Execution Environment of Peer-to-peer Services in a Mobile Environment

Tadashige Iwao[1], Makoto Okada[1], Kazuya Kawashima[2], Satoko Matsumura[2], Hajime Kanda[2], Susumu Sakamoto[2], Tatsuya Kainuma[2], and Makoto Amamiya[3]

[1] Service Management Laboratory, Fujitsu Laboratories Ltd., 4-1-1 Kamikodanaka, Nakahara-ku, 211-8588 Kawasaki, Japan
{iwao, okadamkt}@flab.fujitsu.co.jp
[2] 2nd Development Division, Fujitsu Prime Soft Technologies Ltd., 1-16-38 Aoi, Higashi-ku, 461-0004 Nagoya, Japan
{kawashima, matsumura, h-kanda, susumu, kainuma}@pst.fujitsu.com
[3] Graduate School of Information Science and Electrical Engineering, Kyushu University, 6-1 Kasuga-Koen, Kasuga, 816 Fukuoka, Japan
amamiya@is.kyushu-u.ac.jp

Abstract. This paper describes a execution environment for peer-to-peer services in a mobile environment. These days, mobile devices such as PDAs and mobile telephones have the power and the capability to support a variety of services independently. In the near future, peer-to-peer services, for mobile devices in a mobile environment, will be commonplace. However, it is difficult for these devices to provide peer-to-peer services such as file sharing and the access of CPU power since the performance of these devices is not comparable to those of PCs. Suitable peer-to-peer services and their execution environment in a mobile environment are required. Hence, we propose execution environment for suitable peer-to-peer services in a mobile environment. We have performed a large-scale experiment using a practical application on our execution environment with six hundred participants. This paper also contains the results and discussions of this experiment.

1 Introduction

Recently, the performance of mobile devices, such as PDAs and mobile phones, has increased enough to provide a wide variety of services independently. In the near future, peer-to-peer services will be commonplace for mobile devices. The peer-to-peer services on mobile devices will enable people to use them in a mobile environment. However, performance of these devices is not comparable to those of PCs. In a mobile environment with low power devices, a basic service is information browsing. Therefore, one possible peer-to-peer service is the simple exchange of information among peers in a mobile environment. In addition, information that depends on physical locations will be of use in a mobile environment. In such a situation, the location-dependent information may

change according to users. For example, permitting only the regular members to view the flight information from airports. It is required that the information, in peer-to-peer services in a mobile environment, manages itself according to the location and users.

Current major peer-to-peer systems such as Gnutella[1], Napster[2], and SETI@home[3] focus on PCs, and do not provide services for the mobile environment. These systems are popular on the Internet, and have thousands of users. Gnutella and Napster provide file sharing in a peer-to-peer format. Users of these systems find other peers who have the files they want. These systems focus on PCs because of the large data storage requirements. Also, SETI@home is a project that performs calculations to find signals of extra-terrestrial origin by using PCs, in which the owners of the Internet-connected PCs give consent to participate in the project. The software that performs these calculations operates as a screen saver, since the calculation places a heavy load on the PC during use. These kinds of applications need high CPU performance, and are not suitable for mobile devices such as PDAs and mobile telephones. It is difficult for mobile users to use applications which saps the performance of PDAs and mobile telephones. Thus, file sharing and accessing of computing power are not suitable for the mobile environment. Hence, a study must be conducted to determine what kinds of services are suitable for peer-to-peer services on mobile devices. We also need an execution environment for the peer-to-peer services in a mobile environment such as location and user dependent services.

We propose an execution environment for peer-to-peer services in a mobile environment, called *Virtual Private Community* (VPC)[4, 5], that makes information an agent that controls itself, and interact with other agents autonomously. The peer-to-peer services are defined as *policy packages* that consist of rules to activate agents, agent definitions called *roles*, and contents for the services. VPC is carried out by VPC platforms(VPC-Ps) that manage users' attributes, and decide which agents to activate according to the rules in the policy package, and users' attributes. VPC-Ps works on users' mobile devices. VPC allows services to provide according to location and users, and to propagate among users in a mobile environment. Large-scale experiments were conducted with practical applications using VPC for a period of two weeks. About six hundred people participated in the experiment.

