10

A Dependable Multi-Agent System with Self-Diagnosable Function

Keinosuke Matsumoto, Akifumi Tanimoto and Naoki Mori
Osaka Prefecture University
Japan

1. Introduction

In tandem with the penetration of the Internet, many information systems are connected to large-scale computer network systems or multi-processor systems. These distributed systems cooperate, negotiate, and solve problems by exchanging information each other. A multi-agent system (MAS) (Weiss, 1999) attracts attention as a solution to competition or cooperation in distributed systems. It is necessary that multi-agent systems operate for a long time without human's support from the viewpoint of dependable distributed system. Some techniques (Chiang & Chen, 1994) that assume failures beforehand have been developed for this problem. These techniques need another integrated system that managements all the agents for fault diagnosis. It is practically impossible to apply them to large-scale distributed systems.

We need a technique detecting faults autonomously in multi-agent systems. A self-diagnosable algorithm (Kohda et al., 1997) has been proposed for distributed systems. In this conventional simple highly structured system, all agents mutually diagnose all other agents. It has a problem that the more agents are included in a system, the more communication loads increase. To solve the problem, this paper proposes a new self-diagnosable multi-agent system that mitigates communication loads in the multi-agent system. Our method introduces middle agents that do not interaction with basic client agents to mitigate the communication traffics between agents.

2. Conventional self-diagnosable algorithm

This section describes a conventional self-diagnosable algorithm, which we call it as a simple highly structured system.

2.1 Faulty agent

An agent may malfunction in a multi-agent system and such an agent is called a faulty agent. Our research deals with the following three kinds of failure (Fedoruk & Deters, 2001):

a. Crash failure
b. Byzantine failure
c. Communication failure.

Where, crash failure refers to an agent that is stopped by a processor fault or shortage of system resources. Byzantine failure is a fault that does not assume faulty agents' behavior,

and it is caused mainly by unexpected state transitions or program bugs. Communication failure drops out all or some part of messages, or it cannot establish communication link between agents.

If these failures occurred simultaneously at two or more agents, a multi-agent system will malfunction because of interaction among agents. When faulty agents exist in a system, reliability of the system cannot be maintained without identifying the faulty agents and processing them properly.

2.2 T-fault one-step diagnosable system

Mutual diagnosis in a self-diagnosable algorithm (Preparata et al., 1967) is expressed by a directed graph as shown in Fig. 1. In this figure, a unit and testing correspond to a vertex and directed arc respectively. The system is defined as a directed graph G = [V, E], where V = {v} is a set of vertexes (units) and E = {(u, v)} is a set of directed arcs (u, v). The unit corresponding to starting point u and terminal point v of directed arc (u, v) are called a testing unit and tested unit respectively.

A test result a(u, v) is defined as shown in Table 1, and let it be a weight of the directed arc. This algorithm premises that it does not trust the test result if the testing unit is out of order and each unit does not test itself. As another self-diagnosable algorithm, t-fault one-step diagnosable system (Hakimi & Amin, 1974; Kohda, 1994) is proposed that can detect simultaneously multiple faults of up to t among n units from mutual diagnostic results of the system (hereinafter referred to as t-OD system).

2.3 Highly structured T-OD system

Definitions of a highly structured system and its characters as a t-OD system are described below:

[Definition 1] A system G = [V, E] is a highly structured system if an arbitrary unit v ($v \in V$) of the system G has a subsystem H(v; α, β) expressed in Fig. 2.

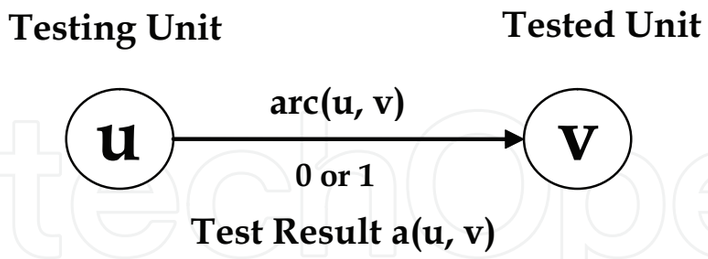


Fig. 1. Basic system component

Testing Unit **u**	Tested Unit **v**	
	Fault-free	***Faulty***
Fault-free	0	1
Faulty	*	*

*: don't care

Table 1. Test result a (u, v)

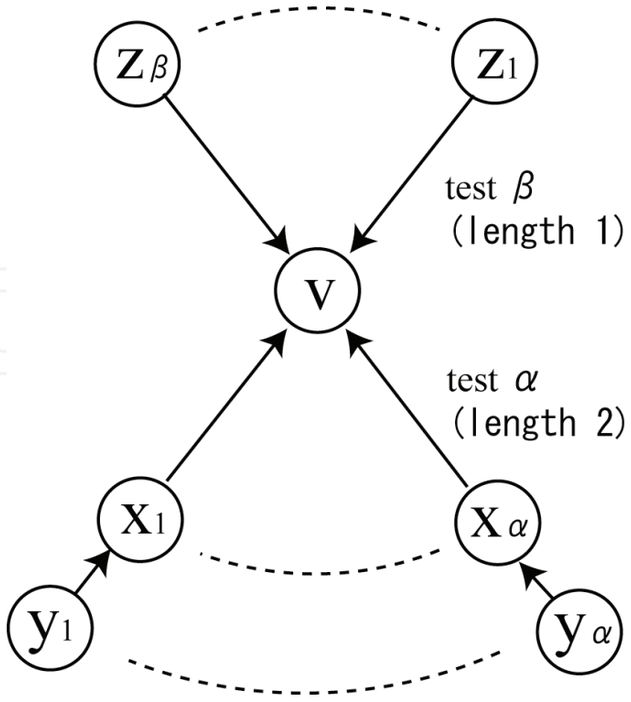


Fig. 2. A subsystem H(v; α, β)

