Journal of Information Processing Vol. 19 25–38 (Feb. 2011)

Regular Paper

A Self-Organized Overlay Network Management Mechanism for Heterogeneous Environments

Tsutomu Inaba,[†1] Hiroyuki Takizawa[†2] and Hiroaki Kobayashi[†3]

The technologies of Cloud Computing and NGN are now growing a paradigm shift where various services are provided to business users over the network. In conjunction with this movement, many studies are active to realize a ubiquitous computing environment in which a huge number of individual users can share their computing resources on the Internet, such as personal computers (PCs), game consoles, sensors and so on. To realize an effective resource discovery mechanism for such an environment, this paper presents an adaptive overlay network that enables a self-organizing resource management system to efficiently adapt to a heterogeneous environment. The proposed mechanism is composed of two functions. One is to adjust the number of logical links of a resource, which forward search queries so that less-useful query flooding can be reduced. The other is to connect resources so as to decrease the communication latency on the physical network rather than the number of query hops on an overlay network. To further improve the discovery efficiency, this paper integrates these functions into a self-organizing resource management system, SORMS, which has been proposed in our previous work. The simulation results indicate that the proposed mechanism can increase the number of discovered resources by 60% without decreasing the discovery efficiency, and can reduce the total communication traffic by 80% compared with the original SORMS. This performance improvement is obtained by efficient control of logical links in a large scale network.

1. Introduction

The Next Generation Network (NGN) is now attracting a great deal of attention as a next generation network infrastructure and incorporates a number of useful features, such as high security, high availability and so on[1)]. With the popularization of NGN, various resources such as personal computers (PCs), game consoles, cameras, telephones, and sensors will be connected to the network through Home Gateway [2)]. The performances of these resources are rapidly improving. However, it is rare for their individual owners to fully exploit the performances; most of the resources are usually in an idle state.

To utilize these idle resources effectively, many grid computing projects such as BOINC are active [3),4)]. However, participants in these projects only provide their resources and cannot run their own jobs. To provide suitable computing resources to a resource user anytime, anywhere, it is important to construct an infrastructure for mutual use of many unspecified resources. This infrastructure sustained by mutual use of individual computing resources extends the period of actively using each computer, and thereby can promote an ecological and sustainable society. This infrastructure requires an effective discovery mechanism to discover a resource that satisfies user's requirements, such as CPU resources and application programs, from a huge number of resources on the network. However, if the discovery mechanism needs central servers to manage information of a huge number of resources, it requires an extremely high cost to update the information on every joining or leaving of individual resources. Therefore, a resource discovery mechanism based on Peer-to-Peer (P2P) communication without central servers or *brokers* is required.

A P2P overlay network consists of many resources that are connected by logical links. In a P2P resource discovery system, a resource user sends a query to neighbor resources via logical links to ask if they satisfy the user's requirements. Hereafter, a *user request* indicates a set of user's requirements for resource peers, and the user request is forwarded as a *search query*. A resource receiving the query further forwards it to the neighbor resources. The resource user can find a resource satisfying the requirements if the query reaches the resource. In the P2P overlay network, each resource can independently maintain its routing table. Therefore, it does not require an administrator who manages the whole network, and thereby avoids traffic concentration on specific resources. In addition, since it has no single point of failure, it realizes high fault tolerance. However, its critical drawback is that query flooding caused by multicasting/broadcasting query packets increases a large amount of network traffics as the search area expands.

Motivated by this, a self-organizing resource management system (SORMS)

†1 NTT EAST-Miyagi
†2 Graduate School of Information Science, Tohoku University
†3 Cyberscience Center, Tohoku University

based on P2P communication has been proposed [5),6)]. Since each user requests resources for its own purpose, the resources requested by the user in a short period likely have similar features. We call this *the user's locality of interests*. Therefore, SORMS routes a query preferentially to resources, which are more likely to satisfy user's requests by a reconnection of logical links that exploits user's locality of interests.

