

Mobile-Agents for Distributed Market Computing

Shinji Tanaka, Hirofumi Yamaki, and Toru Ishida
Department of Social Informatics, Kyoto University
Kyoto, 606-8501, JAPAN
{stanaka,yamaki,ishida}@kuis.kyoto-u.ac.jp

Abstract

This paper discusses the implementation using mobile agent and the performance of the market computing to allocate network quality of service most efficiently, based on users' preference. Though the protection of users' private preference and the efficient calculation are both important, these two requirements often contradict each other.

By implementing QoS Market, a market-based network resource allocation system, using mobile agents, the communication time between agents and an auctioneer, which is the largest overhead in market computing, is reduced dramatically without leaking users' privacy information. The result of experiments show a mobile agent approach is more efficient than others, and the overhead of one market computing cycle in this system is about 450msec, which means that the system achieves sufficient efficiency.

1. Introduction

In usual networks such as the Internet, a number of users share the network resource, which causes the users' demands frequently conflict. In such an environment, the importance of a communication link varies according to the application and its usage, but network control policy of "best-effort" type, such as CSMA/CD, fails to consider such an aspect. What we need here is a resource allocation policy that reflects users private preference and allocates limited network resource efficiently.

We have already proposed an approach which is based on a competitive price mechanism explained by *general equilibrium theory* [10]. In our model, the preference of users and the characteristics of application programs are represented by *consumer* and *producer* agents respectively. Each agent updates their bids based on the price of goods, which is determined by the pricing mechanism of the market to balance demand and supply. A Pareto optimal resource allocation is obtained as the result of the interaction [2].

Our challenge here is to calculate such an efficient resource allocation in a practical length of time without leaking users' private information about preference for applications. A market-based algorithm generally consists of a large number of iterations to derive an equilibrium that balances the demand and the supply of goods, which results in a slow response to the given environment. The situation of the network used by a number of people, however, changes dynamically according to the change in the usage of application programs. This dynamics allows the allocation system only a few seconds of delay at the most.

In previous research [9], we applied the market-based mechanism into FreeWalk [5], a multimedia desktop meeting environment with shared virtual 3D space (Figure. 6), and controlled the transmission of pictures. In this experiment, the whole allocation system is implemented in a single process in order to suppress the number of messages exchanged via network and achieved a practical quality of allocation in real-time. This approach, however, makes the users' private preference in its nature, leak to the auctioneer, which is a part of the market mechanism that modifies the price according to the demand.

To resolve this problem, this paper proposes an approach to implement a group of agents as a mobile agent that dynamically changes its place to an appropriate position. By moving agents to the server where the interaction among agents is performed, the users' privacy is preserved while keeping the smallest delay of the allocation.

We have implemented a market-based network resource allocation system with mobile agents on Windows NT, and validated the above scenario by performing several experiments.

2. Market-Based Resource Allocation

In this section, we present the general idea of market-based network resource allocation and the algorithm of the market calculation.

2.1. QoS Market Model

Our approach to achieve efficient resource allocation by merging various preferences of individual users is to introduce a market mechanism. Below, we describe the framework of our market-based application QoS control as the basis of the implementation issues discussed in this paper.

The two basic ideas to construct a market model for QoS control are as follows.

1. Users evaluate application QoS, which is the quality of service provided by network application programs, rather than the raw network resource, such as bandwidth, that they use.
2. We distinguish the “current” and the “future” networks, so that inactive users can obtain the incentive to transfer their rights to use current network resources to other active users, in exchange for the rights to use future network resources.

Figure 1 shows the market model for application QoS allocation. The rectangles in the figure represent the goods exchanged in the market. There are two types of goods, bandwidth and QoS, and each of them is divided into current and future goods. CBW and FBW stand for current and future bandwidth respectively, which are both shared by all the users.

The QoS of the communication from user j to user i is represented by q_{ij} , and $FQoS_i$ stands for the QoS that is received by user i in the future. The future QoS is assumed to be a single good for simplifying the model.

The circles in the figure are agents. Those on the left are *consumer* agents each of which represents the preference of each user, and those on the right are *producer* agents, each of which represents the conversion from bandwidth into application QoS performed by application programs.