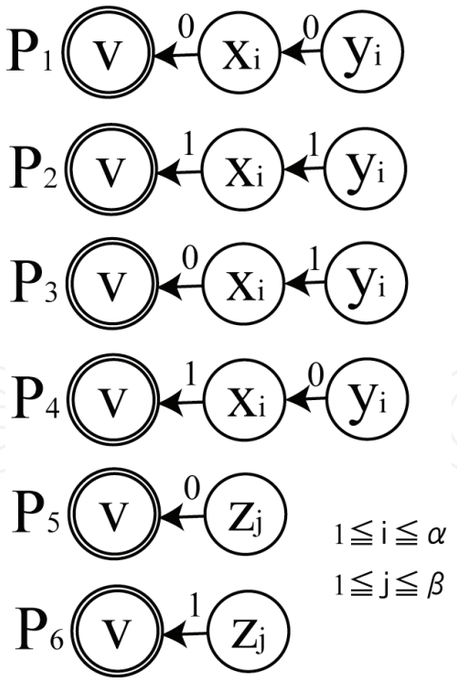


Fig. 3. Six test result patterns of type P_k in H(v; α, β)

The subsystem H(v; α, β) consists of a kernel unit v, (2α+β) other units, and (2α+β) testing links, where α is the number of testing links of length 2 and β the number of testing links of length 1. A highly structured system has the following character 1.

[Character 1] If every unit v ($v \in V$) of a system G = [V, E] has a subsystem H(v; α, β) of the definition 1 and

$$\alpha + [\beta/2] \geq t\ , \tag{1}$$

then G is a t-OD system.
Where, t > 0 and [x] means a maximum integer not exceeding x.
[Definition 2] A highly structured system satisfying (1) is called a highly structured t-OD system.
The following character 2 is suggested as an analyzing method of highly structured t-OD systems.
[Character 2] For an arbitrary unit v ($v \in V$) of a system G = [V, E] that satisfies the character 1, v is normal if and only if the following (2) is satisfied.

$$H(v; \alpha, \beta) \equiv p_1 + [\beta/2] - (p_4 + p_6) \geq 0 \tag{2}$$

Where, p_k is the number of test result patterns of type P_k shown in Fig. 3.

Mutual test results do not depend on the order of diagnosing each unit, and its character is a strong point of the highly structured t-OD system.

2.4 Problems of the self-diagnosable algorithm

The foregoing self-diagnosable algorithms have the following two problems: First, the more agents are included in a system, the more communication traffics increase. This is because the highly structured system (hereafter, it will be called a simple highly structured system in order to distinguish from our method) applies a simple mutual diagnosis for all units. Second, generation of mutual test results must synchronize for all units. If not all the mutual test results are completed, it does not satisfy the condition of t-OD systems. It means that faulty agents cannot be uniquely identified.

3. Self-diagnosable multi-agent system in consideration of load distribution

This section describes an approach to solving the problems that simple highly structured systems have.

3.1 System configuration

The problems can be solved by introducing not only distributed type of elements but also concentrate type of elements. Specifically, middle agents are introduced as elements. All agents of pure MAS are not controlled by any concentrated agents. But it is natural that a hybrid MAS (Hagras et al., 2003; Alur et al., 2003) exists as a middle class between a pure MAS and a pure concentrated system. There is a reason for existence of the hybrid MAS if some advantages could be obtained when it replaces the pure MAS. Middle agents are located in a different layer from a client agent's layer. Each middle agent has the following characters:
[Character 3]

- It is distinguished from basic client agents.
- It does not take part in interaction among basic client agents.
- It carries only information of mutual test results.

- It is assumed not to break down.

Our approach introduces middle agents that do not interaction with basic client agents to mitigate communications between agents. An initial system configuration is shown in Fig. 4. One middle agent is prepared for three client agents (they may become four client agents since a fraction is taken into consideration) that are the minimum composition of a t-OD system in order to keep domain of mutual tests to a minimum. They are grouped as a partial domain. The client agents carry out mutual tests within the group through the middle agent. In this group, only one faulty agent can be detected using (1).

We define middle agents that directly carry client agents' communication as a bottom layer, and another middle agent that carries communication between middle agents is defined as a top layer. This composition technique can limit range of mutual tests and mitigate each agent's load because communications between agents are carried within every local group in parallel.