Our previous research shows that the discovery mechanism of SORMS achieves high efficiency. However, the original SORMS have not fully considered influences from the *usage frequency* at which a resource is used by the others. Since the number of logical links per resource is fixed, the degree of relationship between two peers that is represented by each logical link is significantly different. Generally, a frequently-used peer has strong relationships with others. In an environment where each resource has a different performance, individual resources have different usage frequencies. Therefore, if all logical links are evenly managed in spite of the difference, resource discovery efficiency might degrade.

Moreover, in a real network with various communication delays and bandwidths, which is called a physical network, the communication delay must be considered in the overlay network optimization because an optimized packet route on the overlay network tends to be redundant on the physical network. Nonetheless, the original SORMS does not consider the communication delay in a physical network at all, but only the number of P2P sessions in the overlay network.

In this paper, we propose an adaptive overlay network management mechanism that enables SORMS to efficiently adapt to such a heterogeneous environment. The proposed mechanism is achieved with two ideas. One is that a resource which has a high usage frequency can have a lot of logical links. The other is that resources located near on the physical network are preferentially connected by logical links. The experimental results indicate that the proposed mechanism can effectively increase the probability of discovering requested resources and then, decrease the delay for discovery.

The rest of the paper is organized as follows. Section 2 explains an overview of the original SORMS. Section 3 points out the problems of SORMS, and then proposes an adaptive overlay network management mechanism to extend the capability of SORMS for heterogeneous environments. Section 4 discusses the performance of the proposed mechanism through simulation experiments. Section 5 reviews the related work and Section 6 concludes the paper.

2. SORMS: A Self-Organized Resource Management System

In a P2P-based resource discovery mechanism, resources are discovered by flooding query packets over an overlay network without a special broker such as directory servers. However, this flooding search mechanism causes a large amount of communication traffic. To reduce this problem, SORMS reconnects logical links among resources based on users' locality of interests, and reconstructs the overlay network so that the resources can be discovered in a fewer number of hops.

In SORMS, since a computing resource such as a PC or game console is considered as a peer on an overlay network, a computing resource is called a *resource peer* in the following. From the perspective of mutual use of resources, a resource peer sometimes becomes a client peer.

2.1 Exploit Locality of Interests in Resource Acquisition

In general, a lot of continuity and similarity are found in a sequence of actions in terms of time and space, named *locality*, in many situations. For example, if a user is doing one thing right now, the probability to do it again in the near future will become high. In addition, the probability of taking an action somewhat related to the current one will be higher than that of taking a completely independent action. Therefore, if the future actions/requests can successfully be predicted with some locality from current and past ones, the efficiency in dealing with the next action will be improved remarkably. Cache memory is a typical example, which takes advantage of temporal and spatial locality of memory reference to bridge the gap in speed between a processor and an off-chip memory system [7)].

SORMS defines two types of the locality, *temporal locality* and *spatial locality*, which can be exploited locally from the features observed in some recent resource requirements of each user. The temporal locality is derived based on the observation that recently-used resource peers are likely to be needed again in the next future, i.e., if a client peer A is using a resource peer B right now, resource peer B will be requested again in the near future with a higher proba-

bility. Temporal locality suggests that a user's request should be propagated to recently-used resource peers first. Accordingly, each client peer should be connected with recently-used resource peers so as to directly send query packets to the resource peers. A set of the resource peers directly connected with a client peer A is called a *cluster* of the client peer A.

In addition, some peers are clustered and work together to realize a virtual computing environment with requested specific functions and performance. Such relationship among these peers is called spatial locality.

If two clusters of users A and B have a shared resource peer, they share a common interest. In this case, user A can be expected to have a higher possibility of using the resource peers in the B's cluster. The similar idea can be seen when we check book titles at Amazon.com [8]. Amazon.com lists the most popular items related to the item that a customers are checking, because customers who bought that item also tend to purchase its related ones that were suggested by the interests of other users. Therefore, in *Social Networking Service* (SNS) and *Contents Delivery Network* (CDN), people with similar interests can be clustered with SORMS.

