Available online at www.sciencedirect.com
SCIENCE DIRECT®
Microprocessors and Microsystems 29 (2005) 9–20
MICROPROCESSORS AND MICROSYSTEMS
www.elsevier.com/locate/micpro

An adaptive adjusting mechanism for agent distributed blackboard architecture

Y.C. Jiang[a,*], Z.Y. Xia[b], Y.P. Zhong[a], S.Y. Zhang[a]

[a]*Department of Computing & Information Technology, Fudan University, Shanghai 200433, China*
[b]*Department of Computer, Nanjing University of Aeronautics and Astronautics, Nanjing 210043, China*

Received 23 November 2003; revised 11 May 2004; accepted 5 June 2004
Available online 1 July 2004

Abstract

Distributed blackboard is one of the popular agent communication architectures. However, in current agent systems, the distributed blackboard architecture is kept fixed after its initial setting, which may influence the system performance when network topology or agent cooperation relations are changed during operation. To solve the problem, this paper presents a novel mechanism for adjusting agent communication architecture. Based on graph theory, this mechanism provides a way to adjust the distributed blackboard architecture. The adjustment made to the architecture kept its validity, and the adjusted architecture outperforms the initial one in new network topology or agents cooperation relations, which are proved by the Mobile Ambients Calculus analysis and the simulation experiments. Therefore, the adjusting mechanism presented here can achieve the adaptation of the agent communication architecture to the changes of the network topology and agent cooperation relations.

Keywords: Multi agents; Agents communication; Distributed blackboard; Network topology; Agent cooperation

1. Introduction

In multi-agent systems, cooperation will enable agents to solve the problems that cannot be solved by individual one. To implement cooperation among agents, there is a significant demand for agents to communicate with each other effectively. Nowadays, the communication architectures that commonly used include the message architecture [2,6] and blackboard communication architecture [3,5].

In the message architecture, there is a direct exchange of messages between agents using a common language in a conversational style, where the sending agent specifies for whom the message is intended and the receiving agent accepts the message when it is reached. Obviously, the message architecture is simple and effective. However, it also has some disadvantage [1]. One of the problems with this architecture is that the overheads in message transfer are quite high if the agent community is large, and another problem is the implementation complexity.

By contrast, in the blackboard communication architecture, information is made available to all agents in the system through a common information space and there is no direct communication between agents. Obviously, the message overheads and implementation complexity of blackboard architecture are relative low. Blackboard communication architecture is well suited for dynamic and large agent systems.

Blackboard communication architecture includes central blackboard architecture and distributed one, as shown in Fig. 1. Central blackboard architecture is simple. However, in this architecture the blackboard is subject to become the 'performance bottleneck' of agent system. A popular way of enhancing communication architecture is to implement distributed blackboard architecture, in which some sub-blackboards are set in the system and each sub-blackboard takes charge of the communications of some agents [1]. Here agents are organized into some federated systems where agents do not communicate directly with each other but through their respective sub-blackboards. The agents in a federated system surrender their communication autonomy to the sub-blackboards and the sub-blackboard takes full responsibility for their needs. Fig. 1 shows a simple federated multi-agents system in which there are three multi-agent sub-systems (i.e. federated systems) with agents in each sub-system controlled by a sub-blackboard.

* Corresponding author. Tel.: +86-21-6564-3235; fax: +86-21-6564-7894.
E-mail address: jiangyichuan@yahoo.com.cn (Y.C. Jiang).

0141-9331/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.micpro.2004.06.001

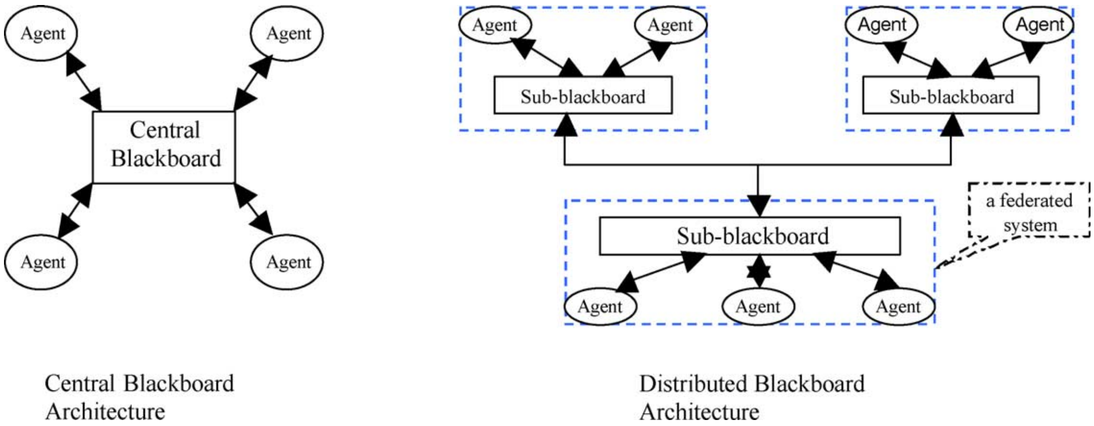

Fig. 1. Blackboard communication architecture.

The sub-blackboards communicate among themselves to express the needs of their respective agents.

Nowadays, there is an emerging phenomenon that network topology is changed after the initial setting of a system. Therefore, the agent communication architecture in such situation needs to be adjusted accordingly. Moreover, the variety of agent cooperation relations also demands that the agent communication architecture should be adjusted effectively when agent cooperation relations are changed. However, there are few researches on the adjusting of agent communication architecture for network topology and agent cooperation relations, currently and, agent communication architecture is fixed after its initial setting, which may influence the system performance when network topology or agent cooperation relations are changed.

To solve the above problem, based on distributed blackboard architecture and graph theory, this paper presents a novel mechanism for adjusting agent communication architecture. According to the current network topology and agents cooperation relations, this mechanism can adjust the setting of distributed blackboard architecture (e.g. sub-blackboard locality, federated system construction, message transfer path, etc.). The new adjusted architecture performs better than the initial architecture in new network topology and agent cooperation relations, which is testified by our simulation experiments.

The rest of the paper is organized as follows. Section 2 presents the related definitions. Section 3 addresses the detailed adjusting mechanism of agent communication architecture. Section 4 makes analysis and validation based on Mobile Ambients. Section 5 describes simulation experiment. Then the conclusions are summarized in Section 6.

2. Related definitions

To describe the cooperation relations among agents in multi-agent system, we present the concept of Agents Cooperation Relations Graph (ACRG).

Definition 1. Agents Cooperation Relations Graph (ACRG): ACRG $=(V,E)$, where:

$-V=\{a_1,a_2,\ldots,a_n\}$, where a_i denotes agent i;

$-E\subseteq V\times V$; $E=\{e_1,e_2,\ldots,e_n\}$, where e_i $=(a_u,a_v)$ denotes the cooperation relation between *agent* a_u and agent a_v.

From the example of ACRG in Fig. 2, we can see that a_1 cooperates with a_2,a_4,a_5 and a_8; a_2 cooperates with a_1,a_5 and a_6; a_3 cooperates with a_5,a_8 and a_9, etc.

Definition 2. Network Topology Graph (NTG) denotes the network topology on which agent system runs. NTG $=(V',E')$, where:

$-V'=\{N_1,N_2,\ldots,N_n\}$, N_i denotes the network node;

$-E'\subseteq V'\times V'$, $E'=\{e'_1,e'_2,\ldots,e'_n\}$.