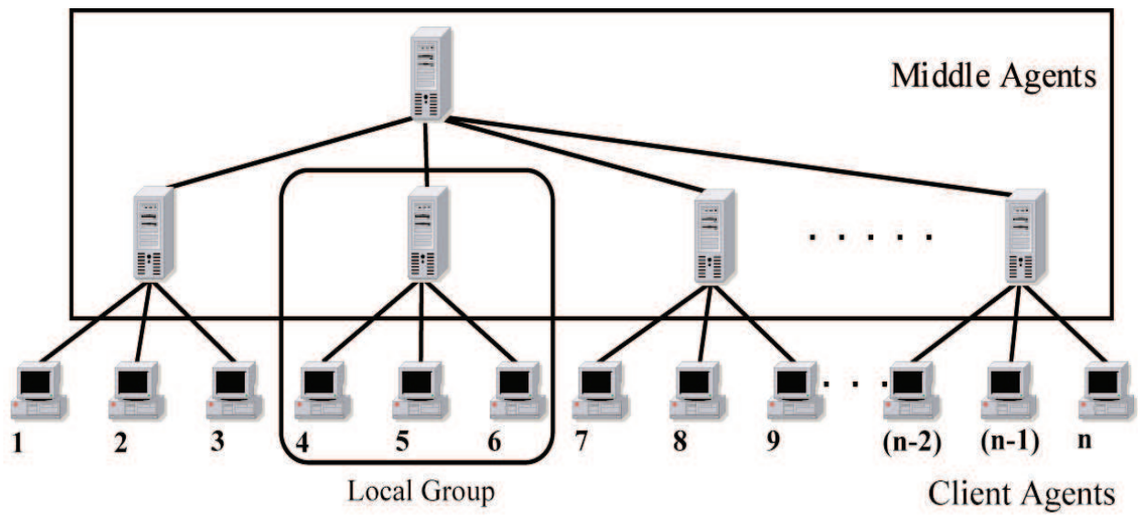


Fig. 4. Initial system configuration of the proposed system

3.2 Dynamic highly structured system

The proposed system can detect only one faulty agent for each group using (1). It is necessary to dynamically reconstitute a system configuration so that faulty agents may not be concentrated in certain groups. For that purpose, the number of branches of the middle agents at the bottom layer is fixed, and candidates of faulty agents are determined by rearranging client agents and changing groups.

Our dynamic highly structured system has the following four processes as shown in the Fig. 5:

1) *System reconfiguration*

At first, we fix a number of client agents that can be connected to a middle agent at the bottom layer. The middle agent exchanges the client agents of each group at random not to concentrate faulty agents in certain groups and to reconstruct mutual testing graphs. Because mutual tests can be conducted in parallel, faulty agents are detected for every group from the test results. These agents are regarded as candidates of faulty agents, and it is not decided to be faulty agents at this time.