2.2 Overview of SORMS and Link Control Algorithm

In order to place more useful resource peers around their client peers, SORMS is designed to efficiently search resources in large-scale P2P systems under the consideration of the locality of user's interests. Since a resource peer sometimes becomes a client peer, each peer owns one-way logical links named *access links* and *forward links*.

Access links from each client peer to recently-used resource peers keep the relationship based on the temporal locality as shown in **Fig. 1**, so that future queries will directly be sent to the resource peers via access links. Here, a user's query is comprised of requirements for static attributes such as CPU type and application type, and for dynamic attributes such as CPU load and available memory. Forward links are provided to connect resource peers within a cluster of users as shown in **Fig. 2**. Figure 2 also depicts the extension of the search space by connecting similar clusters through shared resource peers.

2.3 Access Link Control Algorithm

Access links are one-way links connecting a client peer with most-recently-used (MRU) resource peers, and these are managed in an *Access Link Destination List* of each client peer. From the viewpoint of temporal locality, MRU resource peers are the most likely used ones. Let L_A be the maximum number of access links per client peer. Then, access links are updated to keep connecting a client peer with L_A MRU resource peers.

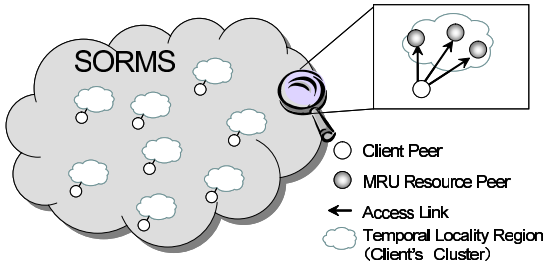

Fig. 1 Temporal locality constructed by access links.

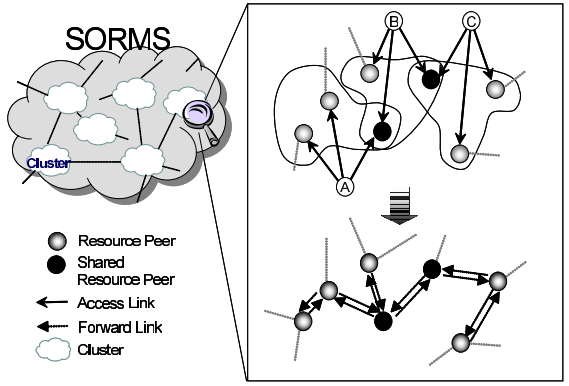

Fig. 2 A concatenation of spatial localities constructed by forward links.

Figure 3 illustrates the procedure of updating access links. A counter is assigned to each access link for its LRU (Least-Recently Used) control, which means that the link toward the least-recently used resource peer becomes a candidate for reconnection. Here, let peer A be the client peer issuing request messages and then finding resource peer C as an appropriate computing resource; that is called *"hit."* Resource peer B is the LRU resource peer of A. Always after

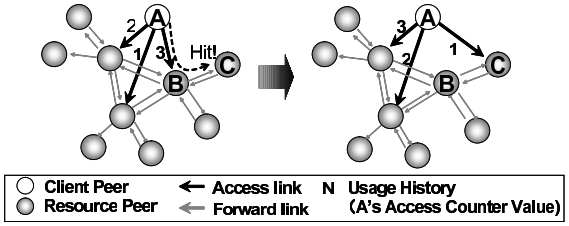

Fig. 3 The procedure of updating access links.

using resource peer C, client peer A updates all access links and their counters as follows:

(1) Check whether there is an access link between A and C. Go to Step 3 if there is a link.
(2) Create a new access link that directly connects A with C. If the number of access links exceeds L_A, delete the access link whose counter is maximum, i.e. the access link from A to B.
(3) Set the counter of the access link for C to zero.
(4) Finally, increment all counters.

2.4 Forward Link Control Algorithm

Forward links are managed by resource peers rather than client peers. Therefore, each resource peer has a *Forward Link Destination List*, which manages the direction and the counter of each forward link. Each counter indicates the order, in which a forward link has been updated. Let L_F be the fixed number of forward links per resource peer. A forward link is initially created as a pair of two one-directional links, which works as one bidirectional link connecting two resource peers recently used by the same client peer. During self-organization, each direction of a forward link is independently managed by its origin resource peer, using its own counter.