The specifics of this model, such as the modeling of user preference and application performance, and the evaluation of resource allocation by simulation have been reported in our previous work [10].

2.2. Algorithm for Computing Equilibrium

In this market-model, there are two types of elements: agents and auctioneers. Figure 2 shows messages exchanged in the market. The underlying model of network usage is single-cast communication with network resource reservation mechanism, such as an intranet with RSVP [1] support.

A consumer agent represents user’s preference, which is modeled as a utility function that takes a bunch of goods consumed as its input and returns a higher value for more

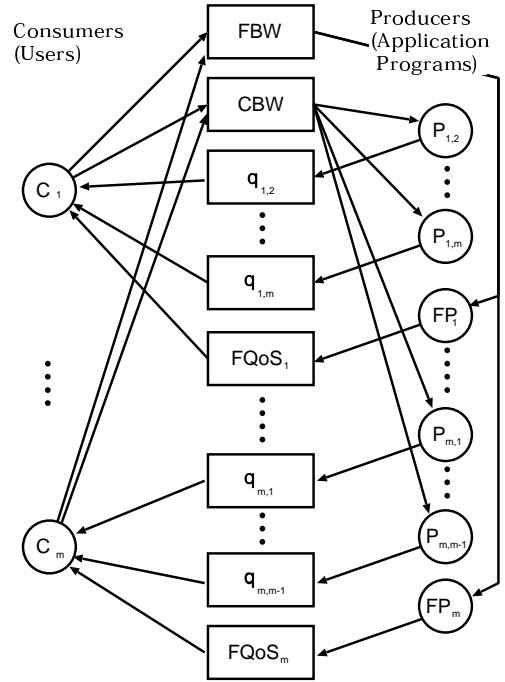


Figure 1. QoS Market Model

preferred bunch. There is a one-to-one correspondence between a user and a consumer agent. The user informs his/her preference on application QoS to the corresponding consumer agent (preference message), which reflects it in the utility function, typically by changing coefficients in the formula. Each consumer acts to acquire a bunch of goods that maximizes the value of the utility function (we call this value as just “utility,” hereafter), by changing the bid for the goods in its interest.

A producer agent represents the characteristic of a network application program, which transforms raw network resource into a service to be consumed by the user. Thus, there is a one-to-one correspondence between a producer agent and a service that is provided by an application program. Note that multiple services can be provided by a single program and that a single service can consist from multiple type of network resources. There is no one-to-one correspondence between a program and a producer agent, nor between a producer agent and any single type of network resources. The transformation from network resources to an application service is modeled as a production function, which takes the set of network resources as its input and output the corresponding service. When a service starts or finishes, a user provides or erases the agent which is corresponded the service (service message). Each producer selects its production level so as to maximize its profit,

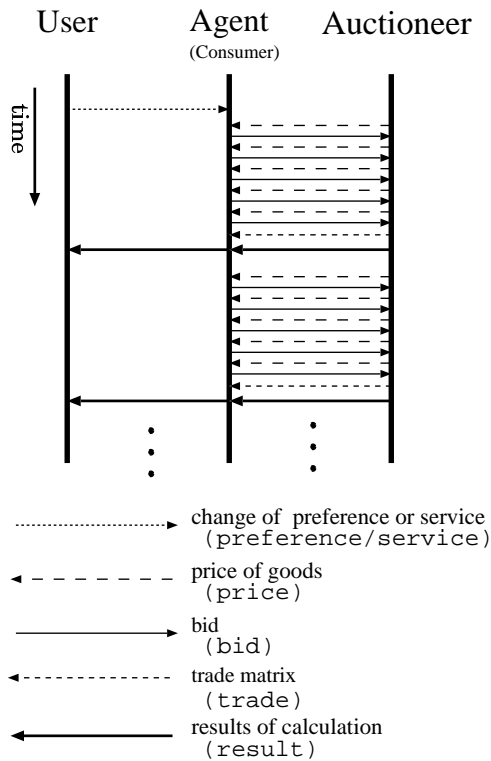


Figure 2. Message Exchange

which is defined as the difference between the income obtained by selling the services and the cost to purchase the network resources.

An auctioneer exists for each good. It adjusts the price of a good based on the bid from consumers and producers, so as to balance supply and demand of the corresponding good. The adjusted price is then sent to the agents and used by them to change their bid.