Section 2 discusses suitable services of peer-to-peer services and their execution environment in a mobile environment. Section 3 describes details of VPC. Section 4 shows the practical application of a VPC prototype system to peer-to-peer services in a large-scale experiment.

2 Execution Environment for Peer-to-peer Services in a Mobile Environment

The exchange of information among peers is the most important feature in peer-to-peer services. Information is provided by, and used by, the peers in a peer-to-peer environment. There are two phases in treating the information; the

propagation, and the utilization of information. The propagation of information is performed by the transfer of information from one peer to another. In the utilization of information phase, there are two modes; a stand-alone mode and an interaction mode. A stand-alone mode involves the use information off-line, such as playing music files. This mode does not require interaction with other peers. An interaction mode involves the interaction among peers to use applications such as chat, network games, and groupware. The number of peers involved in the interaction may be small or large, and the interaction should be conducted in a peer-to-peer manner.

Also, location dependency of information is important in a mobile environment. Information also has dependence on location, time, users, context, and so on. Services are affected by information dependency, and are provided using information. There are services with information dependent on physical location since users move around in a mobile environment. For instance, one of the services is a explanation service of displays in front of users in a museum. Moreover, user dependency of information makes services exclusive. Only the users on which information depends can receive the services with the information. Thus, information dependencies affect services, and should be also considered in peer-to-peer services in a mobile environment. Execution environments for peer-to-peer services have not only to support the two phases and two modes, but also to support information dependencies.

One methodology is to make the information an agent that decides its own behavior among the peers in the propagation and utilization phases. The agents are also able to control information dependencies. The agent propagates among the peers, and authenticates users in the users' own mobile devices. The agent also interacts with other propagated agents that work with other peers. The agent may change itself according to peers by the result of authentification, and copyright information. In order to make this a reality, we need a framework for the agent.

3 Virtual Private Community (VPC)

Our framework, called *Virtual Private Community* (VPC), enables information in peer-to-peer services to act as an agent. VPC provides a mechanism that defines an agent behavior, authenticate users, and executes agents. Necessary agents for a peer-to-peer service are defined in a policy package that consists of a condition rule to decide active agents according to users, a set of agents(called *roles*), and necessary information(contents) for the service. Agents communicate with each other through communities that are created by agents who have accepted the policy packages. Services are offered by interaction among agents in communities. For example, in a music retail service, a policy package contains two agents, an authorized agent which can play the complete music file, and a trial agent which can play only part of the music file. Users who have purchased the music file can listen to the music through authorized agents.

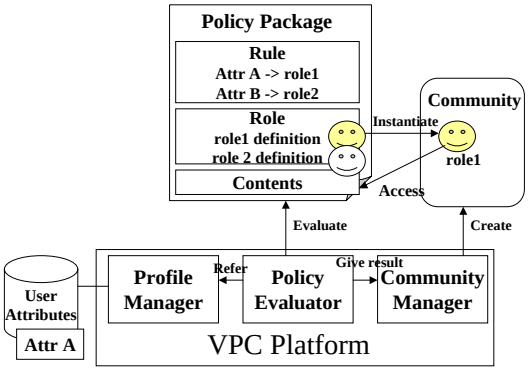


Fig. 1. Basic Model of VPC

Dependencies of information in VPC are separated into attributes of entities such as users and location, and activation rules of information (agents). Entities have their own attributes. Activation rules are described in each service (policy package). The dependencies of information are deduced by a set of attributes of entities and activation rules.

3.1 VPC Basic Model

Figure 1 shows a basic model of VPC. In the basic model, a policy package defines the agents, condition of activation for each agent, and information that is treated by the agents. Agents are activated by *VPC platforms* (VPC-Ps) that manage user attributes, evaluate policy packages according to the user attributes, decide appropriate roles, and create the community. The agents access contents in the policy packages, in which the agents are defined.