$e'_i=(N_u,N_v)$ which denotes the interconnection relation *between* network nodes N_u and N_v.

E.g. Fig. 3 is a NTG where the agent system runs. From Fig. 3, we can see that agent a_1 locates on the node N_1,a_2 locates on N_3,a_3 locates on N_4,a_4 locates on N_6,a_5 locates on N_7,a_6 locates on N_8,a_7 locates on N_9,a_8 locates on N_{10}, and a_9 locates on N_{11}.

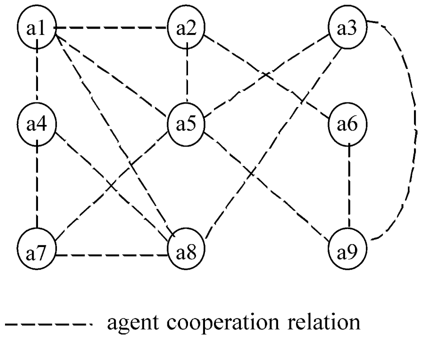

Fig. 2. An agent cooperation relation graph (ACRG).

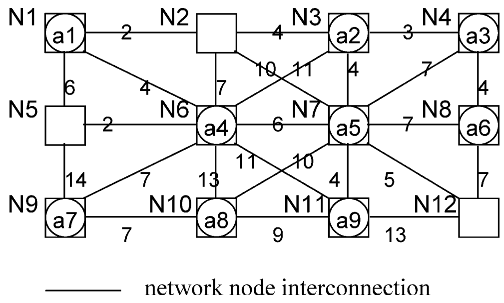

Fig. 3. A network topology graph and agent system.

Therefore, Fig. 3 can definitely describe a multi-agent system with the cooperation relations that shown in Fig. 2.

Definition 3. Agents Communication Topology Graph (ACTG): ACTG denotes the topology graph that is composed of agent communication paths in the underlying network environment. Since the agent communication often goes along the shortest path between the nodes on which agents locates, ACTG is composed of the shortest paths between nodes on which cooperated agents locates.

ACTG $=(V'', E'')$, where:

- if the agent on N_i cooperates with the one on N_j, then $N_i, N_j \in V''$;
- if the agent on N_i cooperates with the one on N_j, then the edges along the shortest path between N_i and N_j are attributed to E''.

The ACTG of the agents system shown in Figs. 2 and 3 can be seen in Fig. 5.

From above description, we know that ACTG can describe the agent communication situation well, and the agent communication architecture should be adjusted based on ACTG.

Therefore, the question of ‘Adjusting mechanism of agent communication architecture’ can be described as follows: when network topology or agent cooperation relations are changed, the ACTG should be computed according to current NTG and ACRG, and the setting of distributed blackboard architecture (e.g. sub-blackboard locality, federated system construction, message transfer path, etc.) is adjusted on the base of the new ACTG.

3. Adjusting of agent communication architecture

3.1. The overall framework

To adapt to the change of network topology and agent cooperation relations, we present the adjusting mechanism of agent communication architecture. The mechanism can adjust the agent communication architecture according to the current agents cooperation situation (ACRG) and network topology (NTG). Through this mechanism, agents can implement effective communication in the light of the new adjusted architecture, thereby the agent communication adaptation for new network topology and agent cooperation relations can be satisfied. The overall flow chart of the mechanism is shown as Fig. 4.

In our adjusting mechanism, there is a management station in the network. The management station can monitor the change of underlying network topology and agent cooperation relations, and adjust the distributed blackboard communication architecture accordingly.

We will introduce the principle of the mechanism in the following sub-sections.

3.2. Computing the ACTG

When the network topology is changed, we compute the ACTG according to the new NTG and current Agent Cooperation Relations Graph (ACRG).

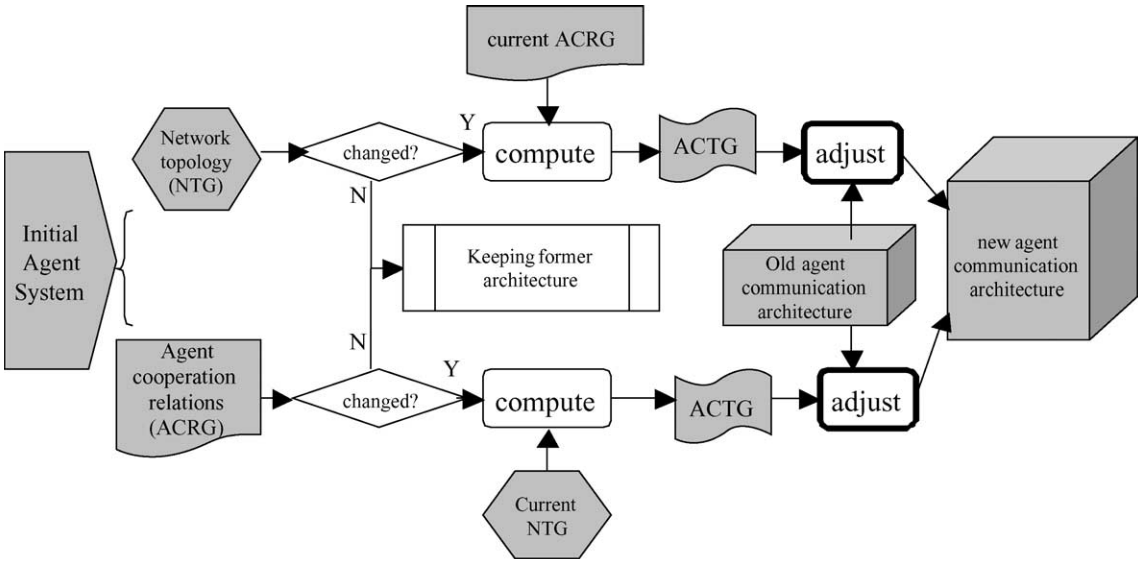

Fig. 4. Flow chart of the adjusting mechanism.

ACTG is composed of the shortest paths between nodes on which cooperated agents locate. Therefore, let a_i cooperates with a_j, and a_i locates on N_i, a_j locates on N_j, then the shortest path between N_i and N_j is attributed to ACTG.

We can describe the NTG as a weighted graph. Let the weight on each edge be the communication cost from a vertex of the edge to another vertex. The memory structure of NTG is an adjacency matrix cost[i,j], where if there is a edge from N_i to N_j, cost[i,j] = the weight of the edge, or else cost[i,j] = ∞.

The detailed ACTG computing process can be seen in Algorithm 1.