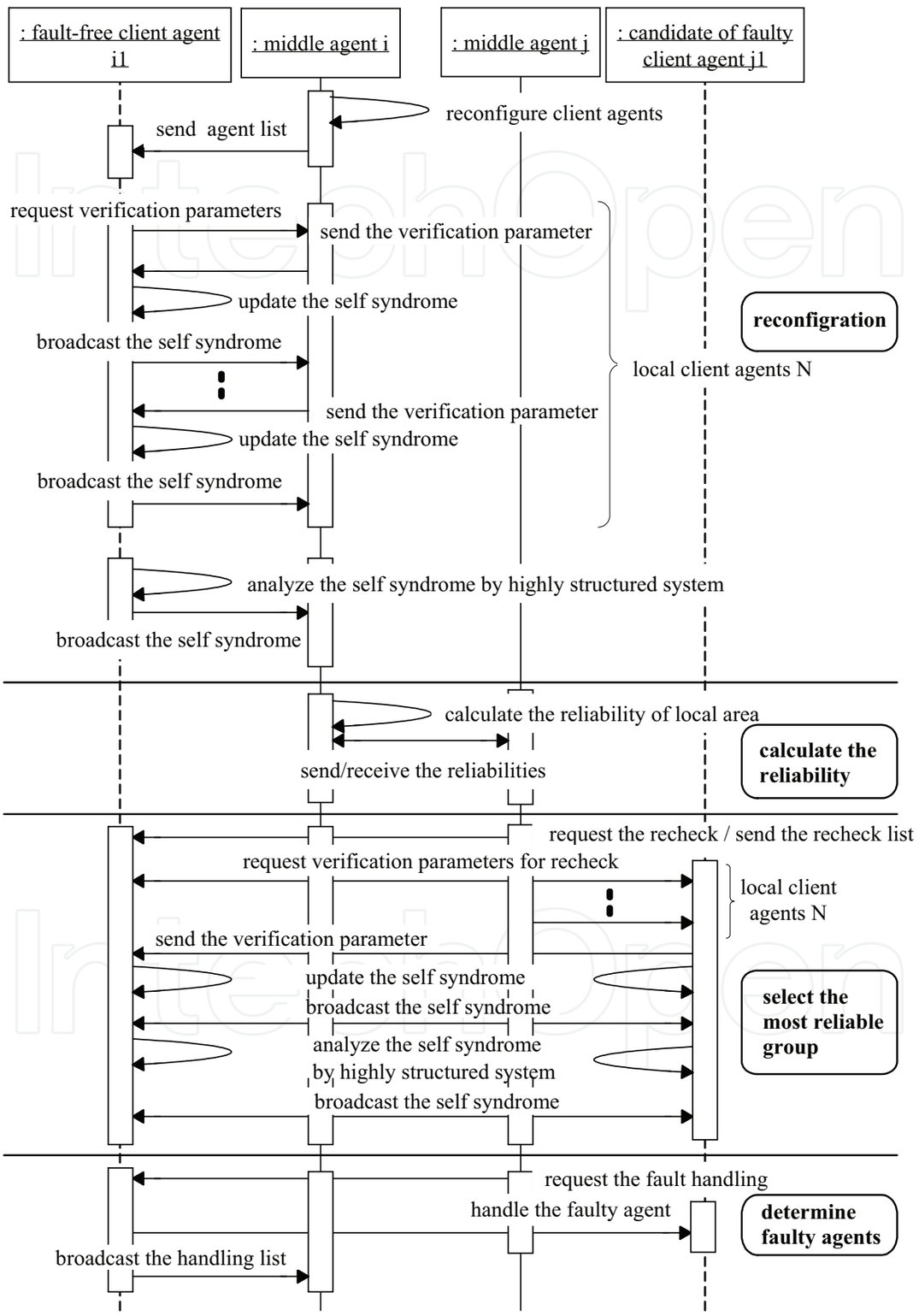


Fig. 5. A process of dynamic highly structured system

2) *Calculation of reliability*
Equation (3) is introduced here. It expresses reliability of a middle agent based on the number of doubtful agents within its group.

$$R(m_i) = 1 - \frac{Nm_i}{Am_i} \tag{3}$$

Where, m_i is an ith middle agent group, Nm_i is a number of faulty agent candidates that belong to m_i, and Am_i is a number of agents that belong to m_i. The value of this function $R(m_i)$ will go down if the number of faulty agent candidates increases. Each middle agent at the bottom layer computes (3) at each group and it shares the information among middle agents.
3) *Selection of the most reliable group*
We omit a group that has more than two faulty agent candidates, which means that it is not a t-OD system. We choose a group of the highest reliability value among the remaining groups. If there are more than two groups have the same value of $R(m_i)$, we give priority to a group whose value of H(v; α, β) in (2) is the largest. H(v; α, β) is a parameter showing allowable degree of failures.
4) *Determination of faulty agents*
Combining faulty agent candidates with agents connected to another middle agents having high reliability, mutual tests are performed again, and faulty agents are finally determined.
Because the number of times of mutual tests changes with the number of failures in an actual system, the dynamic highly structured system has feature that it is able to mitigate the communication loads between agents when all agents are trouble-free or a few agents are faulty. Unlike the conventional techniques, our approach has an advantage that it should synchronize generation of test results only within each local group.
Our self-diagnosable system conducts independent mutual tests only in a middle agent layer. This algorithm has restrictions that more than a half must be normal among the agents that constitute a mutual testing group. It can be said that the element of a meta-level does not need to guarantee high reliability because it is tested to be normal or faulty. Calculation amounts necessary for mutual tests are estimated for a simple highly structured system and a dynamic highly structured system. If the number of client agents is n, $O(n)$: orders of calculation to diagnose are as follows:

- Simple highly structured system

$$O(n) = n^2 - n \tag{4}$$

- Dynamic highly structured system

$$2n \le O(n) \le 8n - 12 \qquad (t = 0) \qquad \left(t = \left[\frac{n-1}{2}\right]\right) \tag{5}$$

where, t is the number of faulty agents.
For the dynamic highly structured system, (5) shows that calculation orders in the cases in which the trouble is not completely occurring, and the number of faulty agents is fewer

than half of all agents by one that is an upper limit of the number of faulty agents. The calculation order changes between these two values according to the number of faulty agents which exists in the system. The dynamic highly structured system needs synchronizing mutual tests only in each local group, but no necessity of synchronizing in all client agents. Therefore, the proposed system can reduce the communication traffics between agents.

4. Simulation results

Our approach has been applied to a multi-agent system (circle multi-agent system) that forms a circle autonomously, and compared with a simple highly structured system.

4.1 Circle multi-agent system

Initial conditions of the circle multi-agent system are shown below:

- Each agent knows a value of diameter that should be kept, and perceives other agents' locations.
- Each agent determines its action based on two distances to the furthest agent and to the nearest one.
- Each agent is placed at a two dimensional latticed space, and chooses one action among eight directions and no motion.

Behaviors of this system are shown in Fig. 6 to Fig. 8. Fig. 6 shows a normal state in which a circle can be formed in cooperation. Fig. 7 shows a faulty state that troubles arise at Agent1, Agent11 and Agent21, and they cannot obtain right location data of other normal agents. As a result, the cooperative target of forming a circle cannot be attained. On the other hand, in Fig. 8 a restoration system stops the faulty agents and the other normal agents form another circle. The experiment environment is as follows: CPU is Pentium4 (R) 2.4 GHz, OS Windows XP Professional, and the programming language Java JDK version 1.4.

4.2 Experiments on communication traffics between agents

The proposed system was incorporated to the circle multi-agent system, and some simulations were carried out. The experiment conditions are shown below:

- Byzantine failures happen at some agents and they cannot get right location data of other agents. It means that they fail to identify the environment.
- The experiments are carried out for two cases:
  1) 10% of agents are faulty, and
  2) 40% of agents are faulty, which means a fatal failure to the system.
- Faulty agents are selected by uniform random numbers.
- An agent is tested comparing location data of its own with the same location data that another agent has.
- A restoration system only stops faulty agents.

For a simple highly structured system and a dynamic highly structured system, the experiment measured the number of communication traffics between agents to detect faulty agents. For both systems, the number of agents was changed every ten agents from 10 to 100. Fig. 9 shows average values of ten trials.

