An Access Control Mechanism based on Permission Delegation in P2P Network

ZHANG Changyou
Computer Dept., Shijiazhuang Railway Institute, Shijiazhuang, China; School of Computer Science & Technology, Beijing Institute of Technology, Beijing, China
zhangchangyou@bit.edu.cn

LIU Renfen
Sifang Institute, Shijiazhuang Railway Institute, Shijiazhuang, China
liurenfen_2000@163.com

CAO Yuanda
School of Computer Science & Technology, Beijing Institute of Technology, Beijing, China
caoyd@bit.edu.cn

LI Yanhua
Shijiazhuang Railway Transport School, Shijiazhuang, China
lyh_dszx@126.com

CUI Liang
Shijiazhuang Railway Institute, Shijiazhuang
ahporro@163.com

Abstract

P2P(Peer-to-Peer) is a popular model in distributed computing. We present an access control mechanism based on permission delegation in this paper. This mechanism consists of three protocols, i.e. agency discovering protocol, permission delegating protocol and resource access protocol. Firstly, the task initiator decomposes the task into subtasks and chooses other peers in high trust degree with satisfied abilities to accomplish these subtasks. We call these neighbors as task agents. Then task initiator temporarily transfers some necessary permission to subtask agents by means of credit certificate and delegation certificate. Finally, the subtask agents consume resources of resource peers followed access protocol. These protocols are analyzed in Colored Petri-Net, and simulated with CPN Tools.

1. Introduction

The computing ability of PC arrests people's attention. People hope to take full advantage of the edge ability in all the nodes. After the publication of Napster in 1999, many P2P (Peer-to-Peer) platforms, such as Gnutella, Freenet, BitTorrent, KaZaA, Skype, appear in succession. Security becomes a very important research field accompanied by the development of P2P computing.

Ian Forster[1] gave a use-case for task delegation in Grid Computing environment. He designed a permission delegation protocol based on identity certificate and credit token. Nagaratnam[2] raised a practical delegation for secure distributed object environments. Trust management is the foundation of permission delegation. In PKI[3], the third party CA issues public-key certificates to bind the holder's ID information and his/her public-key. PKI is a central model, so, it does not work well with unstructured P2P network. Reputation system[4] stores peer's trust value in a variable which is lager than 0 and less than 1, but it is difficult to make an enough fine access control policy just based on a trust value. EigenTrust[5] present a distributed and secure method to compute global trust values based on power iteration. Paper

[6] identified three vulnerabilities that were detrimental to decentralized reputation management and proposed a safeguard framework for providing a highly dependable and efficient reputation system. Damiani[7] designed a reputation-based approach for choosing reliable resources in peer-to-peer networks.

The rest of this paper is organized as follows: Section 2 describes the problems will be solved. Section 3 introduces permission delegation mechanisms in detail. Section 4 states how to calculate the credit value. Section 5 models and analyzes our mechanism in Colored Petri-Net. Section 6 simulates it based on CPN Tools. Section 7 comes to a conclusion and states our future aims.

2. Problem descriptions

A use-case based on P2P computing system is shown in Figure 1. Task is shortened as *T*. PEER_T is the initiator of the task *T*. For more efficiency, PEER_T manages to discover some partners to cooperate with each other in fulfilling task *T*. These partners are called *task agents*, namely PEER_A. Resource *R* is prerequisite for accomplishing *T* and *R* is hosted by PEER_R. In some instance, PEER_A can not access to *R* which PEER_T do. So, PEER_T has to deliver some access powers to PEER_A.

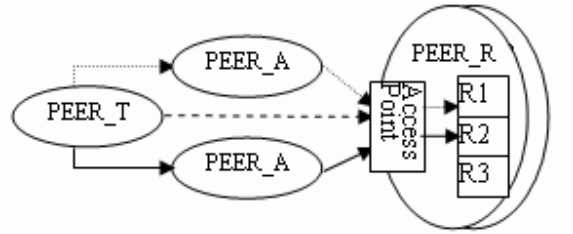


Figure 1. Use Case of Permission Delegating

There are 5 steps in the fulfilling of *T*: (1) From all the neighbors, PEER_T selects several peers act as PEER_As. (2) PEER_T makes sure that PEER_A has the ability to access PEER_R, otherwise, PEER_T manage to power PEER_A to obtain the permission. (3) PEER_T transfers his permission to PEER_A. (4) PEER_A logs on PEER_R and accesses *R*, then accomplishes the subtasks. (5) PEER_As report the results and finishes the delegation.