The market calculation in this model is performed as follows (Figure 3).

1. The auctioneer reports price of goods to agents(price message)(l.3).
2. Each agent calculates supply or demand of goods based on price information, to maximize their utility or profit(l.4). They then report them to the auctioneer(bid message).
3. The auctioneer aggregates the supply and the demand of goods reported by agents(l.5), and raises (lowers) the price if the demand is over (under) the supply(l.6–13).
4. If the difference between the supply and the demand is under the threshold, which is decided by users, the calculation terminates and the price exactly reflects the final allocation of network resources(l.15). If not, the auctioneer and the agents repeat from 1.

```

1: repeat
2:   begin
3:     Actioneer reports price to agents;
4:     Agent calculates supply or demand;
5:     Auctioneer aggregates supply and demand;
6:     if(supply over demand) then
7:       begin
8:         Auctioneer raises price;
9:       end
10:    if(supply under demand) then
11:      begin
12:        Auctioneer lowers price;
13:      end
14:    end
15: until(|supply - demand| > threshold);
16:
17: Agent calculates its share of bandwidth;
18: Agent reports it to QoS Client;
  
```

Figure 3. Market Calculation

5. When the calculation terminates, each producer agent calculates its share of network resources based on the final price, and reports it to QoS Client to control the communication of the corresponding application program (resource message), which uses the network resources and provides services to users(l.17,18).

The allocation has to be recalculated to adapt to the dynamically changing environment, and thus the above process (“calculation cycle” or just “cycle” hereafter) is repeated on every recalculation. If the overhead of computing equilibrium is large, resource allocation cannot follow the change of environment, not to mention that market computing itself occupies a considerable size of resource both in the network and computers. Therefore, the overhead of market computing should be as small as possible.

The resource message, which is used for auctioneers to report resource allocation to users, is transmitted whenever the auctioneers finish calculation. The preference message is used for users to report changes of their preference to agents, while the service message is used for network applications to report the changes in the services they provide to agents. The service message is transmitted whenever the behavior of application programs changes as well as the start of a new process. The number of above messages exchanged in the market computing is comparatively small compared to the messages defined below.

The price message is used for the auctioneers to report the price of goods to agents, and the bid message is used for agents to report the supply and the demand for goods. Both of them are transmitted whenever price of goods are

changed, thus the number of these messages exchanged is largest among other messages (typically, 10 to 100 times the number of agents in a calculation cycle). To minimize the overhead of market calculation, reducing the number of these two types of message is effective.

3. Implementation

In the rest of this paper, we present the underlying implementation issues, and discuss the implementation with mobile agents to our market-based network resource control system.

3.1. Mobile Agent Approach

In our market-based approach, agents send the auctioneers their bids, which are derived from the preference or the usage of network application program by users. Since users are distributed over network, there are typically two possible approaches to implement the agents.

The first one is to implement each agent in the host of the corresponding user, which is called *distributed implementation with static agents*(Figure 4(a), messages in the figure are the same as figure 2) here. In this case, since the calculation for generating bids can be performed simultaneously in their hosts, processors in the network are utilized effectively. On the other hand, as described in the previous section, all the bid and the price messages transmitted between agents and the auctioneers are performed as network communications, which causes large overhead in market computing.

We formulate this overhead as follows. Assume that the number of clients is n_c , and the number of calculation cycles necessary for market to reach an equilibrium is n_r . If the time to send and receive price information of goods is T_{price}^d , the time that agents calculate the most suitable supply and demand of goods is T_{agent} , the time to transmit supply and demand of goods is T_{bid}^d , the time for the auctioneers to adjust price of goods is $T_{auctioneer}$, and the time to transmit the results of calculation to users is T_{result} , then the time of one calculation cycle T_{static} is given as follows.

$$T_{static} = n_r(T_{price}^d + T_{agent} + T_{bid}^d + T_{auctioneer}) + T_{result} \quad (1)$$

The second approach is to implement all the utility functions in the host where the auctioneers reside. This is called *centralized implementation*(Figure 4(b)). In this case, the utilization of computation resource is worse than the first approach, while the communication cost is minimized.