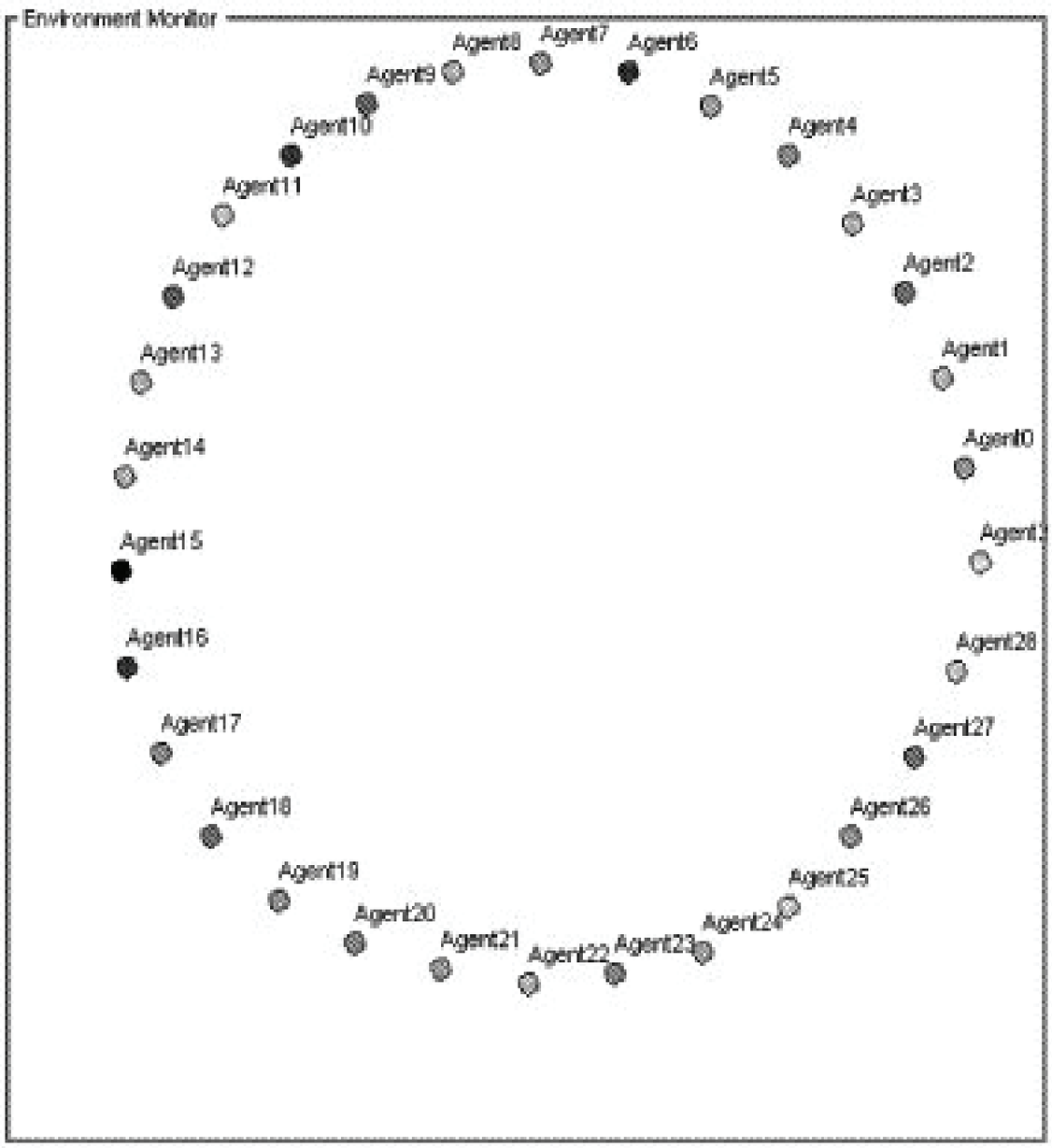


Fig. 6. A normal state of a circle multi-agent system

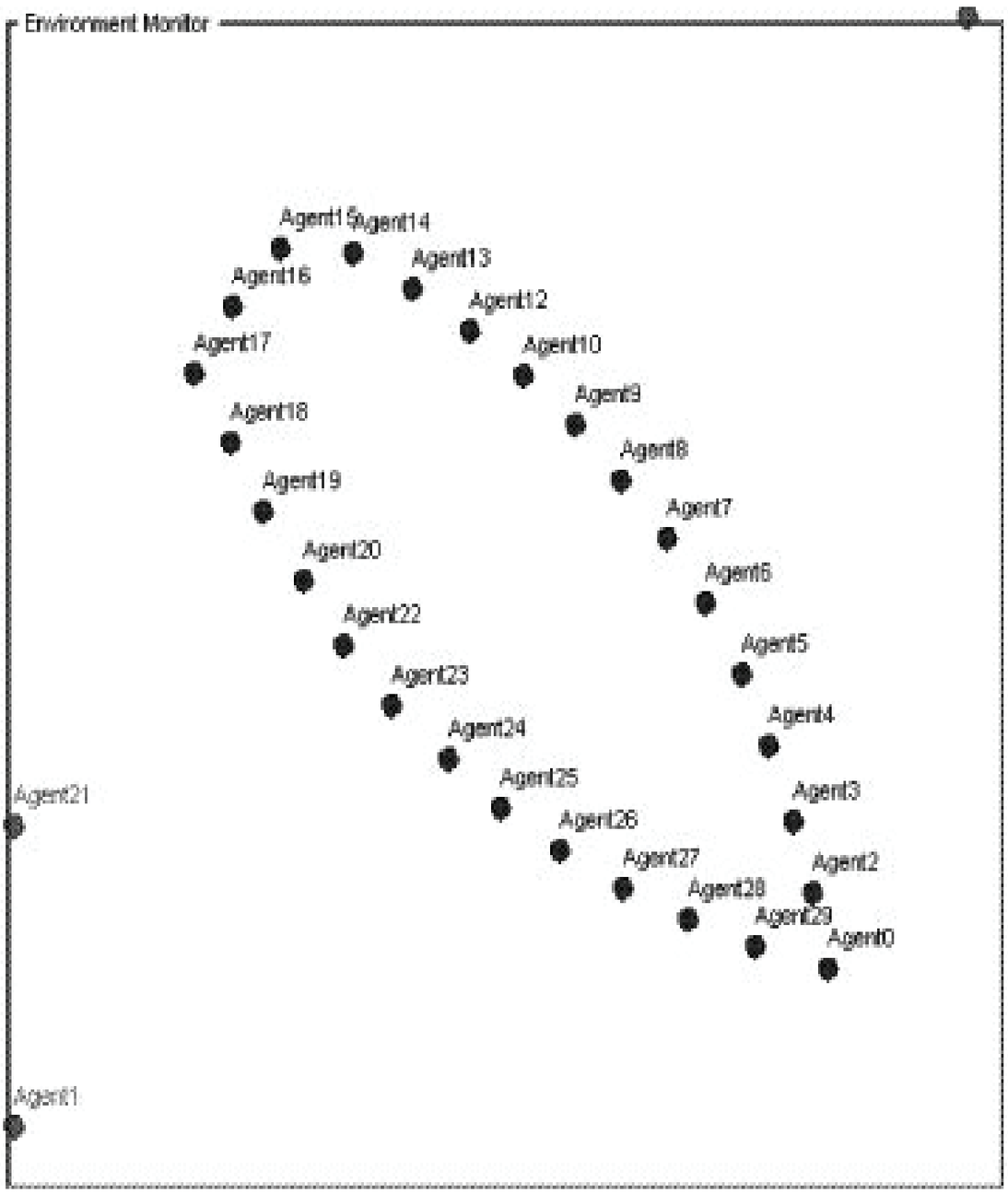


Fig. 7. A state after Byzantine failure

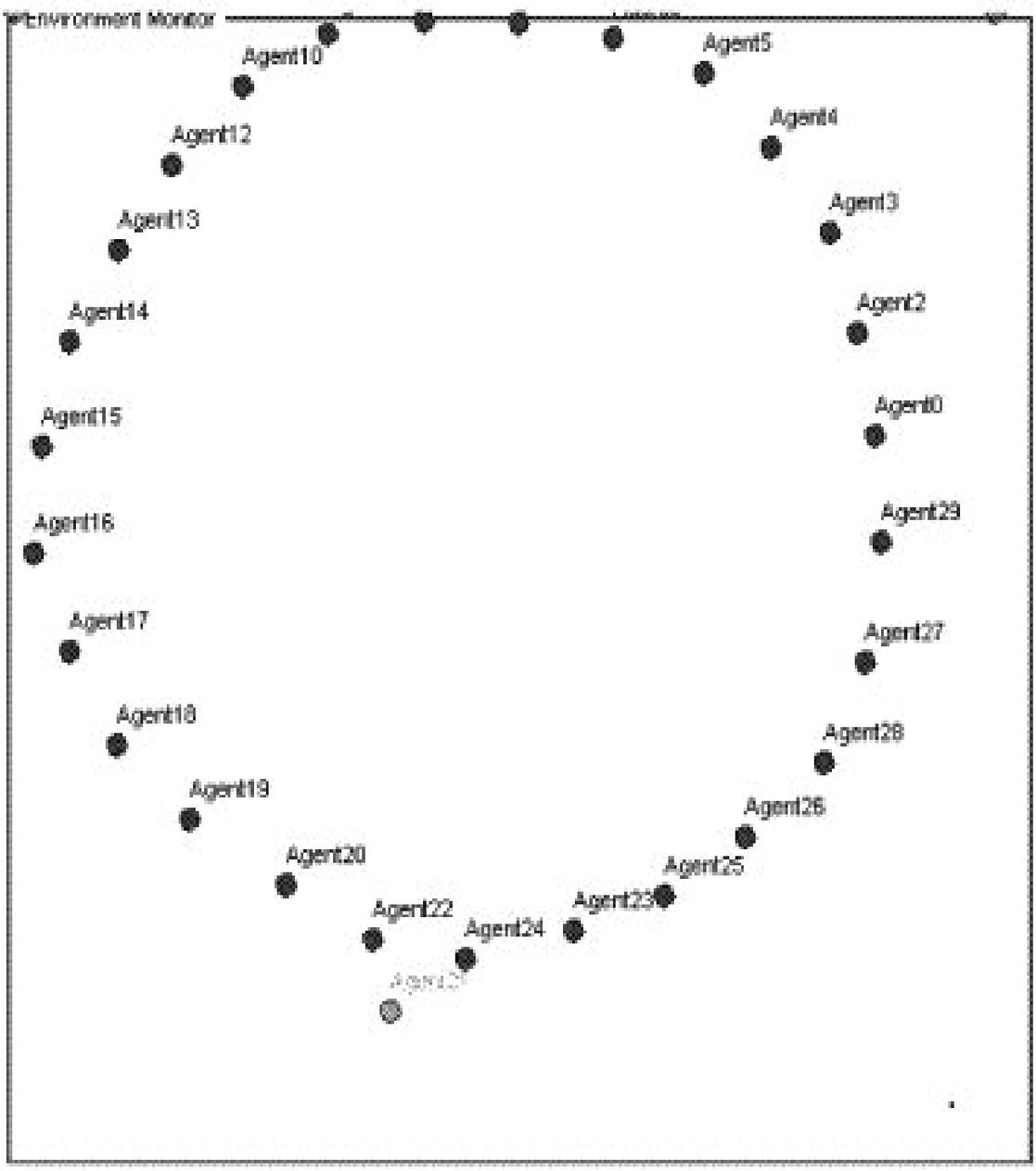


Fig. 8. A state after restoration

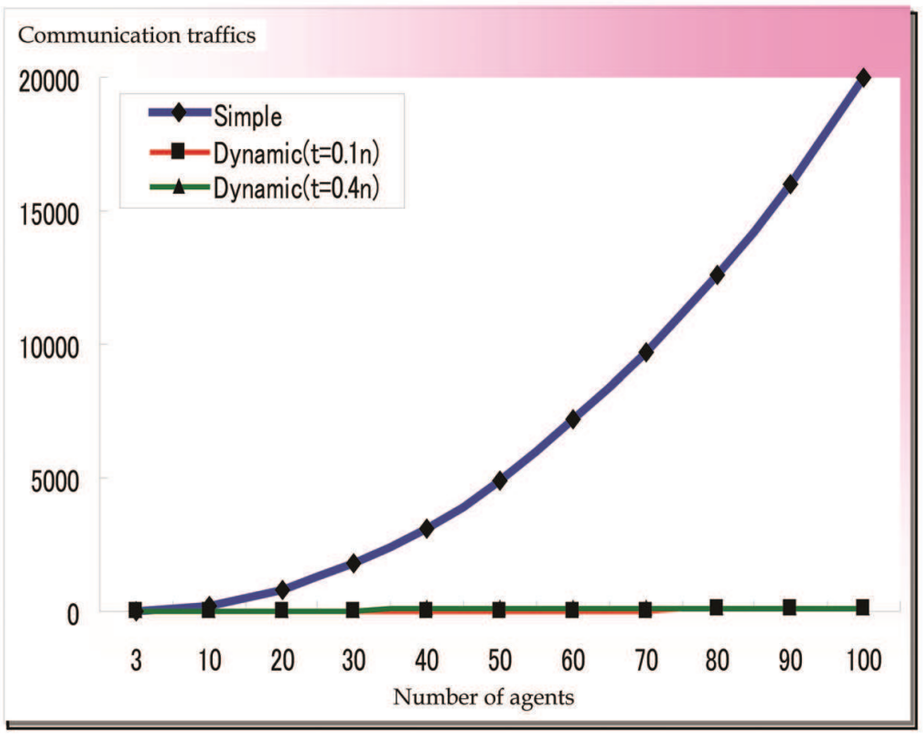


Fig. 9. Communication traffics

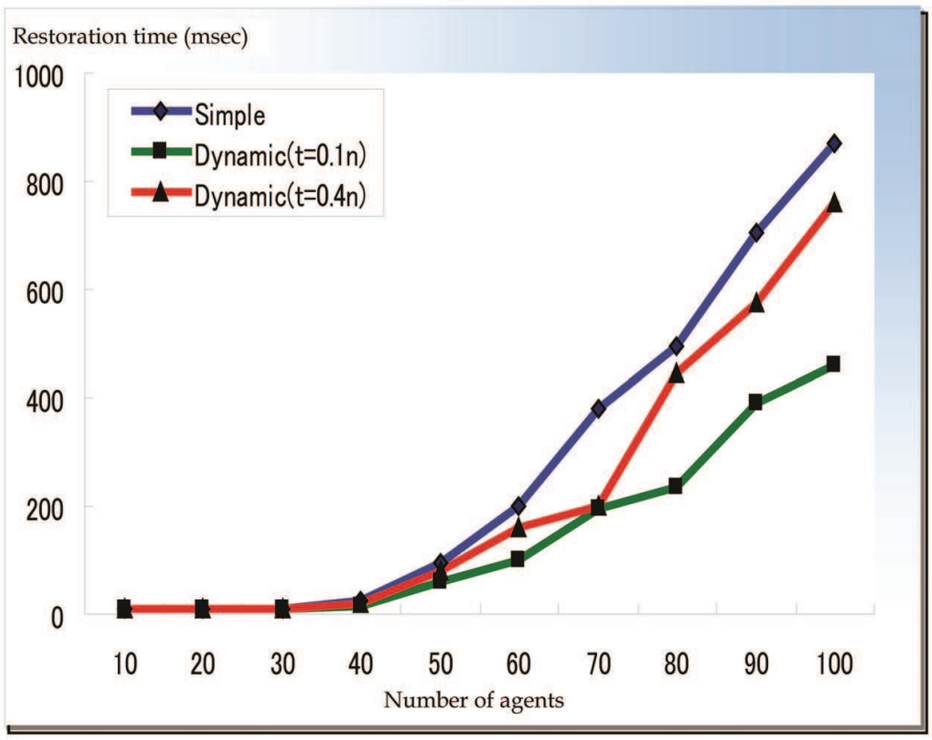


Fig. 10. Restoration time

Fig. 9 proves that the proposed system needs only a few times to communicate while the simple highly structured system remarkably increasing times with the number of agents. It means that mutual tests are carried out in parallel in dynamic highly structured system. Small communication frequencies make the load of the agents reduce.

As for failure processing, we measured the restoration time from occurring of failure to stopping faulty agents. Fig. 10 shows the results. The more faulty agents exist, the longer the restoration time is. The restoration time of the conventional system has exceeded that of the proposed system for both cases of failure rate 10% ($t = 0.1n$) and 40% ($t = 0.4n$). The cause is that the proposed system can be processed in parallel and synchronizing only in each local group.

