Self-Organization Algorithms for Autonomic Systems in the SelfLet Approach

D. Devescovi  E. Di Nitto  D.J. Dubois  R. Mirandola

Dipartimento di Elettronica e Informazione

Politecnico di Milano

Reading Group
Outline

1. Self-Organization Algorithms
2. SelfLet Model and Architecture
3. Self-Organization Algorithms as Abilities of SelfLets
4. Conclusions
Motivating Example (1)

- What we have:
  - A generic network of interconnected nodes;
  - Every node is a piece of software that is able to execute a particular task;
  - Nodes that are able to execute the same type of task have the same color.
What we want:

- Reconfigure this network in response to the needs of its nodes;
- Solving Load Balancing problems;
- Creating collaborative groups in order to solve complex tasks.
Constraints:

- The solution must be independent as much as possible from the size of the network;
- Each node has only information about its direct links (neighbors);
- Removal, insertion, or random/selfish behaviors of some nodes should not impact the achievement of our goals.

Possible Solution: Biologically inspired self-organization algorithms.
A self-organization algorithm is defined as an algorithm capable of making a spontaneous formation of well-organized structures, patterns, and behaviors without central control.

- We consider Autonomic Systems as networks of interconnected nodes called Autonomic Elements.

- Characteristics of Autonomic Elements:
  - They have only partial knowledge of the system;
  - Indirect achievement of the high-level goals of the system;
  - Simple operations at element level constitute the building blocks for more complex operations at system level.

- Examples of self-organization algorithms: Autonomic Clustering algorithms to solve the aggregation problem.
Clustering Algorithm Idea

Simplification of the CASCADAS (Saffre et al., 2007) Clustering Algorithms

1. Election of an initiator node: random in Passive mode, requested by a neighbor in the Active mode;
2. The matchmaker chooses two neighbors that share the same type and creates a connection between them;
3. The matchmaker removes a link between itself and one of the chosen neighbors.

Example

Creation of link 3-4 and removal of link 4-5.
Clustering Algorithms have been validated using mathematical models and simulations.

How to put them in practice: the *SelfLet*\(^1\) model:

- **Internal Autonomicity**: changing internal state and behavior in response to events (*IBM approach*);
- **External Autonomicity**: achieving high level goals through interactions between different elements of the system (*AntHill approach*).

\(^1\)SelfLet model is based on the ACE model of CASCADAS.
Definition

A SelfLet is a self-sufficient piece of software which is situated in some kind of logical or physical network, where it can interact and communicate with other SelfLets.

Characteristics of a SelfLet:

- it is characterized by a unique ID, one or more types and a location;
- it may provide services called Abilities that can be executed locally or remotely;
- it may request and offer goals to other SelfLets;
- its execution is managed by one or more Behaviors, represented as FSMs;
- it has Autonomic Rules that can modify the Behavior in response to changes in the internal state or in the environment.
SelfLet Internal Architecture

- Internal Knowledge
- Behaviour Manager
- Ability Execution Environment
- Negotiation Manager
- Autonomic Manager
- Message Handler
  - Messages to/from other SelfLets
  - Policy
  - Abilities
SelfLet Life Cycle

1. **Definition of a Behavior:**
   - Design of a StateChart that defines the actions that the SelfLet will execute;
   - Definition of how it can be modified in order to support application dependent self-configuration.

2. **Definition of Actions and Goals:**
   - Actions involve the execution of application-dependent services called Abilities;
   - Goals can be achievable or needed and they usually refer to the possibility to install and run Abilities.

3. **Definition of an Autonomic Policy:**
   - A high-level specification of the system goals that will be converted into Autonomic Rules that may install/execute Abilities or change Behaviors.

4. **Deployment.**
Self-Organization algorithms can be used to solve this problem: they can be integrated as Autonomic Abilities.