Algorithm 1. *Computing the Agents Communication Topology Graph (ACTG).*

```
void short_path (cost, dist, path)
/*cost[i,j] denotes the adjacency matrix of NTG, dist[i, j] denotes the distance of the shortest path between Ni and Nj, path[i, j] denotes the shortest path between Ni and Nj. */
{
for (int i = 1; i < = n; i++)
    for (int j = 1; j < = n; j++)
        {dist [i, j] = cost[i, j];
        if (dist[i, j] <max
            path[i, j] = [i] + [j];
        }
for (int k = 1; k < = n; k++)
    for (int i = 1; i < = n; i++)
        for (int j = 1; j < = n; j++)
            if (dist[i, k] + dist[k, j] < dist[i, j])
{ dist[i, j] = dist[i,k] + dist [k,j];
path[i, j] = path[i, k] + path[k, j];
            }
}
main ACTG_computing ()
{
ACTG = {};
short_path (cost, dist, path);
    for (int i = 1; i < = n; i++)
        for (int j = 1; j < = n; j++)
            { if the agent on Ni cooperates with the agent on Nj
            then ACTG = ACTG + path[i, j]
            }
}
```

We can compute the ACTG of the agents system in Fig. 3 with the cooperation relations in Fig. 2 according to Algorithm 1, the result is shown as Fig. 5. In Fig. 5, the over striking line denotes the communication paths. To improve the readability, other edges are also added to the graph with dashed line.

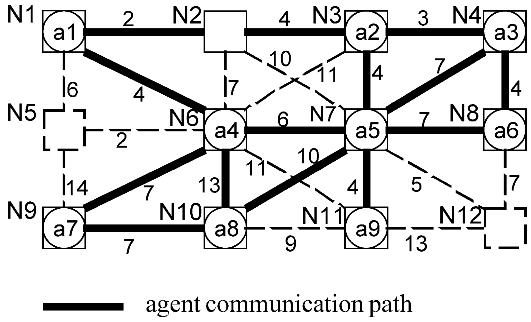

Fig. 5. Agent communication topology graph (ACTG).

3.3. Constructing the spanning tree of ACCTG

In some hierarchical agent systems, there is a management agent that controls other agents. In these systems, many communications take place between the management agent and other agents. Under such situation, we can adjust the agent communication architecture according to the spanning tree of ACTG. The validity of this method can be testified in the simulation experiment.

We select the node on which the management agent locates as the original one, and adopt the well-known *Prim's Algorithm for finding the minimum cost spanning tree* [4] to construct the spanning tree of ACCTG. The detailed algorithm can be seen in Ref. [4].

Therefore, the spanning tree of Fig. 5 is shown as Fig. 6.

3.4. Adjusting the setting of distributed blackboard architecture

After ACTG is computed, we can adjust the setting of distributed blackboard architecture. The process includes the adjustments of the following three parts: sub-blackboard locality, federated system construction, message transfer path construction.

3.4.1. Adjustment of sub-blackboard locality

When network topology is changed, the sub-blackboard locality should be changed too. We select some nodes as sub-blackboard localities according to the ACTG or its spanning tree. We can consider the problem under two situations: one situation is that all agents are equal, and there aren't any management agents in the system; another situation is that agent organization is hierarchical, and there

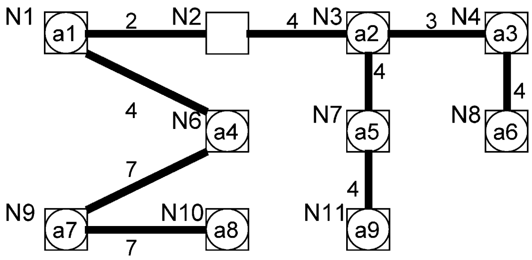

Fig. 6. Spanning tree of ACTG.

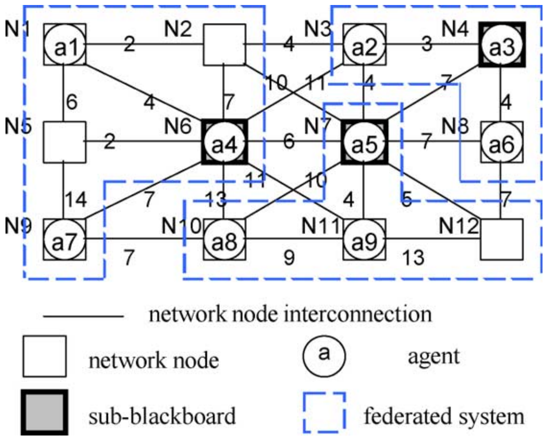

Fig. 7. Federated systems 1.

is a management agent in the system. Under the first situation, we can select some nodes as sub-blackboards localities according to the ACTG. Under the second situation, we can select the nodes as sub-blackboards localities according to the spanning tree of ACTG.

Then how to select sub-blackboard locality according to ACTG or its spanning tree? We can select the nodes with high degree[1] in ACTG or its spanning tree. In this way the communication cost can be lessened, which is also testified by our simulation experiments.

Let the number of sub-blackboards is 3, now we take the agent system in Fig. 2 and Fig. 3 as an example.

If all agents are equal and there aren't any management agents in the system, we select N_6, N_7 and N_4 as sub-blackboard localities since these three nodes have high degrees in Fig. 5, as shown in Fig. 7.

If the agent organization is hierarchical, and there is a management agent a_1 in the system, we can select N_1, N_3 and N_6 as sub-blackboard localities as these three nodes have high degrees in Fig. 6, as shown in Fig. 8.

3.4.2. Adjustment of federated system construction

After the adjusting of sub-blackboard locality, agents should be organized into some federated systems where agents do not communicate directly with each other but through their respective sub-blackboards.

Which sub-blackboard will an agent surrender its communication autonomy to? In this paper, for simplicity, each agent can select the nearest sub-blackboard to surrender its communication autonomy.

Otherwise, considering the request of mobile agent, those nodes on which no agents locate now should also select a sub-blackboard. Obviously, they should select the nearest blackboard, too. In Fig. 7, N_2 and N_5 select the sub-blackboard on N_6, and N_{12} selects the sub-blackboard on N_7. In Fig. 8, N_2 selects the sub-blackboard on N_1, N_5 selects

[1] The degree of a vertex in a graph is the number of the edges that connect the vertex.

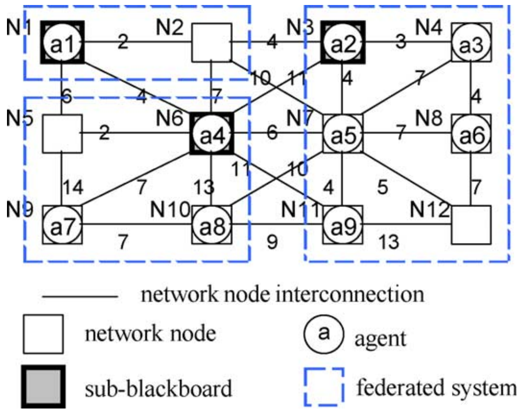

Fig. 8. Federated systems 2.

the sub-blackboard on N_6, and N_{12} selects the sub-blackboard on N_3.

Now we take the agent system in Figs. 2 and 3 as an example, if the sub-blackboards are located on N_6, N_7 and N_4, the constructed federated systems are shown in Fig. 7; if the sub-blackboards are located on N_1, N_3 and N_6, the constructed federated systems are shown in Fig. 8.

3.4.3. Adjustment of message transfer path construction

Now we construct the factual agent message transfer paths in the federated systems.

We compute the shortest path from the nodes to the sub-blackboard in each federated system and the shortest paths among sub-blackboards. Thereby the factual agent message transfer path can be composed of those shortest paths, shown as Figs. 9 and 10. The agents in a federated system communicate with the sub-blackboard and the sub-blackboards communicate among themselves to express the needs of their respective agents.