Figure 4 illustrates the procedure of updating forward links. Every time a resource peer is used, it updates all of its forward links. Hence, forward links are updated after access links of the client peer are updated. In Fig. 4, a resource peer C, which has just been used by a client peer A, updates its forward links as follows. Note that each number within a square in this figure means the counter of each forward link. The smaller the counter value is, the newer the updated

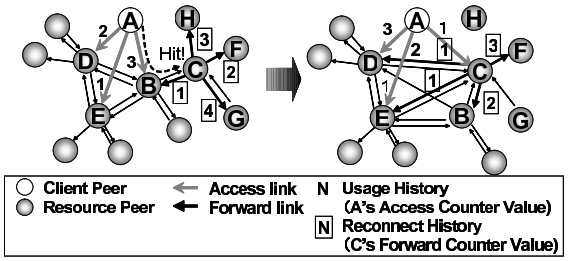

Fig. 4 The procedure of updating forward links.

forward link is. In the following, let L_F be four.

(1) Get the updated list of access links of A that is the Access Link Destination List of A.
(2) Check whether there are unconnected resource peers in the list. Go to Step 5 if all the resource peers in the list except for C are connected with resource peer C.
(3) Create a new forward link to one of the unconnected resource peers, i.e. either D or E. As a forward link is bi-directional, the newly connected resource peer also creates an opposite direction link to C.
(4) Delete one of the forward links whose counter is maximum, and go to Step2. Since L_F is four, the forward link from C to G in Fig. 4 is deleted first, and the link from C to H is deleted next.
(5) Set the counters of the forward links to the resource peers in the received access link list to zero.
(6) Finally, increment all counters.

3. An Adaptive Overlay Network Management Mechanism for Heterogeneous Environments

Compared to an overlay network with no optimization mechanism, the original SORMS can find more peers with less network traffic. The rate of discovered peers to searched peers increases more than tenfold as shown in previous work [5),6)]. However, the network traffic is defined by the aggregated number of P2P sessions for forwarding users' queries over an overlay network.

The original SORMS does not consider the differences in performance among

peers such as CPU types and available memory capacities. Moreover, it does not consider the physical communication delay among peers, which results from the differences among their communication capabilities such as connected network access services and bandwidth of network interfaces. Therefore, in this section, we propose an adaptive overlay network management mechanism to get a higher resource discovery efficiency of SORMS for heterogeneous environments, in which computing resources have different usage frequencies and communication capabilities. This mechanism is achieved by a function to regulate the number of forward links and a function to reduce the communication delay.

3.1 Dynamic Adjustment of the Number of Forward Links

In the original SORMS, forward links of a resource peer are mainly reconnected when the resource peer is used by a client and gets a request for forward link reconnection from the client. As a result, those forward links are frequently updated if the resource peer is frequently used by others. In this way, each resource peer itself remembers how frequently it has been used by others. As the number of times used as a resource peer indicates the number of forward link updates, a resource peer also knows how frequently each forward link has been updated. If the number of forward links of a resource peer is fixed, the lifetime of each link differs depending on the usage frequency of the resource peer. This is because forward links are reconnected in an MRU manner. A resource peer with a high usage frequency repeats reconnecting forward links to keep the user's locality updated. On the other hand, a resource peer with a low usage frequency cannot reconnect its forward links and the links still represent obsolete temporal locality. To enable a resource peer with a high usage frequency to stably keep useful routes, such a resource peer needs more links than a resource peer with a low usage frequency. Therefore, in a heterogeneous environment where resource peers have different usage frequencies, the number of forward links managed by each resource peer should not be a constant.

