology

- Tested the strategy on networks featuring a scale-free topology
 - this topology often recurs in real computer networks
- Test instances generated by *Barabási and Albert Scale-Free Model Algorithm*

Test Instances

- Two scale-free test instances composed of
 - 30 and 100 tuple spaces
- Tuples belonging to 4 different templates
- Insertion of 60 tuples per tuple space (15 per tuple template)
- $$\delta(tu, te) = \begin{cases} 1 & \text{template}(tu) = \text{template}(te) \\ 0 & \text{otherwise} \end{cases}$$

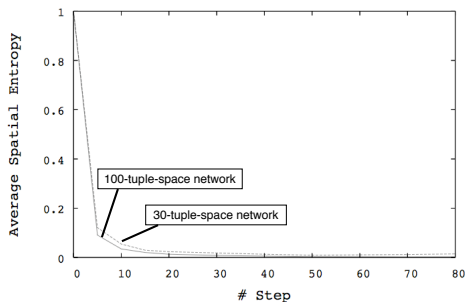


Simulation Methodology

- For each test instance:
 - performed simulations for different values of $Step_{MAX}$ in the range $[0,80]$
- As a measure of the order degree of the information stored in a network we adopted the concept of **spatial entropy** H
- H can be viewed as a measure of **information chaos**
- H features values bounded between 0 and 1:
 - 0 means *complete order* or even better *complete information clustering*
 - 1 means *complete chaos*
- For each value of $Step_{MAX}$:
 - run a series of 20 simulations
 - calculated the value of **average spatial entropy** H_{avg} resulting as average of the single H values obtained by each simulation



Simulation Results



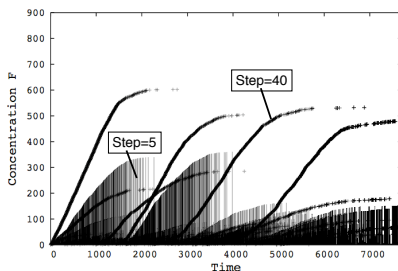
Observations

- Greater values of $Step_{MAX}$ lead to lower values of H_{avg} !
 - *out*-ants can have better chances to explore the entire network
- For $Step_{MAX} > 20$ we have a *horizontal asymptote* = 0.014
 - (quasi-)complete clustering!



Is our Approach Self-Organizing? (1)

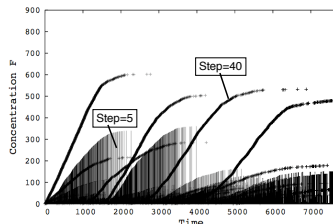
- How to know that our strategy is truly self-organizing?
- A first clue:



- This is a simulation trace showing the trend of concentration F ...
- ... F here refers to the concentration of tuples similar to the current one in the tuple space where the tuple is eventually stored
- The chart shows the trend of F for two different values of $Step_{MAX}$



Is our Approach Self-Organizing? (2)



Meaning

- We know neither the tuple space where a tuple is stored nor the template of that tuple, nonetheless ...
- ... it is easy to see the formation of clusters per tuple template
- Clustering **emerges** from an initially-empty network
- Such behavior is the result of **local interactions** among each tuple space and its neighbors



Proving the Self-Organizing Nature of our Approach (1)

- **Camazine** argumentation (from *Self-Organization in Biological Systems*):
 - Many cases of collective behavior (e.g. wood lice and stable fly) feature organism clustering which arises due to the presence of an **external cue** acting as **environmental template** for the aggregation process
 - In such a situation, if we remove the environmental template, clustering does not occur anymore



Proving the Self-Organizing Nature of our Approach (2)

Camazine's Method to Understand Self-Organization

1. Finding an environmental template suspected to lead the aggregation process
2. Making experiments with initial conditions featuring the presence of such environmental templates: a false self-organizing process shows **insensitivity to different initial conditions**
3. Making experiments with no environmental templates: if clustering occurs, then we have a good indication of the self-organizing nature of our approach

Camazine's Method Applied to our Case

1. Finding an environmental template: **presence of already-formed clusters in the network**
2. Making experiments with initial conditions featuring the presence of such environmental templates: **experiments made by using already-formed clusters of different sizes**
3. Making experiments with no environmental templates; if clustering occurs, then we have a good indication of the self-organizing nature of our approach: **made experiments with initially-empty networks**

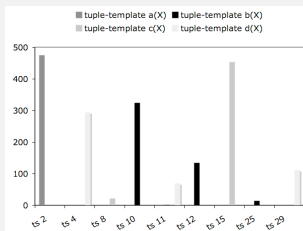


Simulation Results (1)

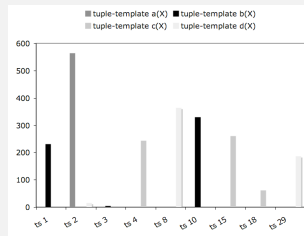
- Simulations executed on a 100-tuple-space network with $Step_{MAX} = 40$ and tuples of 4 different templates

Already-Formed Clusters: One Cluster per Tuple Template

- cluster size: 10



- cluster size: 100



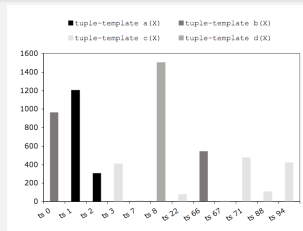
- Clustering occurs and also shows a **dependence** on different cluster sizes (different initial conditions)



Simulation Results (2)

- Simulations executed on a 100-tuple-space network with $Step_{MAX} = 40$ and tuples of 4 different templates

Network Initially Empty



- H corresponding to this simulation: 0.0054

Clustering still occurs even without the presence of clusters!



Simulation Results: Review

- Clustering occurs both with and without the presence of clusters already-formed in the network
- With clusters:
 - clustering evolution is driven not only by interactions among *out*-ants, but also by clusters
 - it is likely to get aggregation in those tuple spaces featuring already-formed clusters
- With no clusters:
 - clustering still occurs only as a result of interactions among *out*-ants



Conclusions and Future Work

Conclusions

- Developed a self-organizing strategy to solve tuple-distribution problem
 - made experiments with larger scale-free networks (> 1000 nodes)
- Provided an argumentation to show the self-organizing nature of the strategy

Future Work

- Making experiments on other network topologies
- Solving the over-clustering problem: for instance by experimenting with dynamic similarity functions
- Devising a tuple-retrieval strategy based on what we have gotten regarding tuple-distribution



Thank you
for
your time!