Assuming the time to send and receive price information of goods is T_{price}^c and the time to transmit supply and

demand of goods is T_{bid}^c , then the time of one calculation cycle $T_{centralized}$ is given as follows.

$$T_{centralized} = n_r(T_{price}^c + n_c T_{agent} + T_{bid}^c + T_{auctioneer}) + T_{result} \quad (2)$$

The difference of one market calculation of equilibrium time between the case of distributed implementation (with static agents) and the case of centralized implementation is given as follows.

$$T_{static} - T_{centralized} = n_r(T_{price}^d + T_{bid}^d - T_{price}^c - T_{bid}^c - (n_c - 1)T_{agent}) \quad (3)$$

The balance between the time to transmit the bid and the price messages, and the time to derive bids based on the maximization problems of the agents determine which approach is suitable. In current network environment, it is obvious that the former surpasses the latter. This fact makes the distributed implementation far more costly than the centralized implementation.

In the centralized implementation, utility and production functions are defined statically in the same process as the auctioneers (“market server” hereafter). Users report their preference on applications by sending coefficients to be set in the functions to the auctioneers. Although this approach is relatively simple, the available utility function cannot be changed, which prevents the users from setting various types of preference, thus the system designer has to know all the types of user preference.

Since the privacy problem is serious in the centralized implementation, we examine the following two approaches by extending the distributed implementation with static agents.

1. Remote agent approach (Figure 4(c)) : Agents are implemented in the market server, but utility functions and production functions can be changed dynamically. Users send the expressions of utility and production functions to the market server. The expressions are interpreted by the remote agents.
2. Mobile agent approach (Figure 4(d)) : A group of agents is implemented as a mobile agent which keeps utility functions and production functions inside, and moves to auctioneer. Users send their preference on applications to their own agents.

The first approach provides more flexibility than the centralized implementation. However, this approach uncovers the users’ private preference, which is not preferable in the context of the market-based approaches, where minimal

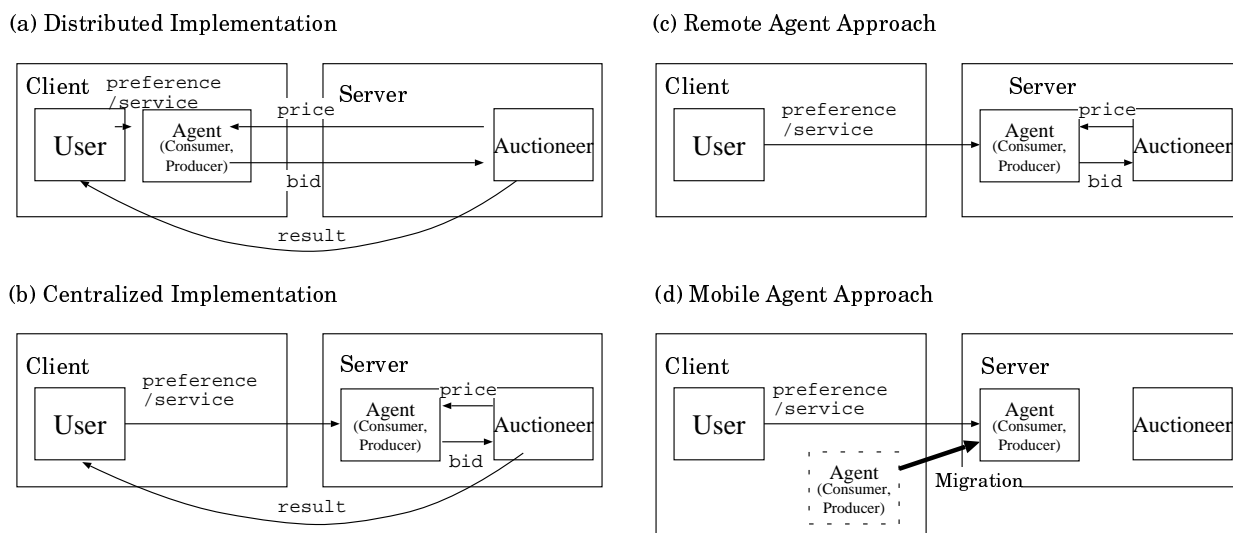


Figure 4. Implementation Model

information is sufficient to achieve the optimality of the whole system.