To keep useful forward links managed by the resource peers with high usage frequencies, the resource peers should have many forward links. Resource peers with low usage frequencies should delete forward links periodically, because the links may reflect obsolete locality. Therefore, this paper proposes a forward link management mechanism based on utilizations of individual forward links. In this mechanism, each forward link is managed during a certain available period, and deleted after the period if it is never used in its available period. As a result, a resource that has a high usage frequency can keep a lot of forward links, while a resource with a low usage frequency can have few links. Whenever a resource peer C receives a search query, C deletes its expired forward links from its Forward Link Destination List by checking the elapsed time of each forward link as follows:

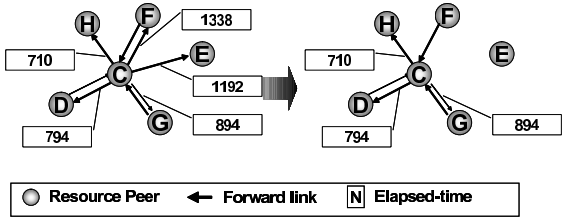

Fig. 5 Proposed forward link deletion.

Let $F_{\max}$ be the maximum number of forward links per resource peer, and $F_{\min}$ be the minimum one. These values are given as system parameters in the entire SORMS domain. **Figure 5** shows the proposed forward link deletion where $F_{\max}$ is set to five. In this figure, the expired forward links to E and F are deleted.

(1) Check whether the number of forward links is equal to $F_{\min}$. If the number is equal to $F_{\min}$, the forward link deletion is terminated.
(2) Select a forward link whose elapsed time is maximum, and check whether the time exceeds an expiration time. If it is expired, delete the link, and go to Step 1. Otherwise, the forward link deletion is finished.

In the forward link control algorithm shown in Section 2.4, each forward link is managed by its origin resource peer, and the forward link with the maximum counter value is deleted when a new forward link is created. However, in the proposed mechanism, each forward link is managed based on the elapsed time since it was created. This mechanism also uses the upper bound of the number of logical links $F_{\max}$ that a peer can have. When the number of logical links exceeds $F_{\max}$, the oldest link is deleted. **Figure 6** illustrates the proposed forward links optimization, and its' procedure is as follows:

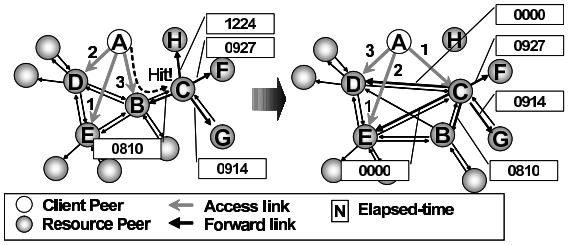

Fig. 6 Proposed forward link optimization.

(1) Get the updated list of access links of A that is the Access Link Destination List of A.
(2) Check whether there are unconnected resource peers in the list. Go to Step 5, if all the resource peers in the list except for to C are connected with resource peer C.
(3) Create a new forward link to one of the unconnected resource peers, i.e., both D and E. As a forward link is bi-directional, the newly connected resource peers also create an opposite direction link to C.
(4) Repeat a deletion of the forward link whose elapsed-time is maximum, until the number of forward links is up to $F_{\max}$ or less. In Fig. 6, because the value of $F_{\max}$ is set to five, the forward link from C to H is deleted only, and the links from C to F and G still remain.
(5) Finally, set the time-stamp of the forward links to the resource peers in the received access link list to updated time.

Figure 7 shows how the number of forward links changes over time. This figure shows an example where each resource peer can have up to five forward links. The destinations of forward links are denoted by A to E, and the length of an arrow indicates the available period of a forward link. The horizontal axis means the elapsed time. Thus, the left end of an arrow means the creation time of a forward link, and the right end is the deletion time. As shown in the left side figure, a resource peer with a high usage frequency can create many forward links in a short period, resulting in having many forward links whose available periods are overlapping. On the other hand, in the right-side figure, a resource peer

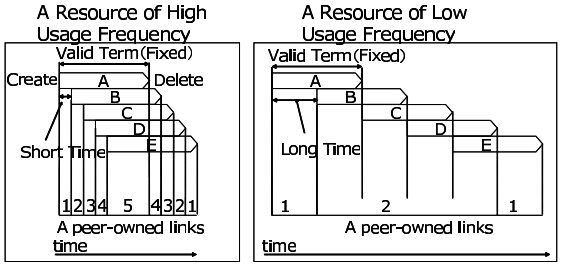

Fig. 7 Mechanism to adjust the number of forward links.