The format of message is as follows: sender location, receiver location, content. Among those the sender location

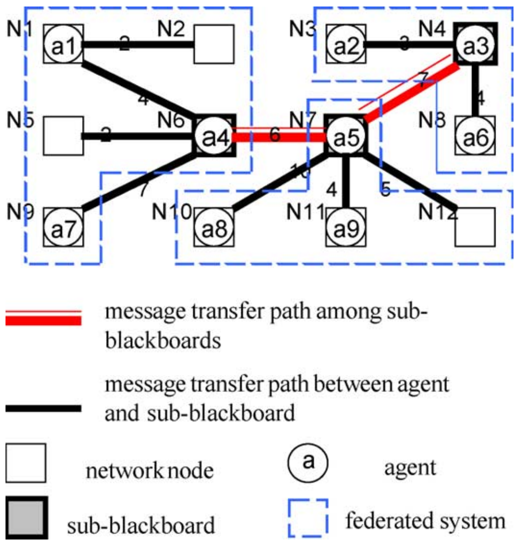

Fig. 9. Agent communication architecture (1).

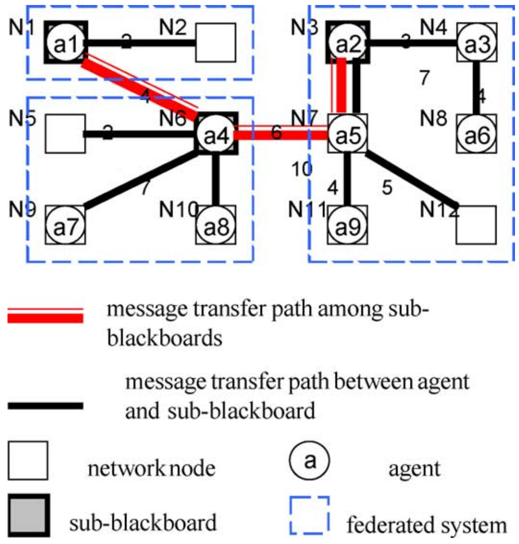

Fig. 10. Agent communication architecture (2).

and receiver location both contain two parts: name of federated system and name of node on which agent locates.

Sender agent firstly transfers the message to the sub-blackboard, if the sub-blackboard finds that the receiver location doesn't belong to it's federated system, it forwards the message to the sub-blackboard that controls receiver agent, and then the receiver sub-blackboard re-transfers the message to the receiver agent.

3.4.4. Coping with mobile agent

There are two situations under which agent moves. One is that agent moves to another node within the same federated system. The other is that agent moves to another node beyond the federated system. Under the first situation, the agent only notifies the place where it is now located to the sub-blackboard. Under the second situation, when the agent moves to a new federated system, the agent firstly notifies the name of its home sub-blackboard (Home-blackboard) to the sub-blackboard of current system (Proxy-blackboard), then proxy-blackboard tells the Home-blackboard that the agent come here. This process can be called Agent Location Renew Process (ALRP). Wherever the mobile agent moves, it registers new address on its Home-blackboard through ALRP.

Now, let other agent (e.g. *a*) want to communicate with the mobile agent (e.g. *m*), and the Home-blackboard of *m* be B_1,the Proxy-blackboard of *m* be B_2. At the first communication, *a* firstly transfers the message to B_1 and B_1 then transfers the message to B_2, then B_2 transfers the message to *m*. Then B_1 tells *a* about the address of the B_2; and B_2 becomes the new Home-blackboard of *m*. At afterwards communication, *a* can transfer the message to B_2 directly, and B_2 forwards the message to *m*.

Now we take Fig. 9 as an example. If a_1 moves to N_{12}, a_1 firstly tells the sub-blackboard on N_7 that its Home-blackboard locates onN_6; then the sub-blackboard on N_7 tells the sub-blackboard on N_6 that a_1 has moved into its federated system.

- Let a_7 want to communicate with a_1, it firstly sends the message to N_6, then N_6 sends the message to N_7, at last N_7 sends the message to a_1.
- Let a_9 want to communicate with a_1, it firstly sends the message to N_7, since N_7 knows that a_1 is in its realm so N_7 sends message to a_1 directly.
- Let agent *ax* in other federated system of *Bx* want to communicate with a_1, it firstly sends the message to *Bx*, then *Bx* sends the message to N_6, and N_6 sends the message to N_7, then at last N_7 sends the message to a_1.

The whole processes can be seen in Fig. 11.

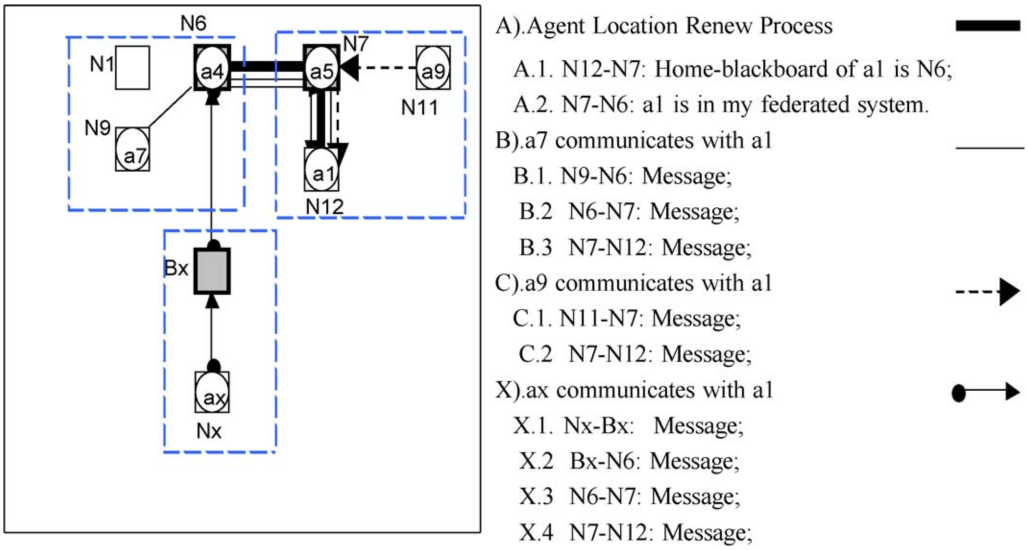

Fig. 11. The example of agents message transfer process.

4. Analyses and validation based on Mobile Ambients

Mobile Ambients Calculus was developed by Cardelli and Gordon as a formal framework to study issues of mobility and migrant code [7]. Boxed Ambients are a variant of Mobile Ambients, from which they inherit the primitives in and out for mobility, with the exact same semantics. But, Boxed Ambients also rely on a completely different model of communication, which results in dropping the open capability [8]. In this paper, we combine the original Mobile Ambients and Boxed Ambients, and reserve the open primitive to analyze and validate the correctness of the adjusted communication architecture.

To test the correctness of the adjusted agent communication architecture, we make analysis for the communication architecture in Fig. 10 based on Mobile Ambients and Boxed Ambients.

We denote the federated system with the sub-blackboard locating on N_1 as b_1, the federated system with the sub-blackboard locating on N_6 as b_2, the federated system with the sub-blackboard locating on N_3 as b_3. We call the three sub-blackboards on N_1, N_6, and N_3 as Blackboard$_1$, Blackboard$_2$ and Blackboard$_3$ respectively.

In followings, $W(x)$ denotes that agent transfers message to the Home-blackboard, $R(x)$ denotes that Home-blackboard transfers message to agent, and $S(x)$ denotes that the message is stored in the blackboard.

4.1. The communication between two agents within the same federated system

In Fig. 10, let a_7 want to send message to *a8*, now we describe the ingredients of the communication process based on Mobile Ambients, shown follows:

$$a_7 \triangleq b_2[\langle x_1\rangle.W(x_1)];\ \text{Blackboard}$$
$$\triangleq b_2[(x_2).S(x_2)|\langle x_2\rangle];\ a_8 \triangleq b_2[(x_3).R(x_3)] \quad (1)$$

The whole communication process can be simulated by Mobile Ambients Calculus as follows:

$$\text{Communication} \triangleq a_7|\text{Blackboard}|a_8$$
$$\equiv b_2[\langle x_1\rangle.W(x_1)|(x_2).S(x_2)|\langle x_2\rangle|(x_3).R(x_3)]$$
$$\rightarrow b_2[W(x_1)|S(x_1)|\langle x_1\rangle|(x_3).R(x_3)]$$
$$\rightarrow b_2[W(x_1)|S(x_1)|R(x_1)] \quad (2)$$

From Eq. (2), it is obvious that a_7 can successfully transfer the message to a_8, which proves that the architecture in Fig. 10 is correct for the agent communication within a federated system.

4.2. The communication between two agents in different federated systems

In Fig. 10, let a_7 want to transfer message to a_3, now we simulate the communication process based on Mobile Ambients and Boxed Ambients Calculus. Here, since Blackboard$_2$ not only communicates with the agents in its own federated system but also communicates with other sub-blackboards, the ambient can't be denoted as b_2 but two ambients: one is b_2' that denotes the communication ambient in which the agents within the federated system communicates with Blackboard$_2$, other is b_2'' that denotes the communication ambient in which Blackboard$_2$ communicates with other sub-blackboards. Similarly, b_3' denotes the communication ambient in which the agents within the federated system communicates with Blackboard$_3$, and b_3'' denotes the communication ambient in which Blackboard$_3$ communicates with other sub-blackboards.

Now we describe the ingredients of the communication process based on Mobile Ambients, shown as follows:

$$a_7 \triangleq b_2'[\langle x_1\rangle.W(x_1)];\ \text{Blackboard}_2$$
$$\triangleq (x_2)^{b_2'}.(b_2'[S(x_2)]|b_2''[\langle x_2\rangle|\text{in } b_3'']);\ \text{Blackboard}_3$$
$$\triangleq (x_3)^{b_2''}.(b_3''[S(x_3)|\text{open } b_2'']|b_3'[\langle x_3\rangle]);\ a_3$$
$$\triangleq (x_4)^{b_3'}.(b_3'[R(x_4)]) \quad (3)$$

The whole cooperation communication process can be simulated by Mobile Ambients and Boxed Ambients process, shown as follows:

$$\text{Communication} \triangleq a_7|\text{Blackboard}_2|\text{Blackboard}_3|a_3$$
$$\equiv b_2'[\langle x_1\rangle.W(x_1)]|(x_2)^{b_2'}.(b_2'[S(x_2)]|b_2''[\langle x_2\rangle|\text{in } b_3''])$$
$$\times|(x_3)^{b_2''}.(b_3''[S(x_3)|\text{open } b_2'']|b_3'[\langle x_3\rangle])|(x_4)^{b_3'}.b_3'[R(x_4)]$$
$$\rightarrow b_2'[W(x_1)]|(b_2'[S(x_1)]|b_2''[\langle x_1\rangle|\text{in } b_3''])$$
$$\times|(x_3)^{b_2''}.(b_3''[S(x_3)|\text{open } b_2'']|b_3'[\langle x_3\rangle])|(x_4)^{b_3'}.b_3'[R(x_4)]$$
$$\rightarrow b_2'[W(x_1)]|b_2'[S(x_1)]|b_3''[S(x_1)]|b_3'[\langle x_1\rangle]|(x_4)^{b_3'}.$$
$$\times b_3'[R(x_4)] \rightarrow b_2'[W(x_1)]|b_2'[S(x_1)]|b_3''[S(x_1)]|b_3'[R(x_1)] \quad (4)$$

From Eq. (4), it is clear that a_7 can successfully transfer the message to a_3, which proves that the architecture in Fig. 10 is correct for the agent communication between two different federated systems.

4.3. The communication between the mobile agents

In Fig. 10, let a_7 communicates with a_8, but a_8 now migrates onto N_{12}. In such a communication process, after

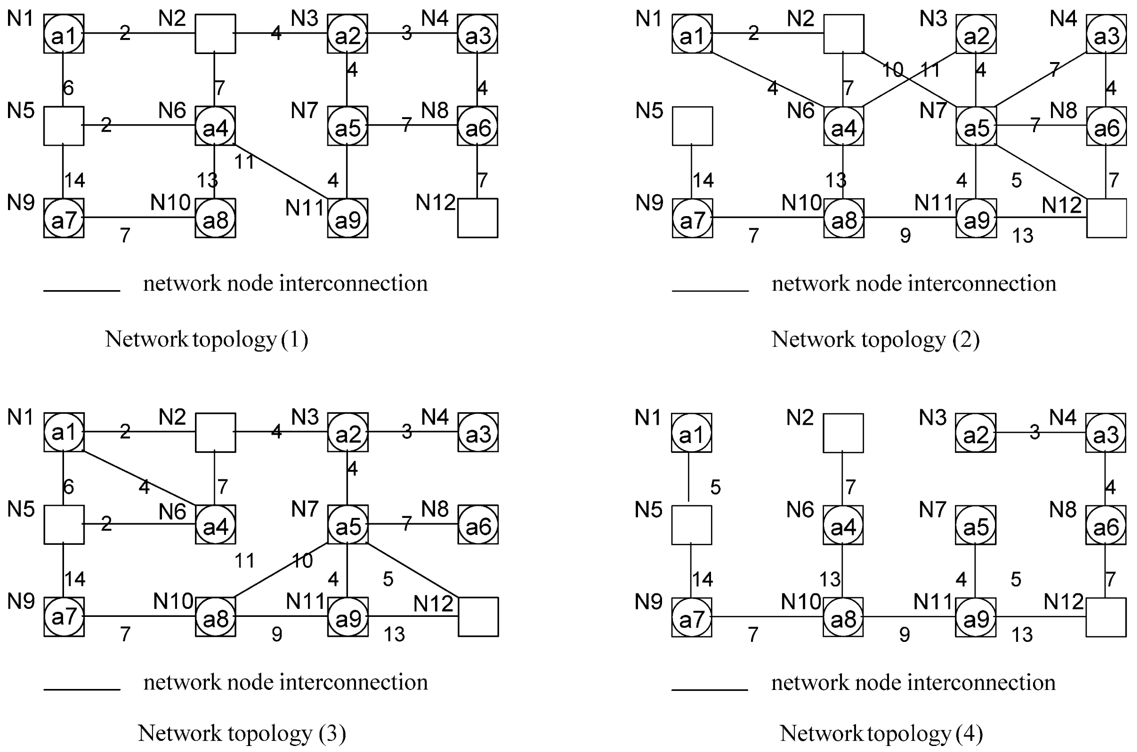

Fig. 12. Changing network topology for simulation experiment.

A_8 migrates onto N_{12}, the ALRP is executed according Section 3.4.4. Then the communication process between A_7 and A_3 is similar to the one of Section 4.2.

From the above three simulation processes based on Mobile Ambients, we can see that the adjusted communication architecture is correct for the following three situations: the communication between two agents within the same federated system; the communication between two agents in different federated systems; the communication between the mobile agents. Therefore, the adjusting mechanism can produce correct communication architecture.

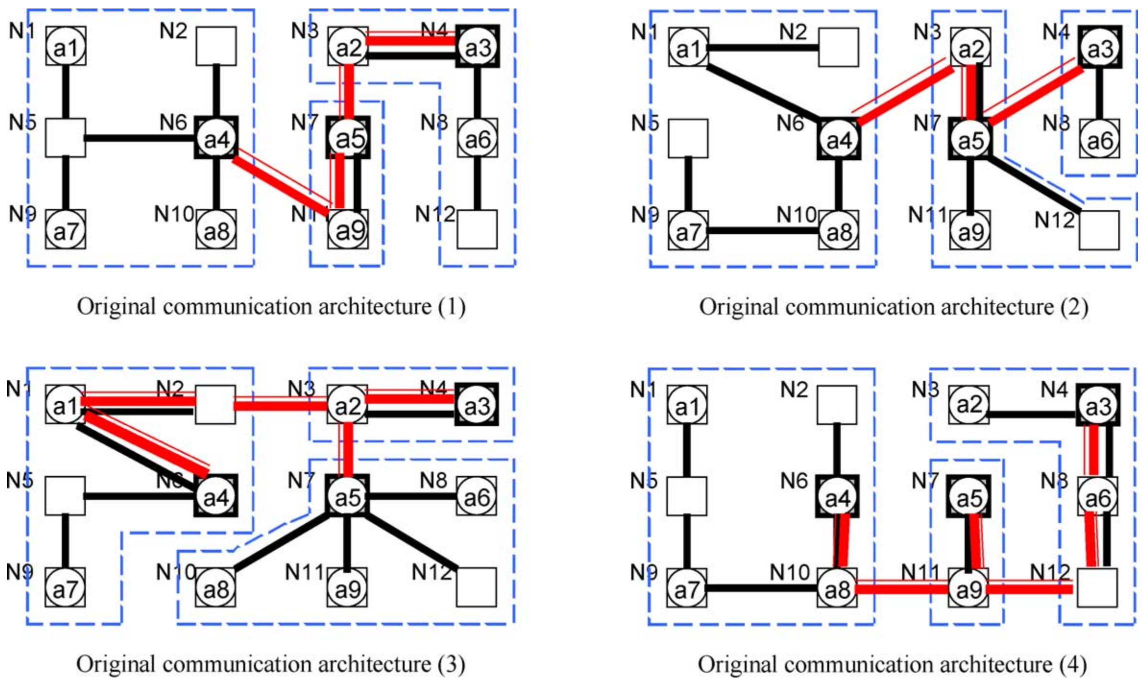

Fig. 13. Communication architecture by using initial distributed blackboards (for changed network topology).

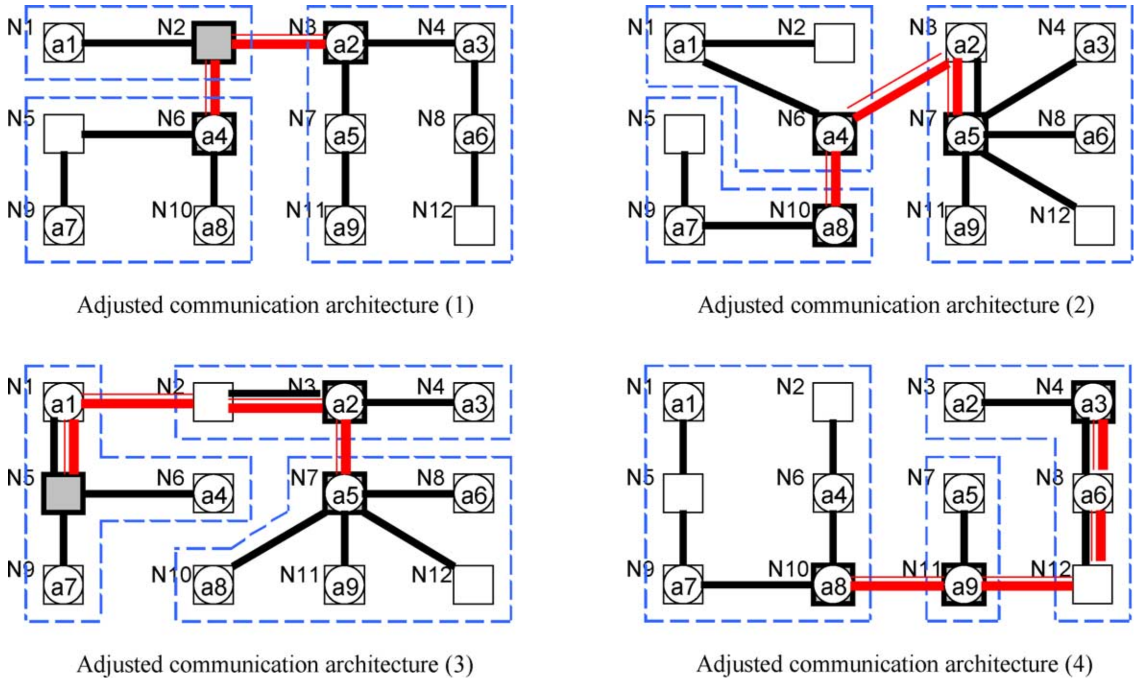

Fig. 14. Communication architecture after adjusting (for changed network topology).

5. Simulation experiments

For the purpose of our experiment, we have developed a minimal platform that provides the basic functions required to program agents. By studying from the method of [9], we have implemented a prototype which is developed with Tcl/Tk, Tclx, Tix and Binprolog [10–12]. And the prototype was also partly based on the work of Aglets Software Development Kit v2 (Open Source release) [13,14].

In order to show how effectively our proposed mechanism can work, we compare the performance of the original distributed blackboard architecture and the adjusted one when network topology and agent cooperation relations are changed, shown in Section 5.1, 5.2 and 5.3.

5.1. Test for changed network topology

We take the ones in Figs. 2 and 3 as the initial agent cooperation relations and network topology for our experiments. The initial distributed blackboard architecture is shown in Fig. 9.

Then, we change the network topology as Fig. 12; the agent cooperation relations keep unchanged, shown as Fig. 2. If the initial distributed blackboard architecture is still used (i.e. the localities of sub-blackboards are kept the same), the federated systems and message transfer paths are shown in Fig. 13. If we adjust the distributed blackboard architecture according to the mechanism presented here, the federated systems and message transfer paths are shown in Fig. 14.

In the simulation experiment, all agents cooperate according to the ACRG in Fig. 2. Let each agent send a message to its cooperation partner, now we test the total communication time under the initial architecture and the adjusted architecture.

The test results are shown as Fig. 15. From Fig. 15, we can conclude that: when network topology is changed, the total agent communication time in the adjusted architecture is less than the one in the original architecture that uses the initial distributed blackboard. Therefore, the adjusting mechanism is efficient when network topology is changed.