The second approach conforms both of above requirements, flexibility and privacy. It can implement dynamic change of user preference and network usage by nature, and security issue in mobile agent systems has been discussed extensively in the preceding work [7].

A mobile agent is a process object that can travel from one place to another in the middle of its execution preserving its context, and continue to run in the new place. A mobile agent can interact with the place it visits or with other mobile agents. Some of such environments provide a bidirectional authentication mechanism, where the servers authenticate the agents to protect themselves against malicious agents while the agents authenticate the servers.

These features enables our market-based allocation mechanism to achieve the efficiency and the flexibility of market calculation resolving the security issue that the users' preference must be kept private.

3.2. QoS Market

The system called QoS Market that we implement adopts mobile agents to solve the problem of computing equilibrium shown in the previous subsection.

Below, we explain the configuration of QoS Market, and the message interactions performed in it.

Figure 5 shows the configuration and the message flow of the system. The messages in the figure are the same as figure 2. The system consists of three types of processes : QoS Client, Mobile Agent, and Auctioneer. The arrows

indicate the interaction among them, including the move of agents and the flow of information, such as bids, price and resource allocation. Their functionality is as follows.

1. QoS Client

A QoS Client locates at each user's local computer, provides user interface to obtain preference from its user. A QoS Client tells the user's preference and the usage of network application to the corresponding Mobile Agents. It also receives the result of market computing, i.e., the allocation of network resource, and controls the transmission of data performed by the network application programs.

2. Mobile Agent

A Mobile Agent includes a consumer agent that represents its user's preference, and producer agents that represent application programs, and participates in market calculation generating bids based on current price information. The move of a Mobile Agent to the market server is done only once when its user starts QoS Client. The update of utility or production functions is done by receiving parameters from the corresponding QoS Client. A Mobile Agent informs the current resource allocation at the end of every calculation cycle.

3. Auctioneer

Auctioneers reside in the market server, and perform the adjustment of the price of goods. In our current system, they are implemented as a single thread that

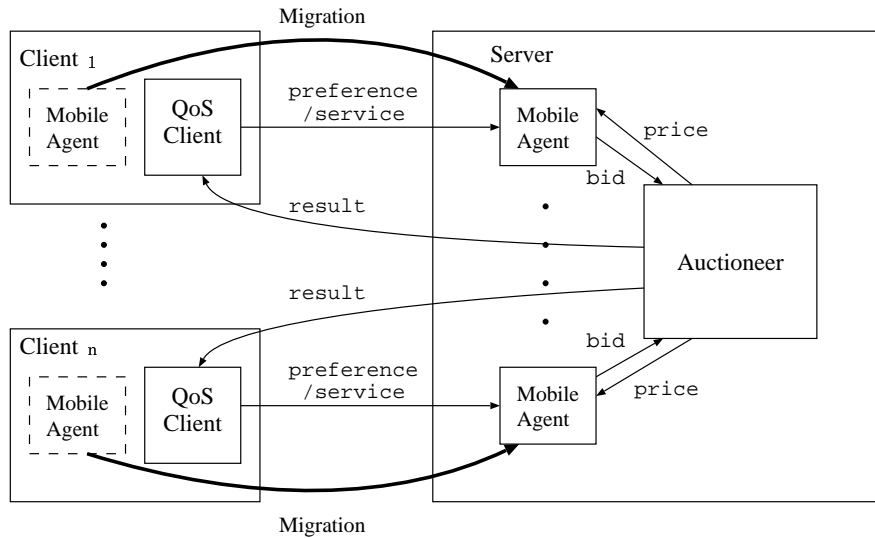


Figure 5. System Configuration

handles all the goods. The thread judges the convergence of the resource allocation, and tells the end of a cycle to all Mobile Agents.

All these components have been implemented on Windows NT, and, except for the move of mobile agents, the network communication are performed using Microsoft's Distributed COM(DCOM) technology, which extends the Component Object Model (COM) to support communication among objects on different computers

Even though many mobile agent systems are being developed, such as Aglets, Odyssey, and Voyager [4, 3, 6], most of them are based on Java, Tcl or other interpreters, thus are not suitable for our application where the overhead in computation is critical. We decided to implement a subset of a mobile agent system with the minimal set of functionality, using Microsoft's ActiveX technology, a Component Object Model optimized for a network, on WindowsNT. A group of agents is implemented as an ActiveX component, which are downloaded from the client processes to the market server and run as a part of it.