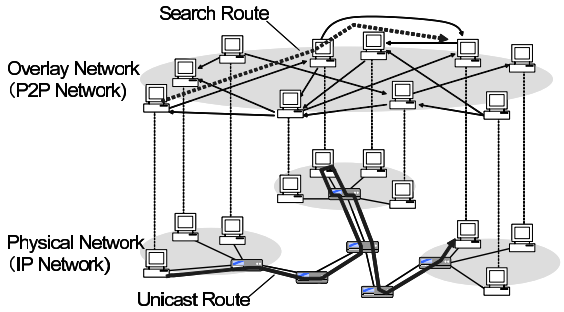

Fig. 8 The difference between an overlay network and a physical network.

with a low usage frequency can have fewer forward links because their available periods are hardly overlapping. If a resource peer that is the destination of a forward link is used again, the available period of the forward link is extended. Therefore, strongly-correlated resources can be connected for a long time. In this way, the proposed mechanism can adjust the number of forward links so that each peer can have an appropriate number of forward links depending on its usage frequency.

3.2 Reduction in the Communication Delay

The original SORMS cannot optimize a communication traffic delay, because it does not consider the physical network lying under the overlay network of logical links. **Figure 8** illustrates the difference between an overlay network and a physical network. The shortest path on an overlay network might be a long path on a physical network. In general, a search path on the overlay network

progressively becomes different from the unicast path on the physical network as the number of hops increases, if the communication delay is not taken into account at logical link reconnection.

In the original SORMS, an access link directly connects a client peer with the resource peer used by the client peer. If there are several resource peers that satisfy the client's requirements as shown in **Fig. 9** (a), the client peer has several prospective destination resource peers, which can become the destination of a newly created access link. Meanwhile, forward links are created between the two resource peers that are recently used by the same client peer. Hence, if a client peer selects the nearest resource peer, to which the communication delay is the shortest, two resource peers connected by a forward link are likely to be close to each other.

To measure a communication delay between two peers on the physical network, an ICMP echo message is commonly used. Although ICMP echo is not accurate to measure the communication delay in the proposed mechanism, it is not important to select the nearest resource, but to avoid selecting a resource peer with long delay. Therefore, a client peer can know which is the most nearest resource peer on the physical network, by using the round trip time of an ICMP echo.

As shown in Fig. 9 (b), if a client peer always selects a hit resource peer with the minimum communication delay for job execution, the selected resource peers connected via forward links are likely to exist near to each other. As a result, an effective overlay network is organized so that nearby peers are connected via forward links. Although the communication delay between two peers changes over time, SORMS continuously reorganizes the overlay network and thereby adapts to the changes.

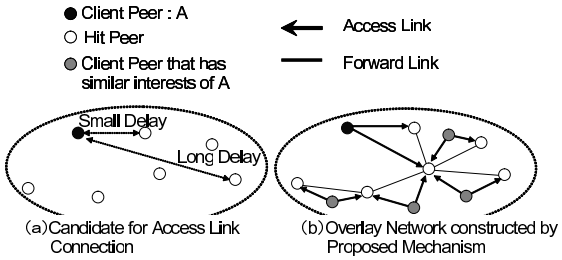

Fig. 9 Mechanism to optimize the discovery latencies.

4. Performance Evaluation

4.1 Simulation Model

In this section, we evaluate the effectiveness of the proposed mechanism based on the simulation model described in Ref. 9).
In our experiments, each peer is in one of four states as follows.

1) *Off* : do nothing.
2) *Requesting* : sending a query.
3) *Waiting* : waiting for a query from other peers.
4) *Processing* : executing a job submitted by a client peer.

The state of each peer changes at fixed intervals based on a statistical state transition model, as shown in **Fig. 10**. The state also changes triggered by the arrival of the messages from other peers. A resource peer in either State *Off* or *Waiting* can stochastically transit to State *Requesting*. After sending a query, the status of the peer changes to State *Off* or *Waiting* immediately.

Table 1 shows parameters used in the simulation. In the simulation, a query is forwarded to the next neighboring resource peers at one simulation cycle, referred to as *one hop*. Queries propagate through access links at the first hop, and through forward links at the second hop and later. If a resource peer satisfies the requirements of a query ("hit"), the resource peer replies to the query. In the case where a client finds two or more hit resource peers, the client submits

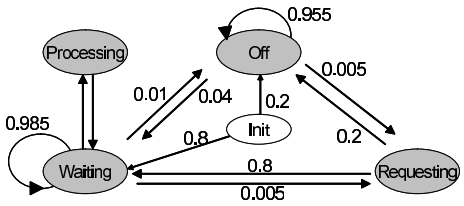

Fig. 10 State transition diagram of each peer.

Table 1 Simulation Parameters.

Parameter	Value
Number of peers	100,000
Request state transition probability	0.005
Number of Access Links	3
Number of Minimum Forward Links	3
Number of Maximum Forward Links	6
Number of Maximum Hops	4
Number of Service Types	F00–F99
Number of Service Types per peer	2
Computing Power (performance)	1–100
V_p	1.0
V_a	0.1

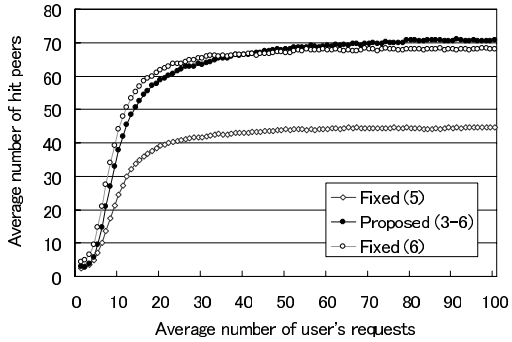

Fig. 11 Comparison in the average number of hit peers.

a job to one of the resources at random. The number of access links per client peer is set to 3, because too many access links rapidly expand the area of discovery [10]. The number of forward links per resource peer ranges from 3 to 6 to demonstrate the benefit of the proposed mechanism, i.e., it needs fewer forward links for discovery. The services provided by each resource peer are modeled by identifications, F00–F99 (100 types of services, and the services with closer numbers are related more strongly). The performance is quantified by a single value ranging from 1 to 100. A query message has a pair of the service and performance as requirements given by a user. For example, pair(F00, 50) can hit only the resource peers, which have functionality F00 and whose performance exceeds 50. The trend in requested performance and services issued by users changes based on the discrete-time Wiener process [11] where the time is defined by the number of requests from the user. These requirements are determined with a normal distribution where the previous requirement is used as the mean. Therefore, a resource request is similar to the previous one from the same user; the behavior of resource requests from a user gradually changes. In this way, a user's behavior with some locality is modeled. In Table 1, V_p and V_a mean the variances of two Wiener processes that statistically generate users' requirements for performances and services, respectively. V_p is set to 1.0 to make the change of performance requirement be less than 30%, and V_a is set to 0.1 to make the change of service requirement be less than 10%. The resource peer receiving a search query forwards the received query to the neighboring resource peers connected via forward links, after checking if it can satisfy the requirements of the query or not. Thus, queries are delivered over a P2P network, and the resource peers that satisfy the user's requirements reply to the query. Logical links, performance, and services of a peer are initially determined at random.

4.2 Simulation Results

4.2.1 The Number of Hit Peers

Figure 11 shows the number of *hit peers* per request on the overlay networks of the original and optimized SORMS mechanisms. In this figure, the horizontal axis indicates the average number of user requests that increases with time, and the vertical axis indicates the average number of hit peers per request. In this figure, "Fixed" means the original SORMS and the number indicates the fixed number of forward links per a resource, and the "Proposed" means the proposed SORMS in which the number of forward links changes from three to six. Then, the average number of hit peers increases with time in any case. This is caused by the optimization of original/proposed SORMS in which resource peers that provide similar services are localized and connected together via logical links. These results clearly indicate that both of the original and proposed mechanism can construct an overlay network effectively to discover the requested peers in fewer hops, based on users' locality of interests.