Main parts of a VPC-P are a profile manager, a policy evaluator, and community manager. A profile manager manages the user profile in anti-tamper devices such as Java Card [6]. A policy evaluator decides roles depending on the user attributes. A community manager executes selected agents, and connects with other community managers on other VPC-Ps that have the same policy packages. VPC supports both phases of propagation and utilization, and also supports both the stand-alone and interaction models in the utilization phase. In the stand-alone mode, the community manager does not connect with the others, even if others have the same community.

Figure 2 shows the overview of the communication process of VPC. VPC-Ps are able to exchange policy packages among themselves, and to communicate with each other using the policy packages. Communities are created by VPC-Ps that accept the same policy packages. VPC-Ps provide services by collaboration among themselves. VPC-Ps can join existing communities by accepting the policy packages used by the VPC-Ps in the respective communities. VPC-Ps

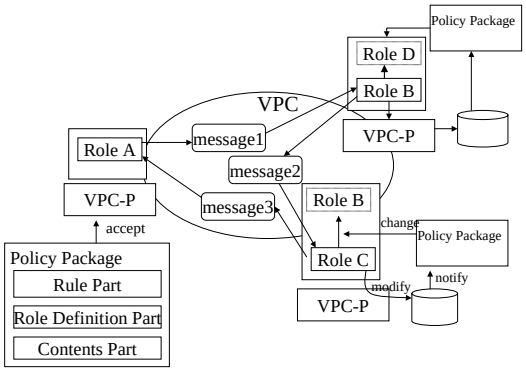


Fig. 2. VPC communication overview

analyze the policy packages, deduce their own roles according to the rules of the policy packages, and user attributes. VPC allows any role such as database access, calculation, and control of other existing systems.

A community consists of VPC-Ps that accept a particular policy package. Thus, VPC-Ps form communities by connecting to each other. Communities consist of sub-communities on each VPC-P. Entities of a community reside on distributed VPC-Ps that connect the community.

3.2 Policy Package

Figure 3 depicts a structure of policy packages. A policy package consists of a set of rules as a condition table, a set of roles as transitions, and a set of contents. A rule consists of a condition and role names. A condition is described with logical expressions using attributes. An attribute consists of a database name, a variable name, and the value of the variable. A role consists of a role name, a program name, and an initialization method. A content consists of a content name, and content path that locates the real content data. Content includes program codes that implements roles. A policy package is written in XML.

Policy packages are encoded by S/MIME [7]. S/MIME enables VPC to detect falsification of policy packages by checking hash codes of policy packages. When VPC detects falsification of policy packages, the VPC discards the policy packages.

3.3 User Profiles

A profile manager has access interfaces for roles. User profiles are stored as a variable and its value, or as digital certificates of PKI[8]. Roles and a policy evaluator are able to access data through a profile manager by specifying the variable name.

```
<policy package> ::= <rules> <roles> <contents>
<rules>          ::= <rule> | <rule> <rules>
<rule>           ::= <condition> <role names>
<role names>     ::= <role name> | <role name> <role names>
<condition>      ::= "TRUE"
                   | "and" <condition> <condition>
                   | "not" <condition>
                   | "eq" <attribute> | "<" <attribute>
<attribute>      ::= <variable name> <value>
<roles>          ::= <role> | <role> <roles>
<role>           ::= <role name> <program name> <init description>
<contents>       ::= <content> | <content> <contents>
<content>        ::= <content name> <content path>
```

Fig. 3. Structure of Policy Packages

A profile manager evaluates expressions of variables and values. A profile management part has corresponding evaluation modules for each database. It evaluates given expressions using the evaluation modules. Therefore, the profile management part allows evaluation of expressions that use types.

3.4 Evaluation of Rules in Policy Packages

A policy evaluator deduces roles by evaluating rules in policy packages according to users' attributes. A policy evaluator refers to conditions of rules in policy packages, evaluates each term of each condition with a profile manager, and decides appropriate roles for the user. Then, it compares a list of current active roles and deduced roles, installs necessary roles that are not currently active into sub-communities, and removes unnecessary roles from the sub-communities.