At first, a mobile agent exists as a DLL file, a Dynamic Link Library, at each client. When a user logs in the QoS Market, the user sends the information about his/her mobile agent to the market server. Then the market server downloads the DLL of mobile agent. The market server links the DLL dynamically, and initializes the mobile agent. After this, the mobile agent communicates the QoS Client, and receives the information about the preference of the user and applications, then provides a consumer and producers. In market calculation, the communication between the mar-

ket server and the mobile agent is executed by the market server's calling a function of the mobile agent.

Because the mobile agent by ActiveX is written in native code, to know users' utility function, which is privacy, we have to analyze the code by reverse engineering, or presume from the answers of the mobile agent to various questions. These usually require a great deal of time, and we think this mobile agent is secure enough.

Since the component of ActiveX moves without context, we do not assert that this is an ideal mobile agent system. Actually, this made our design of the agent interactions a little tricky, and we think this is a temporary, but practical enough for evaluating the feasibility of the mobile agent approach. We plan to adopt other mobile agent systems with more run-time efficiency in the future.

4. Experiment

Here, we examine the performance of QoS Market in order to verify the mobile agent approach to the market-based network resource allocation.

4.1. Settings

To estimate the system performance, we make two experiments as follows.

1. The evaluation environment consists of two PCs with Windows NT 4.0. The market server is running at Host B. They are connected by 10Mbps Ethernet, and the

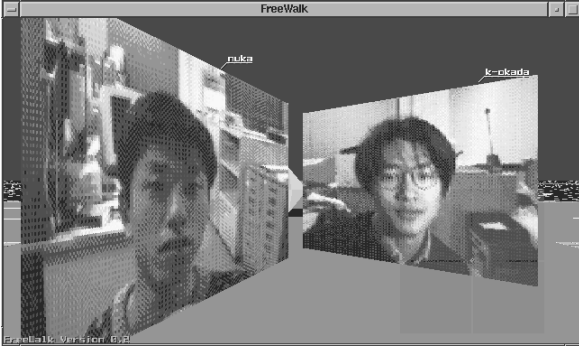


Figure 6. FreeWalk: A Multimedia Conferencing Tool

data have been recorded at Host A without the market server.

Two applications on Host A communicate Host B, so there are two communication from Host A to Host B. Thus there are five goods, which are CBW, FBW, q_{AB1} , which represents the one of the communication from Host A to Host B, q_{AB2} , and FQoS.

To estimate the system performance, we measured the time for the calculation and the communication performed in each stage of a market calculation cycle. This is to estimate the difference between the mobile agent approach and the static agent approach.

2. Evaluation environment consists of three PCs with Windows NT 4.0. The FreeWalk, a multimedia conferencing tool, is running at each host, and the market server is running at one of them. They are connected by 10Mbps Ethernet, and the data have been recorded at the two hosts without the market server.

In this environment, only one user change his/her preference, while the preference of the other two users is kept unchanged. Since there are three users in the 3D meeting space, two network services are provided to each user. The user inputs his/her preference on these services as a ratio of their importance.

The user inputs twenty times a random ratio of importance of one service every three seconds. A ratio of importance of another service is set the rest ratio.

To estimate the system performance, we measured the deviation in the resource allocation, which is caused by the delay of market computing. When the user changes his/her preference, there is always a certain amount of delay before the new resource allocation that reflects the change is derived. During this period, network

Table 1. The Times for Each Stage of the Calculation

	Static Agent Approach(ms)	Mobile Agent Approach(ms)
T_{moving}	0	700
T_{price}	86	0
T_{agent}	0.64	0.64
T_{bid}	86	0
$T_{auctioneer}$	0.2	0.2
T_{result}	82.75	82.75
Estimated Total Time of Market Computing	3419.8	95.8

communication is performed using the old allocation, which causes the data to be transmitted more/less than ideal amount in each connection and to cause loss in the user's utility.