Examples of self-organization Clustering and Reverse-Clustering Algorithm Applications:

- **Load Balancing**: when the load of a SelfLet passes a certain threshold and requires to delegate some tasks;
- **Fault Tolerance**: recreate a group of SelfLets when some of the members of the group are unavailable;
- **Cooperative Grouping**: when a SelfLet is not able to reach autonomously some complex goals.
Initialization and Termination

When self-aggregation ability is invoked on a SelfLet, it will propagate the algorithm initialization on other SelfLets following a domino approach. Termination is triggered by an Autonomic Rule when the SelfLet’s group remains stable.

- **Distribution Issues:**
  - Preventing network saturation: self-adaptive iteration frequency;
  - Synchronization: use of lock primitives;
  - Error and timeout management: rollback of incomplete iterations.
Performance Indexes

- *Exchanged messages* and network *homogeneity*.

**Definition**

\[
\text{Homogeneity} = \frac{\sum_{i=1}^{N} v(i)}{L} \quad \text{(Saffre et al., 2006)}
\]

- \(N\) is the number of nodes, \(v(x)\) is the number of neighbors of \(x\) that has the same type of \(x\), \(L\) is the total number of links of the network.

**Simulations:**

- use of a simulation framework that runs a SelfLet (*Manager SelfLet*) with a centralized initialization and analysis tool as ability;
- the experiments involved the use of Active and Passive (reverse) clustering on a network of 100 nodes and 1000 links and 5 types randomly distributed.
## Analysis Results

Number of Messages vs execution time in clustering algorithms

<table>
<thead>
<tr>
<th>Topology</th>
<th>Normal</th>
<th>Active</th>
<th>Passive</th>
<th>Reverse</th>
<th>Active</th>
<th>Passive</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time</td>
<td>Msg $\times 10^3$</td>
<td>Time</td>
<td>Msg $\times 10^3$</td>
<td>Time</td>
<td>Msg $\times 10^3$</td>
</tr>
<tr>
<td>Random</td>
<td>25s</td>
<td>19</td>
<td>19</td>
<td>1s</td>
<td>3.0</td>
<td>2.9</td>
</tr>
<tr>
<td></td>
<td>100s</td>
<td>64</td>
<td>66</td>
<td>20s</td>
<td>6.5</td>
<td>15.8</td>
</tr>
<tr>
<td>Torus</td>
<td>25s</td>
<td>21</td>
<td>19</td>
<td>1s</td>
<td>2.9</td>
<td>2.8</td>
</tr>
<tr>
<td></td>
<td>100s</td>
<td>72</td>
<td>68</td>
<td>20s</td>
<td>11.8</td>
<td>15.8</td>
</tr>
<tr>
<td>Spiral</td>
<td>25s</td>
<td>20</td>
<td>19</td>
<td>1s</td>
<td>2.9</td>
<td>2.9</td>
</tr>
<tr>
<td></td>
<td>100s</td>
<td>72</td>
<td>67</td>
<td>20s</td>
<td>13.1</td>
<td>15.6</td>
</tr>
</tbody>
</table>
Analysis Results
Normal clustering *Homogeneity* vs algorithm completion time

**Comparison**

The algorithm is successful in increasing the homogeneity with all the different initial topologies we have used.
Analysis Results
Reverse Clustering *Homogeneity* vs algorithm completion time

Comparison

The algorithm shows better performances in all cases, but it is particularly good in the Random topology.
Conclusions

Summary

- We have presented a comprehensive model of autonomic system called SelfLet that addresses some of the limitations of existing autonomic system models.
- We have seen how self-organization algorithms integrate into the SelfLet model in the building of SelfLets neighborhoods.
- Finally we have investigated their effectiveness using a distributed simulator framework based on the SelfLet model.

Future Work

- Clustering algorithms can be made adaptive by considering changing environment and goals.
- Clustering algorithms can be generalized to cluster similar types and to deal with nodes with multiple types.
- Finding a better solution to the initialization/iteration/termination problem in order to reduce the number of exchanged messages.