A policy evaluator requests evaluation of each term of conditions in a rule of a policy package to a profile manager. The profile manager accesses the specified variables in a term, and gets a value for specified variables. Then, it evaluates the term with the value, and returns TRUE or FALSE as a result. The policy evaluator combines the results, and decides the validity of the condition. The policy evaluator deduces appropriate role names by evaluating all conditions in the rule according to users' profiles.

The policy evaluator gets role programs with deduced role names from the policy package. Role programs are Java objects in VPC-Ps. The role assignment part creates instances of role programs with initialization according to init description in the policy package, and adds them into corresponding communities.

A user has a set of attributes $A = \{a_1, ..., a_n\}$. A rule part of a policy is a set of activation rule $W = \{c_1 \rightarrow r_1, ..., c_i \rightarrow r_i\}$. A activation rule consists of a pair of a condition, that is combination of attributes ($c_k = a_{k_1} \wedge ... \wedge a_{k_n}$), and a role r_k to activate. A evaluator $E(A, W) = \{r_k | c_k \rightarrow r_k \in W, c_k \vdash A\}$ is a function that determines a set of roles according to users' attributes and activation rules. A predicate $c_k \vdash A$ means $\forall a_{k_i} \in elements(c_k) \wedge a_{k_i} \in A$. A function $elements(c_k)$

returns a set of attributes used in c_k. Roles that are assigned for users are given by the evaluator E with activation rules.

3.5 Interaction among Roles in Communities

Collaboration among agents in communities is performed by message passing and the messages have to be accepted by appropriate roles in order to enable the collaboration. Our framework adopts the mechanism of collaboration among roles developed by Field Reactor Model [9] that is a coordination model [10] based on Dataflow Computing [11]. The Field Reactor Model (FRM) provides a method of flexible collaboration among agents by employing pattern matching.

VPC-P provides pattern invocation that is based on FRM. The patterns in VPC-P correspond to patterns in FRM and pattern invocation provides a method of invoking functions of roles without signatures of functions and addresses of roles. When a message of a pattern is sent into a community, VPC-P automatically invokes the appropriate functions. As well, values returned from the functions are put into the community as messages. Collaboration among roles is constructed as a chain of message passing and the roles do not require the specification of addresses of roles to invoke. VPC-P allows the user to define patterns and roles.

Role calculation is performed according to the following; a set of patterns that are used in a community C is $P = \{p_1, \ldots, p_m\}$. There are roles $R_1, \ldots, R_n$ in community $C(C = \{R_1, \ldots, R_n\})$ and a role R_k is expressed as $R_k =< I_k, f_k, O_k >$. f_k is a function that has pattern $I_k(\in P)$ as an argument and returns pattern $O_k(\in P)$. The pattern matcher $M(m)$ checks whether a message m matches the pattern of function f_k, and if there is success, then it invokes function f_k. Then the function f_k returns messages as its values with this process being described by $M(m) = \{\cup_{i=1}^{n} O_k | I_k = m\}$. Thus the computation between roles in the community C proceeds as $M(m), \forall m_n \in M(m_{n-1})$. The community C repeats this process until complete collaboration among roles is created. This method of collaboration is similar to the coordination model such as Linda [12].

Patterns of roles are object types. A VPC-P allows any Java objects as roles and messages. VPC-Ps provide type match invocation that is a method, in which methods of objects are invoked when the same type of messages as an argument of the methods are put in the communities. Return values of the methods are put back into the communities when the return values are not null. The methods of roles are merely declared as normal methods.