3. Permission delegation mechanisms

In this section, we give some definitions and introduce permission delegation mechanisms in detail. These protocols resolve the issues as follow: (1) How to select appropriate peers act as PEER_A? (2)How does the PEER_T upgrade PEER_A to access PEER_R? (3) How does the powers of accessing R transfer from PEER_T to PEER_A?

3.1. Related definitions

Definition 1: Permission Delegation Certificate (*Cd*) . *Cd* is a certificate which PEER_T issues for PEER_A to transfer some powers. When PEER_A accesses *R* to accomplish *T*, it need show its *Cd*. The data structure lists as follows,

Cd = Sign{ CertID, IssuerID, SubjectID, TaskID, PowerList, Date, Validity, Constrains }

In above formula, *CertID* is the serial number of this certificate. *IssuerID* is the number of PEER_T who issue this certificate. *SubjectID* is the number of PEER_A who hold this certificate. *TaskID* is the identifier of the task to which the certificate related. *PowerList* list the permissions that will be delegated. The certificate is issued on the *Date*. *Sign* is a signature on the whole certificate created by PEER_T with the private key. *Validity* refer to a period in which the certificate can be used, and its data structure is defined as *[NotValidBefore, NotValidAfter]*. *Constrains* is a choice list to restrict the illegal usage of the certificate. These constrains often include task-associated, provisionality, delegation steps, reusing, resource associated.

Definition 2: Credit Certificate (*Ct*) . After the communication between two peers, the initiator issues a *Ct* to the other one.

Ct = Sign{ CertID, IssuerID, SubjectID, TrustValue, successCount, unsuccessCount, Date, Validity }

CertID is the serial number of this certificate. *IssuerID* is the number of the valuator. *SubjectID* is the number of the evaluation acceptor. *TrustValue* express the measurement of the credit. *successCount/unsuccessCount* is the times of successful/unsuccessful communication. The other field has the same meaning as which in *Cd.*

Definition 3: Access Ability to PEER_R(*$ability_{RA}$*). This ability is a function of the trust value in *Ct* which PEER_R issued to PEER_A, $ability_{RA} = f\ (Ct.TrustValue)$. This ability can not assure the PEER_A has the permission to use the resource in PEER_R.

Definition 4: Access Permission to *R* (*P*). It is a permission set which consists of two parts, (1) P_0, the permission that PEER_A possessed without delegation. (2) P_{Cd}, the permission that PEER_A obtained from PEER_T by means of *Cd*. So,

$$P = P_0 \cup P_{Cd}.$$

Definition 5: Set of Credit Certificate (*Certs*). It is a collection of the total valid *Ct* accepted by this node.

3.2. Agent discovery protocol

The task initiator, PEER_T (shortening as PT in following formula) selects a group of peers matching the described abilities as agent peers, PEER_A(shortening as PA in following formula) , through agent discovery protocol.

(1) Delegation request: $PT \xrightarrow{DeleReq(TkID, TkPro, Certs_{PT})} PA$

PEER_T select *n* peers randomly as target-peers. PEER_T sends delegation request, *DeleReq (TkID,TkPro, $Certs_{PT}$)* to these targets. In the request, *TkID* is the ID of the task, *TkPro* is the profile of the task, $Certs_{PT}$ is the credit certificate set of PEER_T.

(2) Check credits: $PA.check(Certs_{PT}, SecPolicy, delePolicy)$

When each target-peer receives the *DeleReq,* it checks the validity of the $Certs_{PT}$ and extracts the credit values. Then, target-peer decides whether it will accept this delegation or not in the light of local security policy and delegation policy.

(3) Accept delegation: $PA \xrightarrow{deleResp(Certs_{PA},TkID)} PT$

The target-peers who accept the delegation request response to the PEER_T in a given period. The response includes all this peers *Cts*. $Certs_{PA}$ is the *Certs* of this target peer.

(4) Choice agent: $PT.choice(Certs_{PA})$

When PEER_T receives the *deleResp,* it validates $Certs_{PA}$ and calculates every respondent's credit value. In these respondents, PEER_T choices the peer with highest credit value as PEER_A.

(5) Notification: $PT \xrightarrow{Notify(TkID,List_{PA})} PA$

PEER_T sends a message to notify all these respondents. This message composes two fields, *TkID* and $List_{PA}$. $List_{PA}$ is the list of PEER_As.

3.3. Permission delegation protocol

As soon as the PEER_A receives the notification of winning, it begins to search the indispensable resources according to the task *T*. In this process, PEER_A (shortening as PA in following formula) should manage to get the access permission to PEER_R (shortening as PR in following formula) and *R* if it has not the ability.

(1) Resource searching: $PA \xrightarrow{disc(List_R,Certs_{PA})} (P2P)$

PEER_A broadcasts the resource profiles and *Certs* to the P2P network in order to find matched resources through given algorithms.

(2) Resource response: $PR \xrightarrow{resp(Certs_{PR},List_{resID})} PA$

When a peer receives the resource request, it will check the local resource list. If it matches some of the profile, then calculates the credit value of PEER_A, and make a decision of accepting the request or not according to the local policy. In the case of acceptance, it sends a resource response including its *Certs* and the matched resource list.

(3) Notification: $PA \xrightarrow{notify(List_{PR})} PR$

PEER_A calculates the credit values of the peers who respond. Then, it chooses the peers with highest credit value as PEER_R according to their resource list. The notification consists of PEER_R list.

(4) Resource report: $PA \xrightarrow{request(List_{<PR,resID>})} PT$

PEER_A reports to PEER_T the PEER_R list and the associated resources.

(5) *Cd* issue: PEER_T analyses the resource list. There will be some resources that PEER_A can not access. PEER_T issues *Cd* for these PEER_Ts to enhance their abilities.

3.4. Resource access protocol

(1) Resource request: $PA \xrightarrow{request(List_{resources},Certs_{PA},Certs_{PT},Cd_{PA})} PR$

PEER_A send resource request to PEER_R.

(2) Check credit: $PR.Verify(Certs_{PA}, Certs_{PT})$

PEER_R calculates the total credit value of PEER_A based on the $Certs_{PA}$ of PEER_A and $Certs_{PT}$ of PEER_T. This algorithm will be explained in detail at next section.

(3) Check permission: $PR.Verify(Cd_{PA})$

PEER_R checks the access permission on the resources which PEER_A request.

(4) Resources obtaining notify: $PR \xrightarrow{notify(yesOrNo)} PA$

PEER_R notifies PEER_A if PEER_A is successful in the checking. In some instances, PEER_R will lock the booked resources.

(5) Finishing notify: $PA \xrightarrow{notify(finished)} PR$

PEER_A notifies PEER_R that it finished the task based on the resources.

(6) Release resources: $PR.\mathrm{Re}lease(resouseID)$

PEER_R releases the locked resources associated this task.

(7) Issue *Ct* mutually: $PR \xrightarrow{valuate(Ct_{PA})} PA$, $PA \xrightarrow{valuate(Ct_{PR})} PR$

PEER_R issues a *Ct* to PEER_A, and PEER_A issues a *Ct* to PEER_R.

4. Calculating credit value

We suppose that *A* and *B* is the two peers in one communication. The credit value consists of two sections. The first one is direct-value, and the second one is indirect-value.

4.1. Direct value

Two peers evaluate each other directly after their communication.

$$T_{AB} = \alpha^{\frac{\beta^{n_2}}{n_1}}$$

In this formula, $\alpha \in [0,1]$ is the direct increasing coefficient. $T_{AB} \in [0,1]$ is the direct value that *A* gives *B*. n_1 is the times of satisfied communication. With the increasing of n_1, T_{AB} increase also. The bigger α will result in a rapider increasing of T_{AB}. β is a positive integer. As a direct decreasing coefficient, it denotes that the decreasing is easer than the increasing of the value of T_{AB}. n_2 is the dissatisfied communication times. With the increasing of n_2, T_{AB} descends. The bigger β will result in a rapider descending of T_{AB}.

4.2. Indirect value

We suppose that *A* and *B* are two peers. *X* represents a set of peers who trust in *B* and is trusted by *A*. Every peer in *X* has ever communicated with *B* directly, but *A* has not. *A* trust *X* because of their communication experience. Now, formula below calculates the credit value from *A* to *B*, and we call this value as indirect credit value.

$$P_{AB} = \frac{1}{K}\sum_{j=1}^{K}(T_{AXj} \times T_{XjB})$$

X_j is the j^{th} peer in X. K is the number of peers in set X. T_{AX} and T_{XB} are direct credit value.

4.3. Total value

Total credit value is a powered average value of direct credit value and indirect credit value.

$$T = \eta T_{AB} + \mu P_{AB}$$

Here, $\eta + \mu = 1$.

5. Protocol analysis in Colored Petri-Net

Petri-net was put forward by German scientist C.A. Petri in 1962. A Colored Petri-Net (CPN) model is usually created as a graphical drawing. It is a graphical language for constructing models of concurrent systems and analysing their properties. A CPN model contains places (drawn as ellipses or circles), transitions (drawn as rectangular boxes) and a number of directed arcs connecting places and transitions, and finally some textual inscriptions next to the places, transitions, and arcs. CPN models are formal for its model language has a mathematical definition of its syntax and semantics. This means that they can be used to verify system properties, i.e., prove that certain desired properties are fulfilled or that certain undesired properties are guaranteed to be absent. CPN is a discrete-event modelling language combining Petri-nets and the functional programming language CPN ML which is based on Standard ML.

5.1. The analysis of permission delegation protocol

Permission delegation protocol mainly used to select agent peers. Then issue delegation certificates to them. According to this protocol, we draw Petri-nets showed as figure 2.

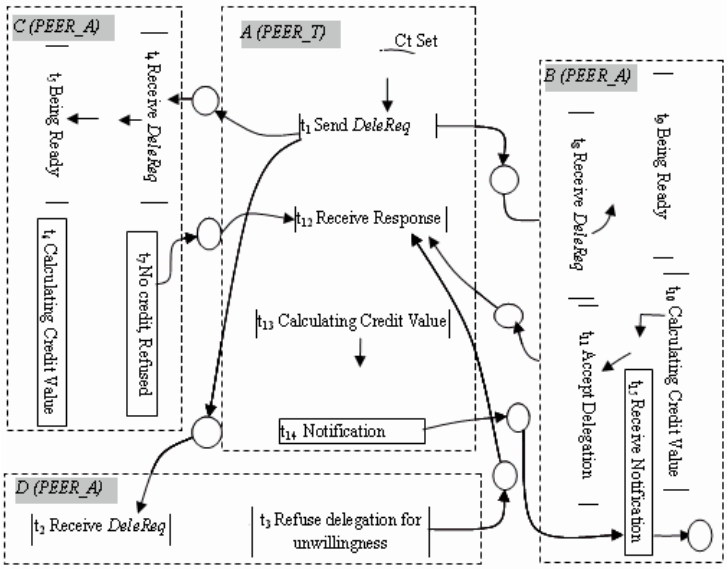


Figure 2. Petri-net of delegation protocol

In figure 2, PEER_T send delegation request to entrusted peers. There are three kinds of manners when a peer receives a delegation request. The first one is refuse delegation for unwillingness, which is shown as section *D*. The second manner is also a refuse for the verification of the credit states are not satisfied with the local secure policy, shown in section C. The third manner is shown in section B. In this case, the peer accepts the delegation request for its credit value is satisfied with the local secure policy. In the above three cases, each peer send a response to PEER_T, but only the responses in case B are accepting response. Finally, the PEER_T receives several accepting response from peers in case B. PEER_T calculates these peer's credit values, then selects the top best peers as PEER_A and notifies them.

The static properties of this model can be analyzed based on P-invariant or T-invariant. The analysis result shows that the model of Petri-net in Figure 2 is formed of concurrent components. The associated matrix *C* is defined as follows.

$$C=\begin{pmatrix}
-3 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1
\end{pmatrix} \tag{1}$$

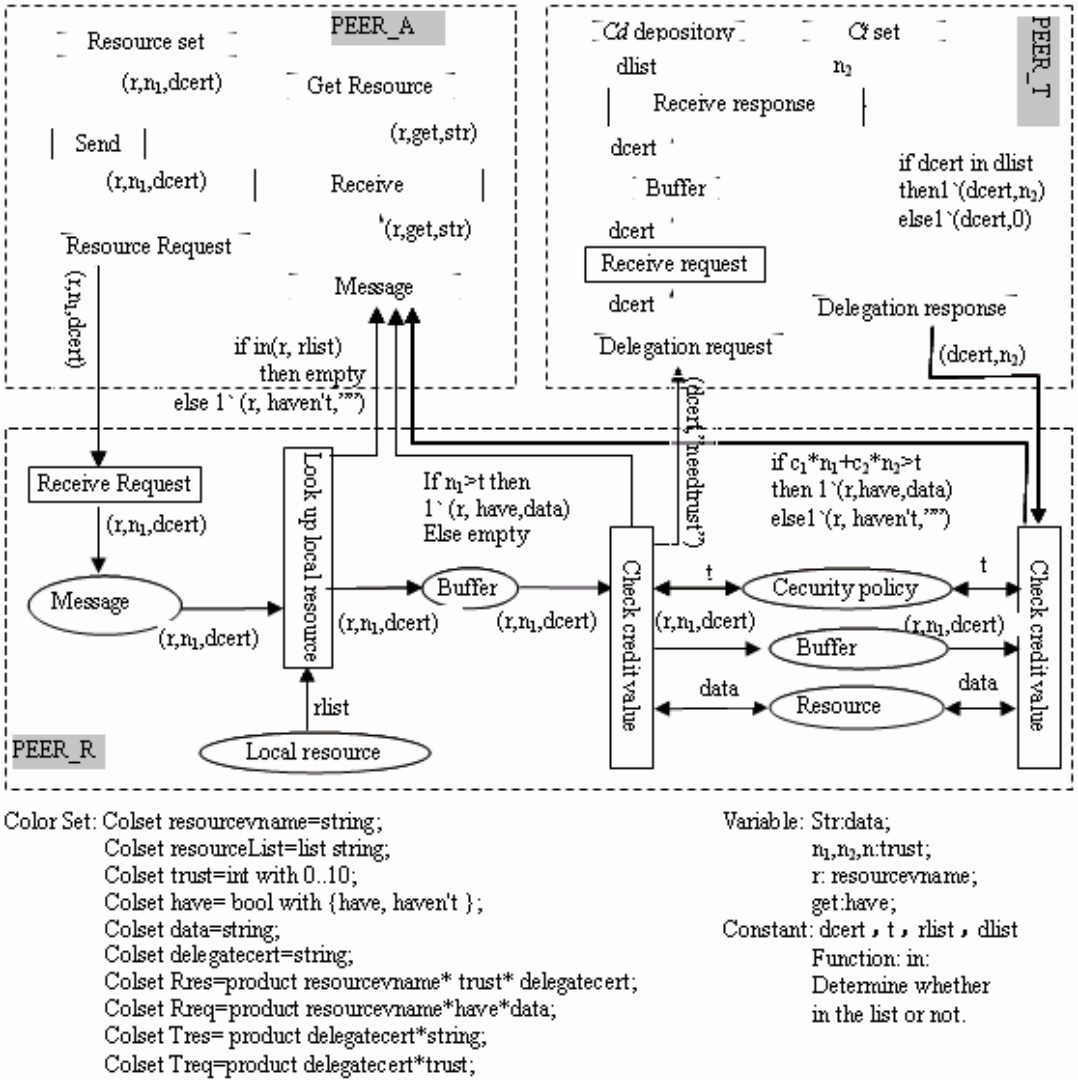
The solution of the linear homogeneous equation, $CY=0$, is

$$\begin{aligned}
Y &= c_1(1\ \ 3\ \ 0\ \ 0\ \ 3\ \ 3\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 3\ \ 3\ \ 3\ \ 3)^{\mathrm{T}} \\
&+ c_2(1\ \ 0\ \ 3\ \ 0\ \ 0\ \ 0\ \ 3\ \ 3\ \ 3\ \ 3\ \ 0\ \ 0\ \ 0\ \ 0\ \ 3\ \ 3\ \ 3\ \ 3)^{\mathrm{T}} \\
&+ c_3(1\ \ 0\ \ 0\ \ 3\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 3\ \ 3\ \ 3\ \ 3\ \ 3\ \ 3\ \ 3\ \ 3)^{\mathrm{T}}
\end{aligned} \tag{2}$$

According to the definition of P-invariant, there is a non-negative vector *Y* that satisfies the system of linear equations, $CY=0$, so *Y* is P-invariant. The delegation model satisfies P-invariant property and the place is bounded. The three manners execute concurrently for there are parallel path of the Petri-net in the above result.

5.2. The analysis of access control protocol

The Colored Petri-net of resources access control process shown in Figure 3.



Color Set: Colset resourcevname=string;
Colset resourceList=list string;
Colset trust=int with 0..10;
Colset have= bool with {have, haven't };
Colset data=string;
Colset delegatecert=string;
Colset Rres=product resourcevname* trust* delegatecert;
Colset Rreq=product resourcevname*have*data;
Colset Tres= product delegatecert*string;
Colset Treq=product delegatecert*trust;

Variable: Str:data;
n_1,n_2,n:trust;
r: resourcevname;
get:have;
Constant: dcert , t , rlist , dlist
Function: in:
Determine whether
in the list or not.

Figure 3. The Petri-net of access control protocol

Verification of system properties is supported by a set of state space methods. The basic idea underlying state spaces is to compute all reachable states and state changes of the CPN model and represent these as a directed graph where nodes represent states and arcs represent occurring events. From a constructed state space it is possible to answer a large set of verification questions concerning the behaviour of the system such as absence of deadlocks, the possibility of always being able to reach a given state, and the guaranteed delivery of a given service.

Reachability graph analysis is one of the most effective methods for Petri net models to verify the dynamic and static properties. A reachable diagram consists of all the reachable states of Petri-net, so it can analyze the reachability, safeness, boundness and liveness according to the structure characters of the graph and the associated token of each node.

The reachability graph of access control mechanism is shown in Figure 5.

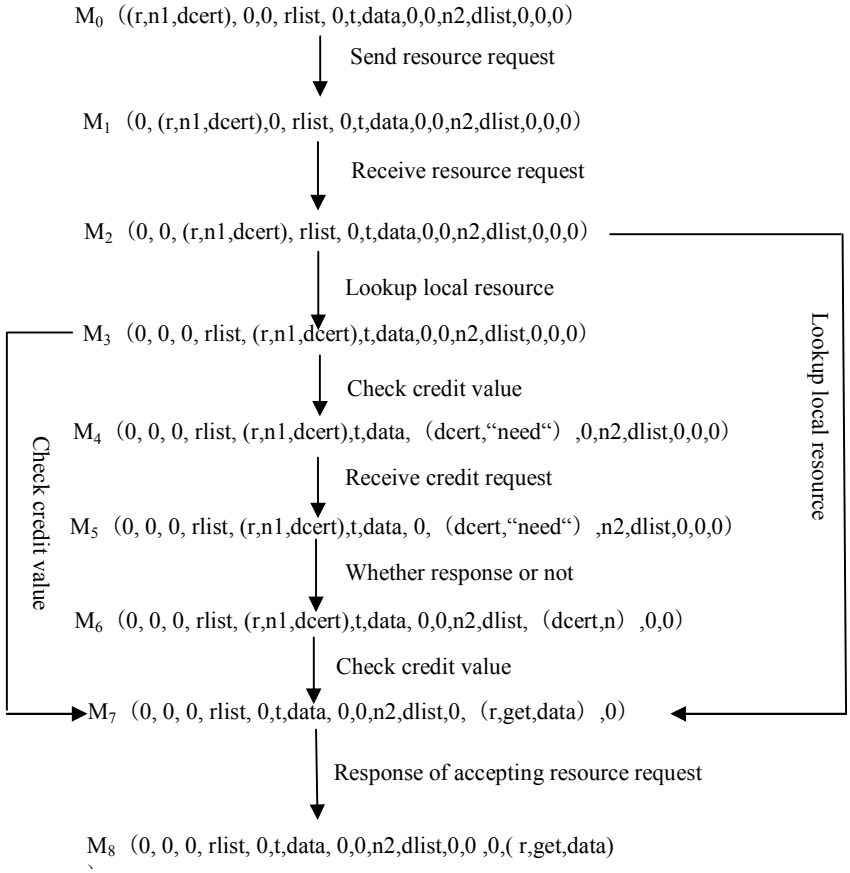


Figure 5. Reachability graph of Petri-net for access control mechanism

In Figure 5, the M_8 is the terminating end state. Petri-net reaches this state when it gets the resource access control result. This proves that no deadlock exist in this model. There are two different results. One is getting access to the resources, the other is being refused. For the latter, it will select another resources provider and continue this access control procedure.

5. Simulation based on CPN Tools

The formalization of network protocol and security protocol based on Petri-Net is very popular. Here we do not dwell on the basic conception in Colored Petri-Net (CPN) [8]. In this section, we carry out a simulation experiment by virtue of the CPN Tool[9] in order to verify the correctness of the three protocols that we designed above. By making simulations of the CPN model, it is possible to investigate different scenarios and explore the behaviours of the system. Very often, the goal of simulation is to debug and investigate the system design. CP-nets can be simulated interactively or automatically. An interactive simulation is similar to single-step debugging. It provides a way to "walk through" a CPN model, investigating different scenarios in detail and checking whether the model works as expected. User can edit almost all elements through the graphic interface and run simple program in ML language.

5.1. Task initiating peer

In this simulation, we design two PEER_As to receive the delegation request. The result is showed in Figure 6.

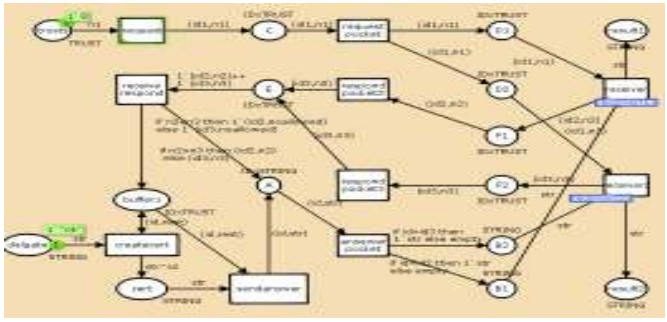


Figure 6. Petri-Net for PEER _T

In Figure 6, the green labels are the initial markings. The result of the simulation shows that the peer with higher credit value obtains the delegation certificate, and the other fails. In the whole running process, there is no deadlock, and every state is reachable. The capacity of each place is 1, so called 1-safety.

5.2. Agent peers

Figure 7 shows the Petri-Net for PEER_A accepting delegation. Figure 8 shows the Petri-Net for PEER_A the process of implementing the subtask.

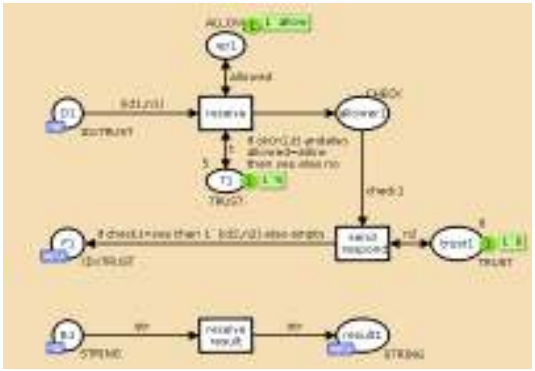


Figure 7. PEER _A accepting delegation

Figure 8. PEER _A implement subtask

5.3. Resource provider peer

Figure 9 simulates the case in which PEER_R read one of the resources, *"book"*. If PEER_R meet the requirement of security policy, return the resource information to PEER_A, otherwise, PEER_R calculates the total credit value by means of credit transferring and permission delegation certificates. The result shows that this protocol is reachable, deadlock-free and finite in place capacity. In fact, the resource access control process is a sub-net drawn in the sub-page of the hierarchical nets.

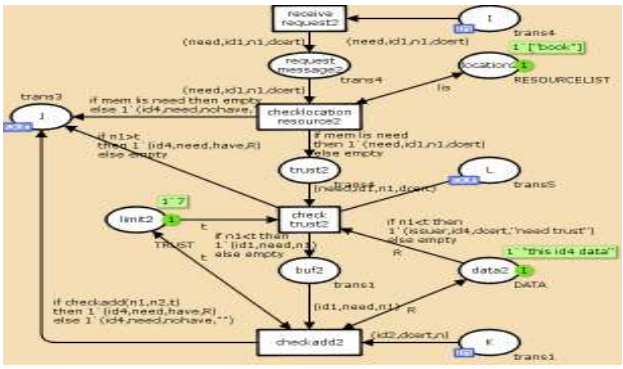


Figure 9. Top page in Petri-net for PEER_R

6. Conclusions

Considering the dynamic cooperation in P2P networks, we designed a dynamic permission delegating mechanism that consists of agency discovering protocol, permission delegating protocol and resource access protocol. We modeled the protocols in Colored Petri-Net, and simulated the result in CPN Tool to verify the correctness. The permission delegating is popular in distribute systems. Our protocols suit for unstructured P2P network. While, this mechanism is broad-brush, especially in the transferring constrains. In addition, we consider that the PEER_T should be punished for the dishonest PEER_A since PEER_A works for PEER_T under delegation. These problems are our future aims.

Acknowledgements

This work was supported by the Natural Science Foundation of China (No: 60563002)

References

[1]Ian Foster, Carl Kesselman, Gene Tsudik, Steven Tuecke, "A Security Architecture for Computational Grids", Proceedings of the 5th ACM conference on computer and communications security, San Francisco, California, United States, Nov. 1998, pp. 83-92.

[2]Nagaratnam N, Lea D., "Practical delegation for secure distributed object environments", Distributed Systems Engineering, 1998, 5(4), pp. 168-178.

[3]W.Ford, D.Solo., "RFC3280 - Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile". 2002.4.

[4]Paul Rcsnick, Richard Zeckhauser, Eric Friedman, etal., "Reputation Systems", Communications of the ACM, 2000, 43(12), pp. 45-48.

[5]S. D. Kamvar, M. T. Schlosser and H. Garcia-Molina, "The Eigentrust algorithm for reputation management in P2Pnetworks", Proceedings of the twelfth international conference on World Wide Web, Budapest, Hungary, 2003, pp. 640-651.

[6]M. Srivatsa, L.Xiong and L.Liu, "TrustGuard: countering vulnerabilities in reputation management for decentralized overlay networks", Proceedings of the 14th international conference on world wide web, May. 2005, Chiba, Japan, pp. 422-431.

[7]E. Damiani, S. Paraboschi, P.Samarati and F. Violante, "A Reputation-based approach for choosing reliable resources in peer-to-peer networks", Proceedings of the 9th ACM conference on computer and communications security, Nov. 2002, Washington, DC, USA, pp. 207-216.

Authors

ZHANG Changyou received the B.S. degree from Shijiazhuang Railway Institute, China, in 1993, and the M.S. degree from Southwest Jiaotong University, China, in 2000. Now, he is working for Shijiazhuang Railway Institute as a vice professor, and earning the Ph.D. degree in Beijing Institute of Technology, Beijing, China.

His research interests include computer networks and information security, large-scale distributed systems, and the related algorithms and applications.

LIU Renfen received the master's degree from Shijiazhuang Railway Institute, Shijiazhuang, China, in 2007.

Her research interests include unstructured peer-to-peer overlay networks and the security of networks.

CAO Yuanda graduated from Beijing Institute of Technology, Beijing, China, in 1969 and then worked as a professor there. Now he is an supervisor for PH.D degree. He has been made a co-research on computer networks in Australia since 1988 to 1989.

His research interests include computer networks and information security, large-scale distributed systems, artificial intelligent, and the related algorithms and applications.

LI Yanhua received the B.S. degree from Southwest Jiaotong University, Chengdu, China,in 1996, and the M.S. degree from the Shijiazhuang Railway Institute, Shijiazhuang, China, in 2007, she works in Shijiazhuang Railway Transport School as a teacher since 1996.

She interests on security of network and the computer teaching.

CUI Liang received the B.S. degree from Shijiazhuang Railway Institute, China, in 2001, and the M.S. degree from Beijing Institute of Technology, Beijing, China, in 2006.

His research interests include computer networks, information security and the related algorithms and applications.