4.2. Results

Table 1 shows the time for each stage of the calculation. The T_{price} , which is the time to transmit price from an Auctioneer to a QoS Client via network, is as large as 0.86msec. On the other hand, in the mobile agent approach, there is almost no overhead because it can be done as an inter-process communication inside the same host. The time consumed by the calculation in an agent and an auctioneer, T_{agent} and $T_{auctioneer}$ respectively, are the same in both implementations. The time to send the allocation result to QoS Market via network, T_{result} , is 82.75msec in both implementation.

Mobile agent approach requires the time to move agents in the network other than those shown in Table1, and the T_{moving} , which is the mean time for a mobile agent to move from the QoS Client to the market server is about 700msec. This, however, does not have much impact in the efficiency, since the move is performed only once in the lifetime of the agent and does not occur in each of the calculation cycle.

The total time of a market calculation cycle is then estimated by applying these results to the equation 1 and 2 in Section 3.1. Assuming $n_r = 20$, the time of one cycle in the static agent approach is

$$T_{static} = 20(83 + 0.64 + 83 + 0.2) + 83 = 3419.8(ms),$$

while that in the mobile agent approach is

$$T_{mobile} = 20(0 + 0.64 + 0 + 0.2) + 83 = 95.8(ms).$$

From this result, we conclude that there is a large difference in the efficiency of market calculation between static / mobile agent approaches.

According to the result of the second experiment, the average overhead is suppressed 457.8msec, which means that the system achieves practically sufficient efficiency.

We believe superiority of the mobile agent approach to the static agent approach will not be changed in several years. Since the size of each message is small enough compared to the total bandwidth, the delay in the transmission of the messages are caused almost exclusively by the network latency. Thus, the performance will not be improved much even if the network technology advances, because such progress is usually in bandwidth rather than in latency.

5. Conclusion

The protection of users' private preference and the efficient calculation are both important in the market computing to allocate network quality of service most efficiently, based on users' preference. In this paper, we discussed the implementation of the market-based application QoS control system, which derives optimal resource allocation based on users' preference, from the viewpoint of efficiency, flexibility and privacy. To achieve these objectives, we proposed an approach based on mobile agent technology, and implemented the QoS Market system to validate the discussion.

From the experimental results, we conclude as follows/

1. The mobile agent approach achieves far more efficient calculation than the static agent approach, preserving the merits in flexibility and privacy.
2. Our implementation succeeds in responding to the change of user preference fast enough, with the delay of approximately 450ms, and causes minimal communication overheads.

In this paper, we assumed an intranet of average size. Future research will cover a large-scale intranet, where the performance must be further improved because the delay by networks becomes more serious in the process of resource allocation.

References

- [1] R. Braden, L. Zhang, B. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification," *Internet Request for Comments*, RFC2205, 1997.
- [2] S. H. Clearwater (Ed.), *Market-Based Control: A Paradigm for Distributed Resource Allocation*, World Scientific, 1996.
- [3] General Magic Odyssey WWW Page, <http://www.genmagic.com/agents/odyssey.html>, 1996.
- [4] D. B. Lange, D. T. Chang, "IBM Aglets Workbench – Programming Mobile Agents in Java," IBM Corporation White Paper, Sept. 1996.
- [5] H. Nakanishi, C. Yoshida, T. Nishimura, and T. Ishida, "FreeWalk: Supporting Casual Meeting in a Network," *Proc. of CSCW'96*, pp. 308 – 314, 1996.
- [6] Object space, Inc., "ObjectSpace Voyager Technical Overview," <http://www.objectspace.com/Voyager>, 1997.
- [7] T. Sander, C. F. Tschudin, "Protecting Mobile Agents Against Malicious Hosts," Giovanni Vigna (Ed.), *Mobile Agents and Security*, LNCS 1419, Springer, 1998.
- [8] M. P. Wellman, "A Market-Oriented Programming Environment and Its Application to Distributed Multicommodity Flow Problems," *Journal of Artificial Intelligence Research*, Vol. 1, pp. 1–22, 1993.
- [9] H. Yamaki, Y. Yamauchi and T. Ishida "Implementation Issues on Market-Based QoS Control," *ICMAS-98*, pp. 357 – 364, 1998.
- [10] H. Yamaki, M. P. Wellman and T. Ishida "A Market-Based Approach for Allocating QoS to Multimedia Applications," *ICMAS-96*, pp. 385 – 392, 1996.