3.6 Peer-to-peer services with VPC

VPC makes information agents act as roles in a policy package. Authorized users that have certificates can see and use the information through assigned roles. In addition, VPC supports location dependent services; Figure 4 shows overview of location dependent services. In VPC, location dependent services are also described as policy packages. The policy packages contain roles for browsing a list of services that are provided at the location, and for download the policy

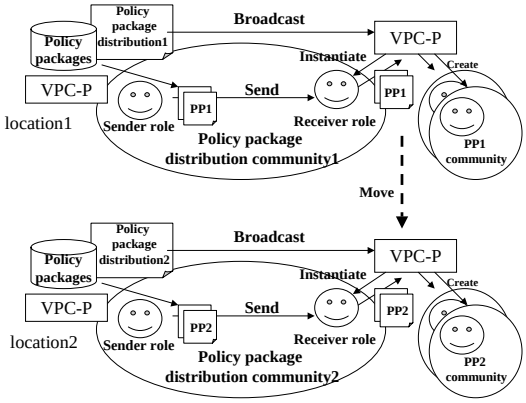


Fig. 4. Location dependent services

packages. A policy package manages several policy packages through roles. Roles to obtain policy packages act as gate keepers of each policy packages. Only users who have certificates can reach the inner part of policy packages. This mechanism allows adaptation of services not only depending on location, but also depending on time, context in a service, and so on.

VPC-Ps that have policy packages depend on the corresponding location broadcast a URL of a policy package that distributes location dependent policy packages. When users come to certain location such as a hotspot, the VPC-Ps of the users receive the URL, get the policy packages distribution package, and evaluate it. Then, a receiver role is assigned to VPC-Ps of the users, and gets corresponding policy packages to the location. By the receiver role evaluating the received policy packages, the users may see information according to the users' attributes.

4 Experiment with Practical Application

We have performed large-scale experiment on VPC in an underground shopping center in Nagoya city (Japan) for two weeks. This experiment aimed to make sure of the effectiveness of a new peer-to-peer (p2p) service and a p2p medium using VPC in a stand-alone mode.

In the experiment, each user has a PDA that is WindowsCE, and users get information, such as special bargain price information of shops around the user, depending on the user's location, through a wireless LAN. The information of the service is provided as a policy package with VPC. Users can see the information that matches the user's attributes. For example, the information in a drug store has two contents, cosmetics for ladies and shaving lotion for gentlemen. When the user is a woman, she can see the cosmetics information. We employed a bingo game as a user interface in order to make the experiment more attractive

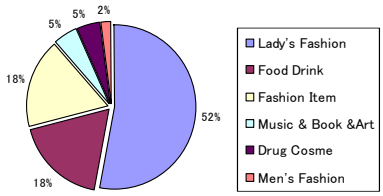


Fig. 5. Categories of shops

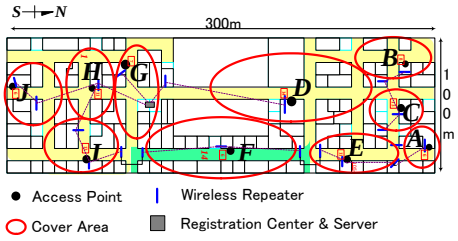


Fig. 6. Overview of the shopping center

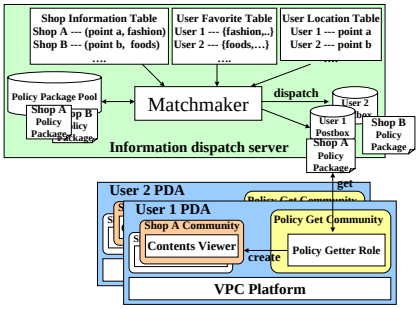


Fig. 7. System overview

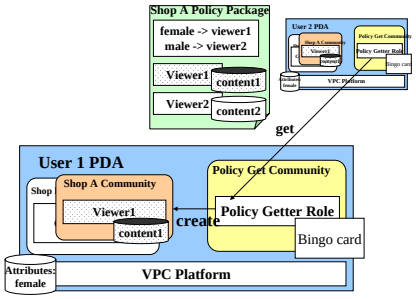


Fig. 8. Exchange information among peers

for users. A bingo card for each user is given during registration. Each set of information includes its own ID that is a bingo number. Users can exchange given information(policy packages) among them. When users win the game, they can get a coupon that is only able to be used in the shopping center.