5. Conclusion

This paper has proposed a self-diagnosable multi-agent system that uses a dynamic highly structured system for fault diagnosis and a restoration system. Some experiments have demonstrated the validity of the proposed system. Future works to be done are as follow: development of a more accurate reconstituting algorithm of basic client agents, a more advanced restoration algorithm, and a reliability improvement method of middle agents.

6. Acknowledgment

This work was partially supported by JSPS KAKENHI 21560430.

7. References

Alur, R.; Thao Dang; Esposito, J.; Yerang Hur; Ivancic, F.; Kumar, V.; Mishra, P.; Pappas, G.J., & Sokolsky, O. (2003). Hierarchical Modeling and Analysis of Embedded Systems, *Proceedings of the IEEE*, Vol.91, No. 1, pp. 11- 28.

Chiang, W. K. & Chen, R. J. (1994). Distributed Fault Tolerant Routing in Kautz Networks, *Journal of Parallel and Distributed Computing*, Vol. 20, No. 1, pp. 99-106.

Fedoruk, A. & Deters, R. (2001). Improving Fault Tolerance by Replicating Agents, *Proceeding of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pp. 737-744, Bologna, Italy.

Hagras, H.; Callaghan, V.; Colley, M. & Clarke, G. (2003). A Hierarchical Fuzzy–Genetic Multi-Agent Architecture for Intelligent Buildings Online Learning, *Adaptation and Control, Information Sciences,* Vol.150, No. 1-2, pp. 33-57.

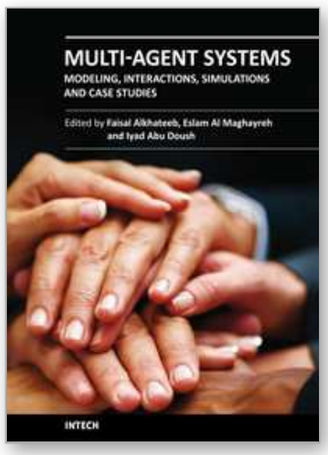
Hakimi, S. L. & Amin, A. T. (1974). Characterization of Connection Assignment of Diagnosable Systems, *IEEE Transaction on Computers*, Vol.23, No.1, pp. 86-88.

Kohda, T. (1994). A Simple Discriminator for Identifying Faults in Highly Structured Diagnosable Systems, *Journal of Circuits, Systems, and Computers*, Vol. 4, No. 3, pp. 255-277.

Kohda, T.; Yoshida, K.; Sujaku,Y., et al. (1997). Decentralized Self-Diagnosis in Multi-Agent Systems, *Proceeding of the 6th Multi-Agent and Cooperative Computing Workshop*, Kobe, Japan.

Preparata, P.; Metze, G. & Chien, R. T. (1967). On the Connection Assignment Problem of Diagnosable Systems, *IEEE Transaction on Electronic Computers*, Vol. 16, No. 6, pp. 848-854.

Weiss, G. (1999). *Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence*, The MIT Press.



Multi-Agent Systems - Modeling, Interactions, Simulations and Case Studies

Edited by Dr. Faisal Alkhateeb

ISBN 978-953-307-176-3
Hard cover, 502 pages
Publisher InTech
Published online 01, April, 2011
Published in print edition April, 2011

A multi-agent system (MAS) is a system composed of multiple interacting intelligent agents. Multi-agent systems can be used to solve problems which are difficult or impossible for an individual agent or monolithic system to solve. Agent systems are open and extensible systems that allow for the deployment of autonomous and proactive software components. Multi-agent systems have been brought up and used in several application domains.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Keinosuke Matsumoto, Akifumi Tanimoto and Naoki Mori (2011). A Dependable Multi-Agent System with Self-Diagnosable Function, Multi-Agent Systems - Modeling, Interactions, Simulations and Case Studies, Dr. Faisal Alkhateeb (Ed.), ISBN: 978-953-307-176-3, InTech, Available from: http://www.intechopen.com/books/multi-agent-systems-modeling-interactions-simulations-and-case-studies/a-dependable-multi-agent-system-with-self-diagnosable-function

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen