ELSEVIER

Advances in Engineering Software 36 (2005) 59–66

ADVANCES IN ENGINEERING SOFTWARE

www.elsevier.com/locate/advengsoft

Autonomous trust construction in multi-agent systems—a graph theory methodology

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

[a]*Department of Computing and Information Technology, Center of Networking and Information Engineering, Room 409, Yifu Building, No 220 Handan Road, Fudan University, Shanghai 200433, China*

[b]*Department of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210043, China*

Received 9 February 2004; revised 23 July 2004; accepted 27 August 2004
Available online 31 October 2004

Abstract

Trust mechanism always has two popular architectures: centralized fashion and distributed fashion. However, those two architectures are not well suited for multi-agent system since they cannot achieve the trust management autonomy. To achieve the trust management autonomy, the paper presents an autonomous trust construction model based on graph theory methodology. The presented model adopts the graph to describe the trust information, and uses the graph combination and path searching to construct the trust relation. Every agent can implement trust management autonomously; agent system can construct the global trust concept by the combination of trust information among agents; an agent can achieve the trust relation with other agent by trust path searching or trust negotiation. The simulation experiment results prove that the autonomous trust construction based on graph theory methodology is effective.

Keywords: Multi-agent systems; Autonomous; Trust; Graph theory; Security

1. Introduction

Multi-agent technology is a research focus of artificial intelligence, which can make agents cooperate to perform the assigned task. Multi-agent technology can support agent migration among hosts and make network application more flexible and effective [1]. However, the trust management of multi-agent remains as an unsolved question. If we do not solve such question well, the multi-agent technology cannot achieve effective application in practice [2].

Now there have been some relative research works about the trust management of multi-agent, which are often based on the traditional security mechanism, such as access control list, role-based access control, PKI, etc. [3]. In those works, there always was a central repository to provide the access control and security trust information to individual agent or agents group, or has a trusted third party to provide the trust mechanism. We call those relative works as *Central Trust Mechanism*, which adopts the Trusted Authority (TA) or trusted third party to manage trust. Central Trust Mechanism is simple and effective, but it has many drawbacks, such as single point failure, the requirement of infrastructure, TA may become performance neck, etc. Obviously, the Central Trust Mechanism cannot satisfy the requirement for the mobility and dynamics of multi-agent system.

To solve the deficiency of Central Trust Mechanism, Distributed Trust Mechanism was proposed [4,5]. The well-known PGP trust model is a typical distributed trust one, which breaks the traditional hierarchical trust architecture and need not a central authorization organization to ensure the trust management, and the trust relation among entities is constructed by the credential signature [6]. In [7], Yosi Mass and Onn Shehory define and propose a distributed trust infrastructure in multi-agents system based on credential, which need not a central credential mechanism or trusted third party; each agent can issue credential, an agent can be trusted if it can provide enough required credentials. However, in [7] only local trust can be realized and not global one. Moreover, Kagal et al. [3] proposes

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

0965-9978/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.advengsoft.2004.09.001

a secure agent system model based on distributed trust and delegation mechanism, which integrates trust management and the delegation of permission trust together so as to provide effective security protection for agent.

In those relative works about distributed trust mechanism, the trust relation is often constructed by the valid credential signatures among agents, threshold cryptography and trust delegation; an issuer is trusted when it can provide sufficient certificates from other issuers to satisfy the requester's policy [4].

Obviously, in distributed trust management system, the agents should interact to construct trust relations *each time* they want to cooperate; the communication costs between agents are very high. Therefore, though Distributed Trust Mechanism prevents single point failure, it increases the network load and delay service time. So the large communication costs in trust construction prevents the distributed trust mechanism from being applied effectively in multi-agent systems.

Moreover, autonomy is a characteristic and virtue of agent [8], so we should construct an autonomous trust management mechanism for multi-agent system. Though Distributed Trust Mechanism can also be applied to multi-agent system in some degree, but they cannot exert the autonomy of agent. The autonomous trust mechanism should make each agent have the autonomous trust management function, the trust management autonomy make the whole system more flexible. Otherwise, if some agents are damaged, other agents can still maintain their own trust managements, so the trust management autonomy can also make the whole system more robust. Therefore, we launched the project of *Autonomous Trust Management* for multi-agents [16].

Srdjan Capkun et al. presents a self-organized public-key management model for mobile ad hoc networks that allows users to generate their public–private key pairs, to issue certificates, and to perform authentication regardless of the network partitions and without any centralized services [9,10].

We refer the self-organization idea in Ref. [9] in some degree, and aiming at the characteristic of multi-agent system, propose an autonomous trust construction model based on graph theory methodology. In our model, we adopt the graph to express the trust relation; every agent can make trust organization and management autonomously; agents can construct the global trust concept by the combination of trust information among agents. The model can well be suited for the multi-agent system.

In our autonomous trust construction mechanism, if agent a want to cooperate with agent b, a can find the trust relation within its own trust information. Only while a cannot find the trust relation within itself, then they construct the trust relation by interaction. But in distributed trust mechanism, each time two agents want to cooperate, they should provide enough required credentials to satisfy the security policies. Therefore, the communication costs in our autonomous trust mechanism is lower than the ones in distributed trust construction mechanism.

The rest of the paper is organized as follows. Section 2 presents the related definition and makes a description for the question. Section 3 addresses the detail of autonomous trust construction model. Section 4 makes simulation experiments. Then the paper concludes in Section 5.

2. Relative definitions and question description

In Ref. [9], Srdjan Capkun et al. presents the concept of certificate repository and certificate graph, referring to which, we present the concept of trust graph.

In multi-agent system, agents cooperate to perform tasks. Therefore, trust relation should be constructed among agents.

Definition 1. *Agents Trust Relation Graph (ATRG): ATRG is a directed graph, which denotes the trust relation among agents. ATRG=(V,E), where:*

V denotes the agents;
$E=V\times V$ denotes the trust relation among agents;
$\langle v_1,v_2\rangle$ denotes that agent v_1 trusts v_2.

Definition 2. *Agents Trust Sub-Graph (ATSG): $ATSG_i$ is also a directed graph, which denotes the trust information contained in agent i. $ATSG_i=(V',E')$, where:*

$V'\subseteq V$, which denotes the agents that have trust relation with agent i;
$E'=V'\times V'$ denotes the trust relation among the agents of V', and $\langle v'_1,v'_2\rangle$ denotes that agent v'_1 trusts v'_2.

Definition 3. *Agent Trust Path (ATP): the trust path from agent i to j can be defined as an agent sequence $\{v_i,v_{i+1},\ldots,v_n,\ldots,v_j\}$, where: v_i trusts v_{i+1}, v_{i+1} trusts $v_{i+2},\ldots,v_{j-1}$ trusts v_j. The ATP denotes that agent i can get the trust relation with j after a series of trust delegation.*

Each agent stores its own *ATSG*within itself. In the initial phrase of the system, each agent's *ATSG* only holds the trust relations that it trusts or is trusted directly.

ATRG describes the global trust information of multi-agent system. In our model, in fact there is not *ATRG* in the real multi-agent system, each agent only store the trust information related to itself. Only after continuous interactions and combinations among agents, the concept of global trust can be achieved, and in real system such information does not exist really. Agents combine their trust sub-graphs by interaction, so the global trust concept can be achieved step by step. If agent i want to cooperate with j, i should find a trust path to j from the combinated graph of their *ATSGs*.

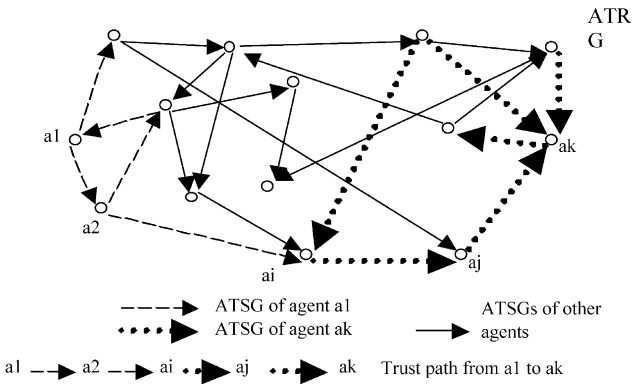


Fig. 1. Illustration of the concept of ATRG, ATSG and ATP.

Fig. 1 illustrates the relation of above concepts.

Therefore, the question of this paper can be defined as:

- each agent stores and manages its respective trust information (*ATSG*) autonomously;
- in the initial phrase of the multi-agent system, the *ATSG* in each agent is very simple, each agent only holds the trust relations that it trusts or is trusted directly;
- agents interact periodically and combine their *ATSGs*, therefore, the global trust information can be achieved gradually;
- if two agents want to construct trust relation, they should find a trust path from their *ATSGs* after combination;
- if there are not any paths between the two agents, then the two agents can construct trust relation by automated trust negotiation.
- if one node fails after the global relation has set up, the agents on this node fail too. However, the agents on other nodes can keep operation, and can construct the new global trust relations gradually by subsequent interactions.

3. Autonomous trust management model

3.1. Combination of agent trust sub-graph

As said above, in the initial phase of the multi-agent system, the trust information in each agent is very simple, which are only trust relations that it trusts or is trusted directly. To achieve the global trust concept, agents need to combine their *ATSGs*.

The combination of *ATSGs* includes two kinds of situations: one is that each agent broadcast its trust information to the agents on neighboring nodes periodically, the broadcasting trust information contains the *ATSG*. The other is that agents exchange *ATSG* while they cooperates.

Each agent has a timer, the agent broadcasts trust information periodically (we can call the interval time as exchange period) to the agent on the neighboring hosts.

If two agents want to cooperate, they should construct their trust relation at first. At the same time, we can utilize the agent cooperation to exchange trust information for combination of *ATSGs*.

In the combination of *ATSGs*, each agent only absorbs the trust information that it has not and is relative to it. In this way, the global agents trust information can be achieved gradually.

Fig. 2 is an example, in which we can see that the combination of *ATSGs* of a_1 and a_2.

The *ATSG* can be memorized as adjacency list which contains two kinds of nodes: *headnode* that denotes the agent, *edgenode* that denotes the one trusted by the agent of the corresponding *headnode*.

By using the C semantics structure [15], the formal description of the data structure of *ATSG* is as follows.

```
typedef struct node
   {agenttype trusting_agent;
      struct node *trust;
   } edgenode;
typedef struct
   {agenttype agent;
      edgenode *trust;
   } headnode;
headnode ATSG[n];
```

Therefore, the *ATSG* in a_1 after combination in Fig. 2 is shown as Fig. 3.

The combination algorithm can be seen in Algorithm 1.

Algorithm 1

```
Void Combination (headnode ATSGi[M],ATSGj[N])
/*Comine ATSGj into ATSGi*/
{
headnode ATSGi[m], ATSGj[n];
edgenode *temp, *point1;
edgenode *newedgenode;
```

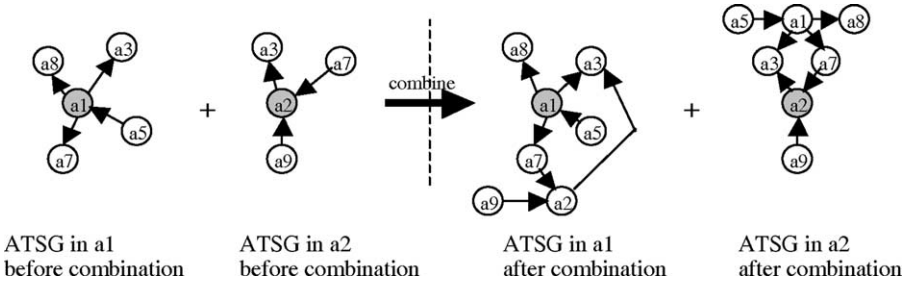


Fig. 2. Combination of ATSGs.

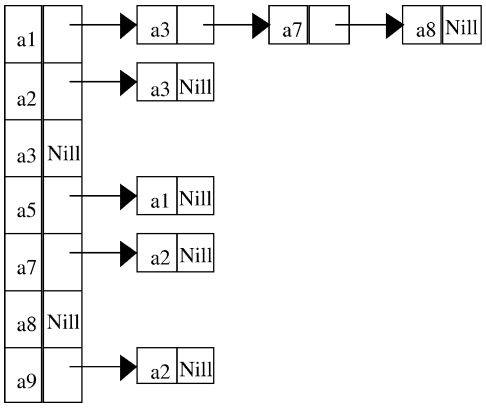


Fig. 3. Memory structure of ATSG in a_1.

```
headnode *newheadnode;
int k,b;
for (int i=0;i<n;i++)
   {
   /*Combine the agents trusted by ATSGi into ATSGi*/
   for (int j=0;j<m;j++)
   {if ATSGi[j].agent==ATSGj[i].agent
      {temp=ATSGj[i].trust;
      point1=ATSGi[j].trust;
      while temp !=nill
      {b=0;
      while (point1.trust != nill) && (b==0)
      {if point1.trusting_agent=temp.trusting_agent
         b==1;
      point1=point1.trust;
      }
      if b==0
         {new newedgenode;/*create a new edgenode*/
         newedgenode.trusting_agent=
         temp.trusting_agent;
         newedgenode.trust=ATSGi[i].trust;
         ATSGi[i].trust=newedgenode;
         }
      point1=ATSGi[j].trust;
      temp=temp.trust;
      }
   }
   }
   /*Combine the agents that trust ATSGi into ATSGi*/
   temp=ATSGj[i].trust;
   while temp!= nill
   {for (j=0;j<m;j++)
      {if temp.trusting_agent==ATSGi[j].agent
         {m++;new newheadnode;/*create a new
         headnode*/
         newheadnode.agent=temp.trusting_agent;
         newheadnode.trust=ATSGj[i].trust;
         ATSGi[m]=newheadnode;
         temp=nill;
         j=m;
         }
      };
      temp=temp.trust;
   }}}
```

From Algorithm 1, we can see that the time complexity degree is $O(n_2m_2)$, so the algorithm can only perform well when the number of agents is few.

3.2. Construction of trust relation

If agent i wants to cooperate with j, firstly it should construct a trust relation to j. The construction of trust relation has two kinds: one is that constructing relation by trust delegation among agents, i.e. looking for a trust path from i to j; other is that if there are not any paths between the two agents, then the two agents can construct trust relation by automated trust negotiation.

3.2.1. Searching for trust path

Trust path is a structure based on trust delegation networks [11], which describes the trust relation delegated by a series of agents.

If agent i want to cooperate with j, i should firstly combine its ATSG with j's. Then i should search for a trust path to j in the new ATSG. Fig. 4 shows a trust path searching process from a_1 to a_9.

The algorithm that search for trust path can be seen in Algorithm 2.

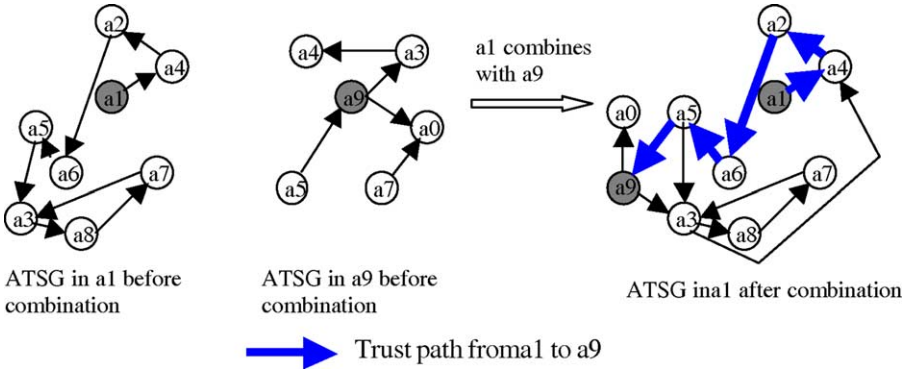


Fig. 4. An example of searching for trust path.

Algorithm 2

```
int TrustPath_Searching (agenttype ai,aj; headnode ATSGi[m],ATSGj[n])
/*for simplicity, next we denote the headnode or edgenode that contrains agent a as
node(a), and describe the data struct both of headnode and edgenode as node*/
{node *temp;
   int b=0;
   stack s; /*define a variable of stack data structure*/
   combination (ATSGi, ATSGj);
   push (s,node(a));
   while (!empty(s) and (b==0))
   {temp=pop(s);
   if temp==node(aj)
      B=1;
   while temp!=nill
   {temp=temp.trust;
      push(s,temp);}
}
return (b);}
```

If agent i cannot find a trust path to agent j, then it concludes that it cannot get the trust relation to j by trust delegation mechanism, so it may construct trust relation by automated trust negotiation, shown as Section 3.2.2.

3.2.2. Automated negotiation of trust

If agent i cannot find a trust path to j, then it can make automated negotiation with j about the trust relation. In fact, the automated negotiation of trust is one kind of distributed trust mechanism. However, in distributed trust mechanism, each time two agents want to cooperate, they should make negotiation to provide enough required credentials to provide enough required credentials to satisfy the security policy.

Automated trust negotiation mainly manages the exchange of credentials between strangers for the purpose of property-based authentication and authorization when credentials are sensitive [13], which constructs trust relation if the credentials exchanged can satisfy the requirement of security policy. Credentials flow between i and j through a sequence of alternating credential request and disclosures, which we call a trust negotiation.

A credential is a digitally signed assertion by the credential issuer about the credential owner credentials can be made unforgeable and verifiable by using modern encryption technology: a credential is signed using the issuer's private key and verified using the issuer's public key [14]. If the exchanged credentials can satisfy the security polices of the two agents respectively, then a trust relation can be achieved.

Fig. 5 is a simplified automated trust negotiation process.

If agent i can get trust relation with j by negotiation, then this new trust relation should be added into the ATSGs both of i and j.

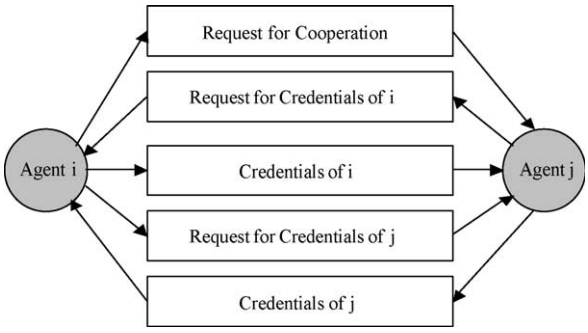


Fig. 5. Automated trust negotiation process.

3.3. Revocation of trust

Each agent can revoke a trust relation related to itself if it confronts the following two situations: one is that the total ATSG content exceeds the memory scope of the agent. Other is that the trust relation is damaged.

About the first situation, we should adopt some principles to realize the deletion of trust relations, such as FIFO (first in first out), NUR (Not used recently), LRU (Least recently Used), etc.

About the second situation, if the trust relation between i and j is damaged, we not only revoke it in the ATSG of i and j if the relation is damaged, but also revoke it in the ATSGs of all other agents.

4. Simulation experiments

For the purpose of our experiment, we have developed a minimal platform that provides the basic functions required to program agents. We have implemented a prototype which is developed with Tcl/Tk, Tclx, Tix and Binprolog [17,18]. And the prototype was also partly based on the work of Aglets Software Development Kit v2 (Open Source release) [19]. In our experiments, by software modulation, we can make the network transmission rate, network load and host CPU load change.

To test the performance of our autonomous trust management mechanism, we make three kinds of tests, shown in Sections 4.1–4.3.

4.1. Trust path searching vs. trust negotiation

Firstly, we compare the performance between trust path searching and trust negotiation. We define the performance as the ratio between the successful trust construction numbers by path search or trust negotiation, and the total number of trust construction tests. In our simulation experiment, the memory size of agent is limitless. In the simulation experiment, when trust relation should be constructed between agents, the trust negotiation method

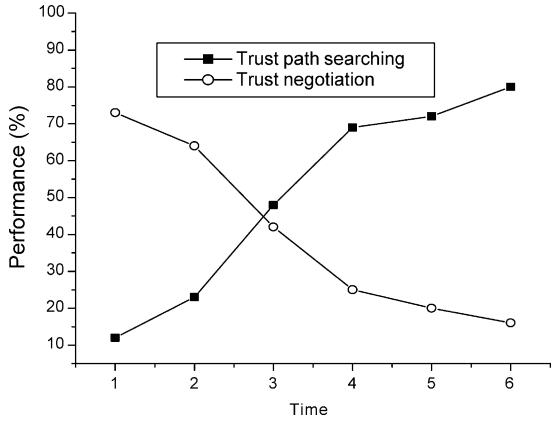


Fig. 6. Number of agents = 5.

can be used only after the trust path searching fails to find one path. The results are shown as Figs. 6–9.

From the simulation result, we can conclude that:

- at the initial phrase of system, since there is little information of ATSG in agent, so it is very difficult to find a trust path in ATSG. Therefore, the trust construction mainly adopts trust negotiation, so the performance of trust negotiation is more than the one of trust path searching;
- with the time going, agents exchange their ATSGs periodically. Therefore, the information of ATSG becomes more abundant and the performance of trust path searching becomes higher with the time goes;
- when agents' number is higher, the time cost of ATSGs combination also becomes higher. Therefore, the performance of trust path searching will become lower while the agents number becomes higher.

4.2. Trust construction among the agents on the same host vs. trust construction among the agents on different hosts

In multi-agent systems, it is important to take into consideration the cooperating agents could be located on

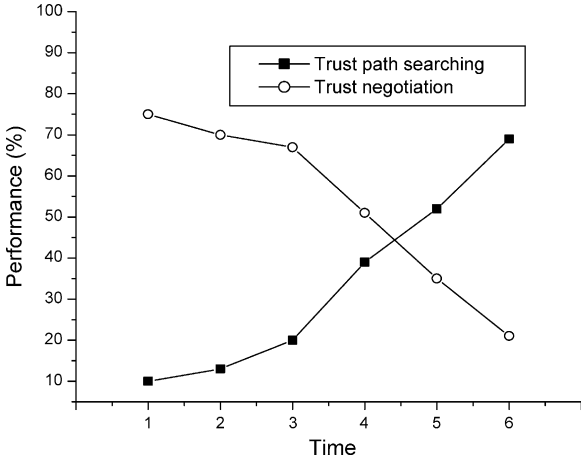


Fig. 7. Number of agents = 10.

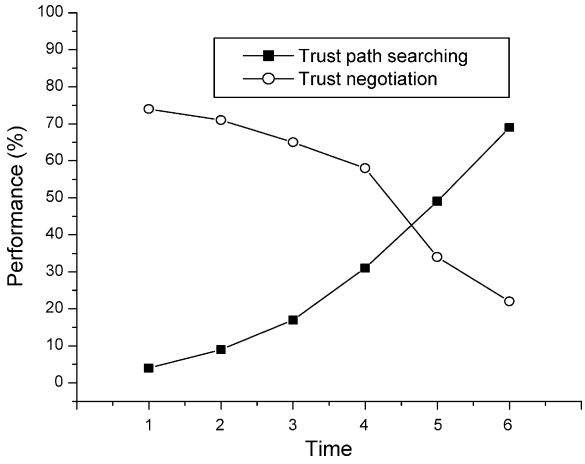


Fig. 8. Number of agents = 15.

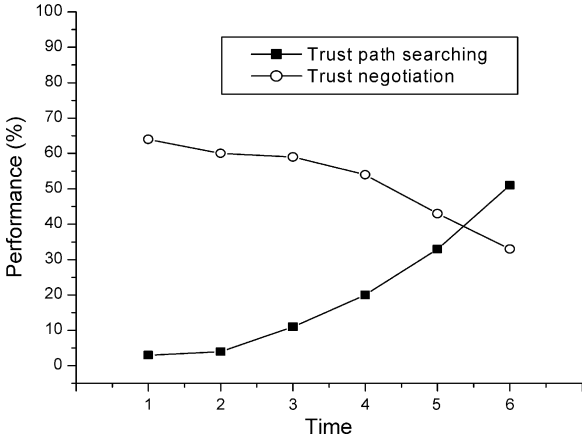
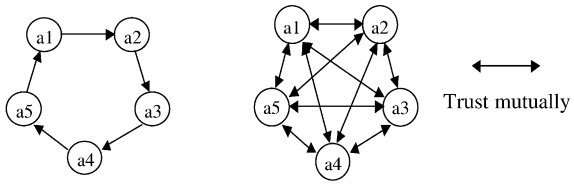


Fig. 9. Number of agents = 20.

different hosts. Obviously, the construction of more complex trust sub-graphs or trust paths, which contain agents located on different hosts, will be probably costly operations. Now we can illustrate such situation with the experiments.

In our experiments, there are five agents. The initial trust relations are shown as Fig. 10(a), where a_1 trusts a_2, a_2 trusts a_3, a_3 trusts a_4, a_4 trusts a_5, and a_5 trusts a_1; the final trust relations are shown as Fig. 10(b), where all agents trust mutually. In our experiments, we will construct the final trust relations from the initial ones.

For testing how the agent locations influence the trust construction, we can set the agents to locate on the same



(a) The initial trust relations (b) The final trust relations

Fig. 10. Trust relations in the experiments.

Table 1
Agent locations

No.	Host1	Host2	Host3	Host4	Host5
1	$a_1a_2a_3a_4a_5$				
2	$a_1a_2a_3$	a_4a_5			
3	a_1a_2	a_3a_4	a_5		
4	a_1a_2	a_3	a_4	a_5	
5	a_1	a_2	a_3	a_4	a_5

hosts or some different hosts, shown as the five situations in Table 1. Now we make five tests separately according to Table 1 for constructing trust relations of Fig. 10.

In our experiments, the five hosts are fully connected, and agent a_i only interacts with $a_{(i+1)mod5}$ and $a_{(i+4)mod5}$. The time costs for constructing the final trust relations from the initial ones are shown in Fig. 11.

From Fig. 11, we can see that the communication costs among agents located on different hosts can influence the trust relation construction effectively. The trust construction of agents located on different hosts is a costly operation. When the agents those want to construct trust relations locate on the same host, the efficiency will be high, or the efficiency will be low.

4.3. Autonomous trust mechanism vs. distributed trust mechanism

In fact, the automated trust negotiation is a kind of distributed trust mechanism. So now we can compare the efficiency of autonomous trust mechanism (includes trust path searching and automated trust negotiation, and trust negotiation takes place only after the trust path searching fails) and distributed trust mechanism (only includes automated trust negotiation).

We make six tests for agent cooperation in series; each test is on the base of the trust relations of the former test. In each test, we can adopt the autonomous trust mechanism and distributed trust mechanism. In autonomous trust mechanism, agents firstly find the trust path within its trust information if they want to cooperate,

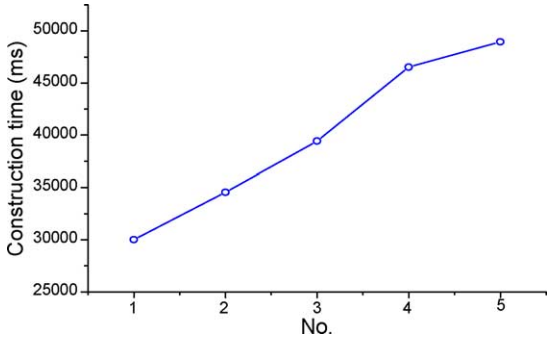


Fig. 11. Experiment results of different agent locations.

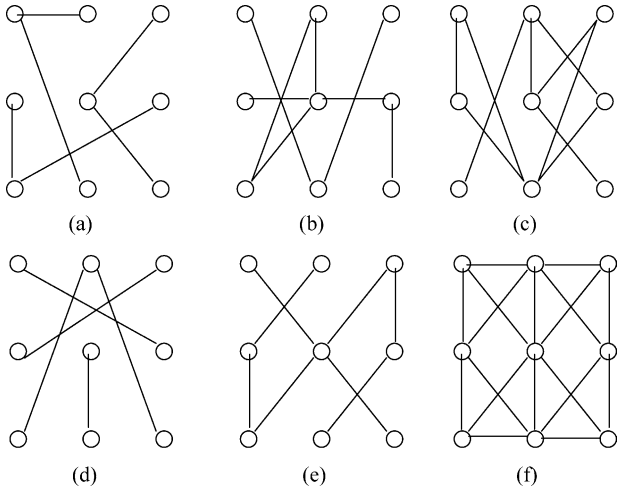


Fig. 12. Agent cooperation relations.

and the agents adopt the trust negotiation only while the trust path cannot be found. In distributed trust mechanism, agents adopt the trust negotiation each time they want to cooperate.

In this simulation experiment, we adopt nine agents, and let they locate on different hosts. The agents cooperate according to the cooperation relations in Fig. 12. If an agent wants to cooperate with another agent, it firstly constructs the trust relation with it. For simplicity, in cooperation, an agent only sends a simple message to its cooperation partner. Now we test the total time.

The results are shown as Fig. 13. From the results, we can see that the agent cooperation that adopts distributed trust mechanism is more time-consuming than the one that adopt autonomous trust mechanism. Therefore, the distributed trust mechanism costs more communication time than the autonomous trust mechanism.

From Fig. 13, we can also see that: as the time goes, agents contain more and more trust information within itself. So the more probably agents can find the trust path within itself. Therefore, the performance gap between the two mechanisms becomes bigger as the time goes.

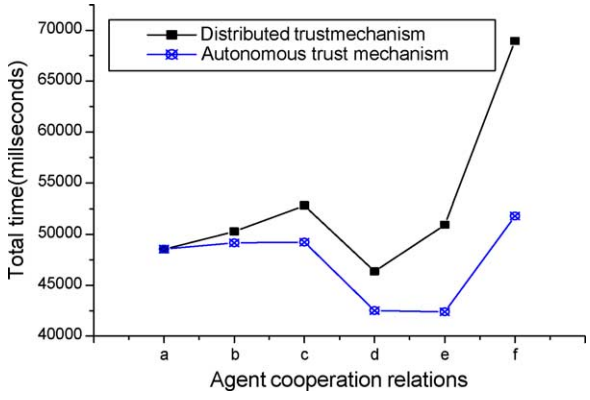


Fig. 13. Comparison between distributed trust mechanism and autonomous trust mechanism.

5. Conclusion

In this paper, aiming at the characteristic of multi-agent autonomy, we presented a autonomous trust construction model for multi-agents based on graph theory methodology. The presented model can make multi-agent manage the trust information autonomously and exchange individual trust information to achieve the global trust information. Our model can exert the autonomy of multi-agent, which is more flexible and need not any trusted authority. With the simulation experiments, we can see that our autonomous trust mechanism outperforms the distributed trust mechanism in multi-agent systems.

Our future work includes further exploration and improvement of the autonomous trust mechanism in dynamic multi-agent system and mobile agent system.

References

[1] Zhong-Zhi SHI. Intelligent agent and application. Beijing: Science Press; 2000 p. 9–11 [in Chinese].
[2] Hu Y-J. Some thoughts on agent trust and delegartion. Proceeding of fifth international conference on autonomous agents (AGENTS'01), Montreal, Quebec, Canada 2001.
[3] Kagal L, Finin T, Joshi A. Developing secure agent systems using delegation based trust management. In Security of Mobile MultiAgent Systems (SEMAS 02) held at Autonomous Agents and MultiAgent Systems (AAMAS 02); 2002.
[4] Blaze M, Feigenbaum J, Lacy J. Decentralized trust management. Proceedings of the 1996 IEEE symposium on security and privacy 1996.
[5] Abdul-Rahman A, Hailes S. A distributed trust model. ACM new security paradigms workshop 1997.
[6] Abdul-Rahman A. The PGP trust model. EDI-Forum; 1997. http://www.cs.ucl.ac.uk/staff/F.AbdulRahman/docs/
[7] Mass Y, Shehory O. Distributed trust in open multi-agent systems. Workshop on deception, fraud and trust in agent societies, autonomous agents 2000.
[8] Huhns MN. Trusted autonomy. IEEE Internet Comput 2002;6(3): 92–5.
[9] Capkun S, Buttyan L, Hubaux J-P. Self-organized public-key management for mobile ad hoc networks. IEEE Trans Mobile Comput 2003;2(1).
[10] Cpakun S, Hubaux J-P, Buttyan L. Mobileity helps security in ad hoc networks. Proceedings of the ACM symposium on mobile ad hoc networking and computing (MobiHOC) 2003.
[11] Aura T. On the structure of delegation networks. Licentiate's Thesis. Technical Report A48, Helsinki University of Technology, Digital Systems laboratory; December 1997.
[12] Jsang A. An algebra for assessing trust in certification chains. In: Kochmar J, editor. Proceedings of the network and distributed systems security (NDSS'99) symposium. The Internet Society; 1999.
[13] Winsborough W, Seamons K, Jones V. Automated trust negotiation, DARPA information survivability conference and exposition (DISCEX '2000) 2000.
[14] Schneier B. Applied cryptography, 2nd ed. London: Wiley; 1996.
[15] Ford W, Topp W. Data structures with C++. Englewood Cliffs, NJ: Prentice-Hall; 1996.
[16] Jiang YC, Xia ZY, Zhong YP, Zhang SY. A novel autonomous trust management model for mobile agents. Lecutre notes in Computer Science 3073:56–65, 2004.
[17] Tcl Developer Xchange: The Tcl and Tk Toolkit, Tcl /Tk Version 8.4.4. URL: http://www.tcl.tk/software/tcltk/8.4.html.
[18] Tcl Contributed Archive: Extended Tcl (TclX), Version 7.0a. URL: http://www.neosoft.com/tcl/. Neosoft Company; 2003.
[19] 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 and Information Technology, Fudan University, China. His research interests include multi-agent system, artificial intelligence and information security.

Zhengyou Xia was born in 1974. He received his MS degree in fuse technology from Nanjing University of Science and 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, multi-agent and active network.

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

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