Categories of shops in the shopping center are shown in figure 5. This experiment had been broadcasted by some TV media, FM radio stations and newspapers. About six hundred people participated in this event, thanks to the media.

4.1 System overview

Figure 6 shows an overview of the shopping center and location of wireless LAN access points. The size of the shopping center is about 300m x 100m. There are 107 shops in it. We employed 802.11b for wireless LAN. Ten access points are set in this area. Each access point is connected by wireless LAN repeaters. The number of repeaters is twenty. Each access point covers an area in which about ten shops reside, and manages MAC addresses of PDA terminals. Channels of access points are different from neighborhood in order to avoid interference. Users are registered at the registration center when they begin the game.

A system overview is shown in figure 7. The system consists of a server, which sends information to each user, and VPC that works on user's PDA terminals. We could not prepare PCs or PDAs for each access point because it is too expensive and is against the fire prevention laws in Japan. Therefore, we decided to use a server and to multiplex it. The server has policy packages of all the shops and three tables; shop information table, user favorite table, and user location table. A user location table maintains connections between PDAs and access points. A user favorite table has a set of users' favorite categories, and is created for each user during registration. The shop information table has pairs of shops' locations and the categories for each shop. The main module is a matchmaker that decides policy packages to send for each user according to matching user's location and favorite with shop information. The server has a postbox for each user inside it, and dispatches corresponding information within the area, in which users exist, to the users' postboxes. Users' VPC-Ps get delivered policy packages and evaluate them.

Figure 8 shows diagram of exchange of information among peers. A "policy get" community that has a "policy getter" role is to exchange information among peers and to get information from the server, and is activated on the PDA terminals. The role also inquires other peers by broadcasting messages periodically. When the user wants information and other peers indicate will giving the information, the two "policy getter" roles on both users' VPC-Ps exchange the information each other. The "policy getter" role also checks the user's postbox periodically. The "policy getter" role creates shop communities, when it gets new policy packages from the postbox or other peers. A "viewer" role that enables a user to see its contents (HTML) is activated. The "viewer" role is changed according to user's attributes.

4.2 Results and discussion

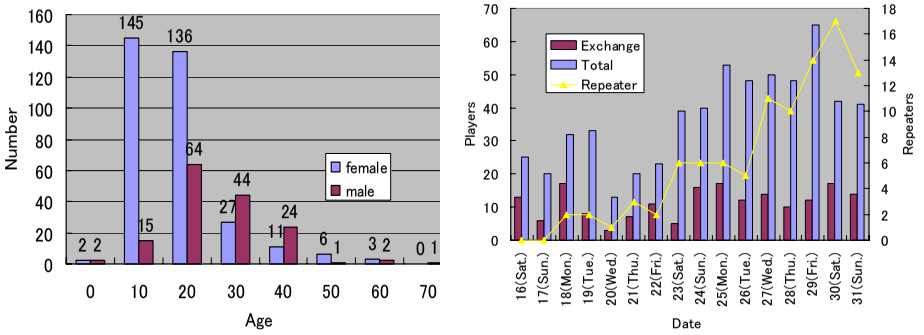
About six houndred users came to participate in the experiment. Figure 9(a) shows the numbers of persons per generation. Figure 9(b) shows the change in the numbers of players per a day. It also shows the total numbers of information exchanges, and the numbers of repeat players of the game. The increase in the number of repeat players was especially remarkable.

The followings are user comments:

- It is a lot of fun as a game.
- I am interested in the time services.
- I want to exchange information more.
- I need to exactly pinpoint information by location.

The followings are shop owners' comments:

- P2P services may be new advertisement media.
- We want to use these services as new market tool to focus on specific customers.



(a) Generation (b) Change in the numbers of players per a day

Fig. 9. Results

- PDAs are not popular for ordinary people. We require these services with mobile phones.
- Please perform this event for a year.