5.2. Test for changed agent cooperation relations

Now we test the efficiency of the adjusting mechanism when agent cooperation relations are changed.

The agent cooperation relations are changed as Fig. 16; the network topology keeps unchanged, shown as Fig. 3.

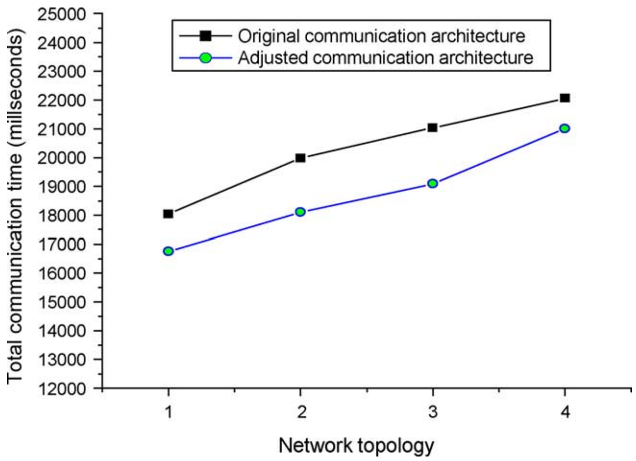

Fig. 15. Test for changed network topology.

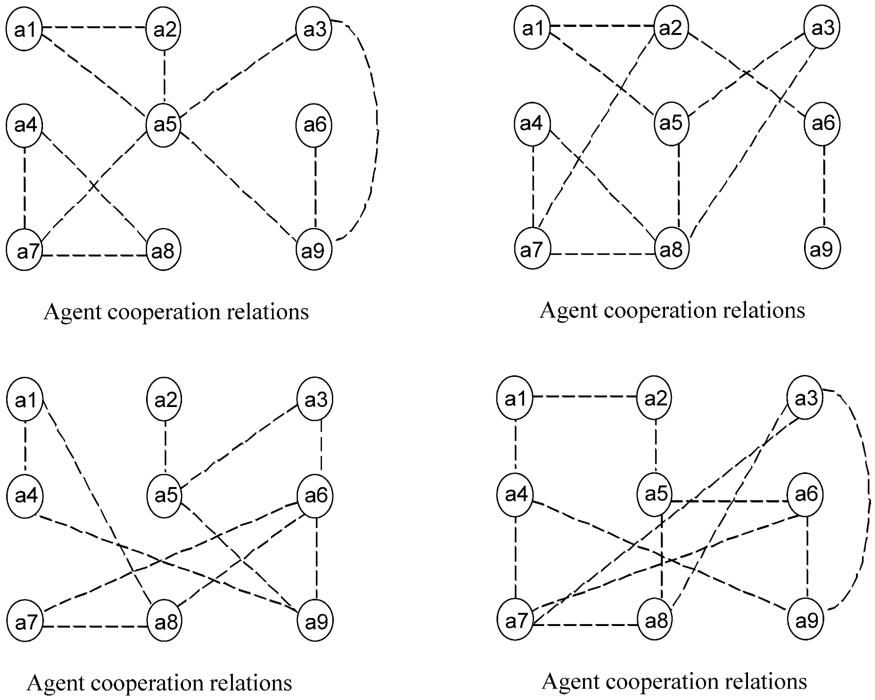

Fig. 16. Changing agent cooperation relations for simulation experiment.

If the initial distributed blackboard architecture is still used (i.e. the localities of sub-blackboards are kept the same), the federated systems and message transfer paths are the same as the one in Fig. 9. If we adjust the distributed blackboard architecture according to our mechanism, the federated systems and message transfer paths are shown in Fig. 17.

We test the total communication time under the initial architecture and adjusted architecture. The test results are shown as Fig. 18. From Fig. 18, we can conclude that: when agent cooperation relations are changed, the total agent communication time in the adjusted architecture is less than the one in the original architecture that uses

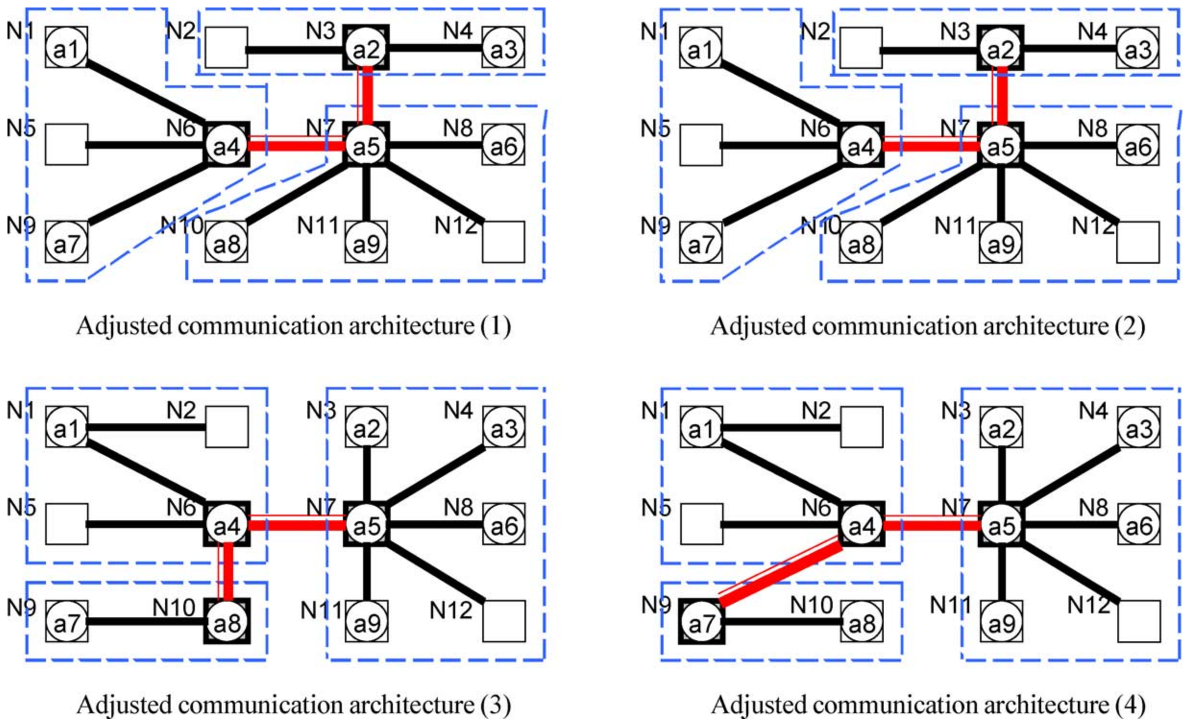

Fig. 17. Communication architecture after adjusting (for changed agent cooperation relations).

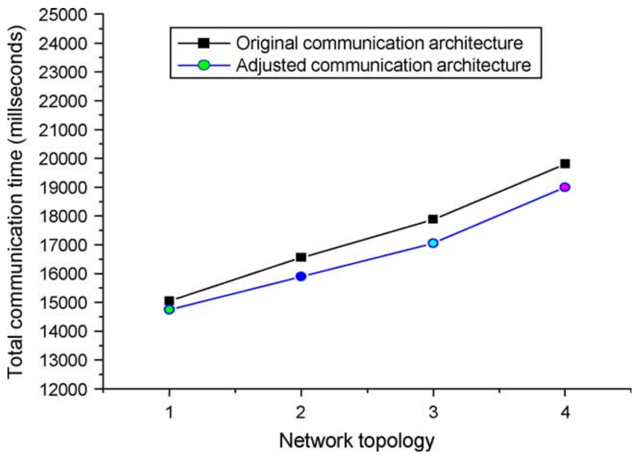

Fig. 18. Test for changed agent cooperation Relations.