We have got one answer for the question that ordinary people use or require p2p services. The customers need media which they can discover goods or services they require in order to get them with efficiency. On the other hand, shop owners want to advertise goods or services to customers who are interested in. There are sufficient possiblity that the p2p services become such media. Especially, the p2p services in VPC enable users to access information according to location, time, service context, and so on.

5 Conclusion

This paper proposed execution environmnet for peer-to-peer services in a mobile environment with mobile devices such as PDAs and mobile telephones. Very restricted mobile devices are not suitable for file sharing and the renting of CPU power. In a mobile environment, information browsing is basic. Also, the information may change according to the location and users. Moreover, as the copyright of information needs to be protected, we provide a framework, called a Virtual Private Community, to make information an agent that manages access rights, and to interact with other agents autonomously. The peer-to-peer services are defined as policy packages that consist of the rules to activate agents, agent(role) definitions, and contents. VPC is carried out by VPC platforms(VPC-Ps) that manage users' attributes and decide the agents to activate according to rules in the policy package and users' attributes. We performed a large-scale experiment with a practical application using VPC about six hundred people enjoying the experiment. In the experiment, young ladies (teens-20s) came again and again, and enjoyed this peer-to-peer service with their friends. The news of the experiment spread through the grapevine for these ages. They also freuently exchanged

information among themselves. The i-mode that is web browser for mobile telephones of DoCoMo Ltd. in Japan increased explosively from people of these ages, and has thirty million users now. There is possibility that peer-to-peer services will become very popular due to the influence of people of these ages.

Acknowledgments

This work was done under a grant to the "Research on Management of Security Policies in Mutual Connection" from Telecommunications Advancement Organization (TAO) of Japan.

References

1. Fernando Bordignon and Gabriel Tolosa, Gnutella: Distributed System for Information Storage and Searching Model Description, http://www.gnutella.co.uk/library/pdf/paper_final_gnutella_english.pdf (2000)
2. Napster: http://www.napster.com. (1998)
3. By Eric Korpela, Dan Werthimer, David Anderson, Jeff Cobb, and Matt Lebofsky:SETI@Home: Massively Distributed Computing for SETI. Computing in science & engineering, January/February 2001 (Vol. 3, No. 1), pp. 78-83, http://www.computer.org/cise/articles/seti.htm, IEEE (2001)
4. T. Iwao, Y. Wada, S. Yamasaki, M.shiouchi, M. Okada, and M. Amamiya, "A Framework for the Next Generation of E-Commerce by Peer-to-Peer Contact: Virtaul Private Community", WETICE2001, pp.340-341, IEEE (2001)
5. T. Iwao, Y. Wada, S. Yamasaki, M. Shiouchi, M. Okada, and M. Amamiya, "Collaboration among Agents in Logical Network of Peer-To-Peer Services, SAINT2002, pp. 6-7, IEEE (2002)
6. Sun Microsystems, "Java Card 2.1.1 Platform", http://www.java.sun.com/products/javacard/javacard21.html,
7. B. Ramsdell, Editor, "S/MIME Version 3 Message Specification", http://www.faqs.org/rfcs/rfc2633.html, 1999
8. Stephen Kent, and Tim Polk, "Public-Key Infrastructure", http://www.ietf.org/html.charters/pkix-charter.html, 2000
9. Tadashige Iwao, Makoto Okada, Yuji Takada and Makoto Amamiya, "Flexible Multi-Agent Collaboration using Pattern Directed Message Collaboration of Field Reactor Model", LNAI 1733 pp.1-16, PRIMA'99
10. G. A. Papadopoulos and F. Arbab, "Coordination Models and Languages", Advances in Computers Vol. 46, pp 329-400, 1998
11. Amamaiya, M., Hasegawa, R.: Dataflow Computing and Eager and Lazy Evaluation, New Generation Computing, 2, pp.105-129, OHMSHA and Springer-Verlag (1984).
12. Carriero, N. and Gelernter, D," Linda in Context", Communications of the ACM Vol. 32-4, pp 444-458, 1989