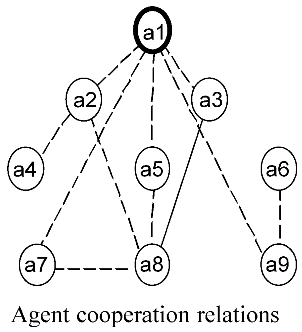

Fig. 19. Agent cooperation relations with a management agent.

the initial distributed blackboard. Therefore, the adjusting mechanism is efficient when agent cooperation relations are changed.

5.3. Test for hierarchical agent cooperation relations

Now we test the adjusting mechanism in the hierarchical agent system that has a management agent. The agent cooperation relations graph is tree-like, shown as Fig. 19. We compare the performance of three communication architectures when network topology is changed: *a*. The one that uses the initial sub-blackboards (i.e. the localities of sub-blackboards keep unmovable); *b*. The one adjusted according to the ACTG; *c*. The one adjusted according to the spanning tree of the ACTG.

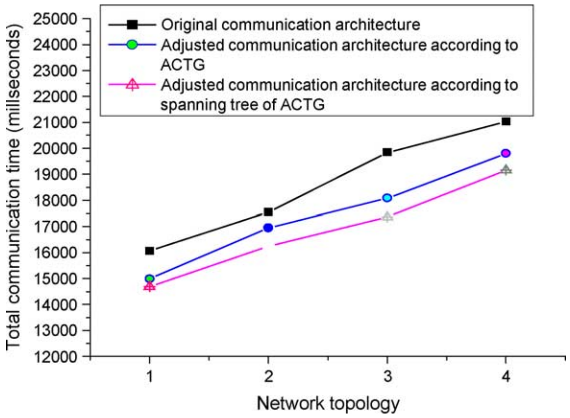

Fig. 20. Test for hierarchical agent system.

The description of the adjusted architectures was omitted here for brevity. The test results are shown in Fig. 20, from which we can see that: the total communication time of *b* is less than that of *a*, and *c* consumes the least amount of communication time. Therefore, when agent system is hierarchical and there is a management agent, we should adjust the communication architecture according to the spanning tree of the ACTG.

6. Conclusion

Based on graph theory and distributed blackboard architecture, a novel mechanism for adjusting agent communication architecture is presented in this paper. When network topology or agent cooperation relations are changed, this mechanism can be used to adjust the distributed blackboard architecture accordingly. The new adjusted architecture is correct and can perform better than the original one in new network topology or agent cooperation relations, which is testified by the Mobile Ambients Calculus analysis as well as the simulation experiments.

However, in the adjusting mechanism presented here, the adjusting process is time-consuming. If network topology changes very frequently and the interval is short, the current mechanism can't satisfy the requirement. Therefore, our future works will focus on the improvement of the real-time property of adjusting mechanism.

Additionally, when the agent cooperation relations are only trivially changed, the performance difference isn't distinct between adjusted communication architecture and the original communication architecture. Therefore, in factual application we adjust the communication architecture only when the agent cooperation relations are changed substantially.

In our adjusting mechanism, there is a management station in the network to monitor the change of underlying network topology and agent cooperation relation, and adjust the agents communication architecture. The setting of a management station can implement the adjusting mechanism effectively and simply, but it may also bring out single point failure. Therefore, our future work will also focus on the distributed implementation of the adjusting mechanism.

References

[1] Cyprian Foinjong Ngolah, A tutorial on agent communication and knowledge sharing. Available at http://www.enel.ucalgary.ca/People/far/Lectures/SENG60922/PDF/tutorials/2002/Agent_Communication_and_Knowledge_Sharing.pdf

[2] FIPA Agent Message Transport Protocol for IIOP Specification (XC00075D), October 2000, available at http://www.fipa.org/specs/fipa00075/XC00075D.doc
[3] Francois Charpillet Philippe Lalanda, Jean-Paul Haton, A real time blackboard based architecture, In ECAI92 10th European Conference on Artificial Intelligence, 1992, pp. 262–266, ECAI, August 92.
[4] B.R. Preiss, Data structures and algorithms with object-oriented design patterns in C++, John Wiley & Sons, New York, 1999.
[5] V. Botti, A. Barber, A temporal blackboard for a multi-agent environment, Data & Knowledge Engineering 15 (1995).
[6] D. Deugo, Mobile agent messaging models, In: Proceeding of Fifth International Symposium on Autonomous Decentralized Systems March 26–28, 2001.
[7] L. Cardelli, D. Gordon, Mobile ambients, Foundations of Software Science and Computational Structures, LNCS No. 1378, Springer, Berlin, 1998, pp. 140–155.
[8] M. Bugliesi, G. Castagna, S. Crafa, Boxed ambients, In 4th International Conference on Theoretical Aspects of Computer Science (TACS'01), vol. 2215, Springer-Verlag, Berlin, 2001.
[9] W. Cao, C.G. Bian, G. Hartvigsen, Achieving efficient cooperation in a multi-agent system: the twin-base modeling, In Proceeding of Cooperative Information Agents (CIA'97), Germany, LNAI, vol. 1202, Springer, Berlin, 1997, pp. 210–221.
[10] Tcl Developer Xchange: The Tcl and Tk Toolkit, Tcl /Tk Version 8.4.4. URL: http://www.tcl.tk/software/tcltk/8.4.html
[11] Tcl Contributed Archive: Extended Tcl (TclX), Version 7.0a. URL: http://www.neosoft.com/tcl/. Neosoft Company, 2003.
[12] Home Page for Tix: Tk Interface eXtension: Tix-Programming Library for the Tk Toolkit, Version 4.0. URL: http://tix.mne.com/. 2003.
[13] Internet Programming Toolkit: Binprolog Version 1.99. URL: http://www.binnetcorp.com/BinProlog/, 1995.
[14] Aglets Software Development Kit: Aglets Software Development Kit v2 (Open Source). URL: http://www.trl.ibm.com/aglets/. 2002.

Yichuan Jiang was born in 1975. He received his MS degree in computer science from Northern Jiaotong University, China in 2002. He is currently a PhD candidate in computer science of the Department of Computing & Information Technology, Fudan University, China. His research interests include mobile agent system, artificial intelligence and network security.

Zhengyou Xia was born in 1974. He received his MS degree in fuse technology from Nanjing University of Science & Technology in 1999, and received his PhD degree in computer science from Fudan University in 2004. He is currently a lecturer in the Department of Computer, Nanjing University of Aeronautics and Astronautics, China. His research interests include information security, mobile agent and active network.

Siyong Zhan was born in 1950. He is now a professor and PhD supervisor, and also the director of the Center of Networking & Information Engineering of Fudan University, China. His research interests include network system, mobile agent system and network security.

Yiping Zhong was born in 1953. She is now an associate professor, and also the associate director of the Department of Computing & Information Technology of Fudan University, China. Her research interests include network system and distributed system.