The number of hit peers in the case of six forward links is 1.55 times larger than that in the case of five links when 100 user requests have been sent. The

number of hit peers generally increases when a lot of forward links are given to a resource peer, because a query can arrive at more peers within the same number of hops. On the other hand, the average number of hit peers in the proposed mechanism is 1.60 times larger than that with five links, even though the average number of forward links in the proposed mechanism is only 4.53. This means that the proposed mechanism can find more resource peers with fewer forward links.

4.2.2 Discovery Efficiency

Figure 12 shows the discovery efficiency. In this figure, the average rate of the number of hit peers to that of the searched resource peers with six links is smaller that with five links. From the results of Figs. 11 and 12, the discovery efficiency degrades if the number of forward links of every resource peer simply increases. However, the proposed mechanism can obtain more hit peers whilst maintaining the high discovery efficiency, because the number of forward links of each resource peer can be effectively decreased. Those results clarify that the proposed mechanism is effective to improve the discovery efficiency by managing forward links based on their available periods.

However, if a resource peer has been in the *Off* state for a long time, all the forward links heading to the resource peer are often deleted because their available periods are expired. Although such a resource peer can still participate in the overlay network as a client peer, it will never be used by the others because it cannot receive any queries. In this case, the peer as a resource is isolated from the others. It is observed that several percent of the resource peers are isolated from the overlay network in the case where resource peers reuse their previous links when they change from the *Off* state to another state. Therefore, these results also suggest that the link reconnection at rejoining the overlay network is one important technical issue. This issue will be discussed in our future work.

4.2.3 Available Period of a Forward Link

Next, the available period of a forward link is evaluated. **Figure 13** shows the average number of hit peers and the average number of *searched peers*, which receive a query in a certain period. This figure shows that the longer period a forward link can live, the more resource peers a client peer can get. The number of hit peers reaches 76.6 as a peak in the case where the available period is 20,000 cycles, but it shows no further increase for a longer available period. On the other hand, the number of searched peers continues to increase with an increase in the average period. This means that the discovery efficiency gradually decreases as the available period becomes longer, even though the area of resource search expands. This is clear from **Fig. 14** that shows the average rate of the number of hit peers to the number of searched peers. This figure shows that the average rate decreases when the available period is too long. These results mean that the number of searched peers increases because the number of available links increases when the available period is long, even though the searched peers rarely hit.

From these results, it is clearly demonstrated that the proposed mechanism

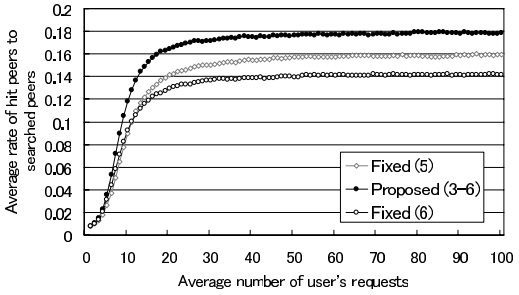

Fig. 12 Comparison in the average rate of hit peers to searched peers.

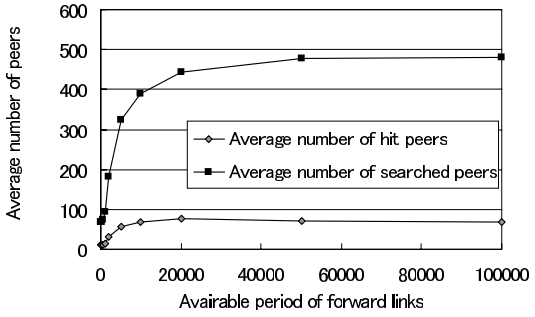

Fig. 13 Effects of the available period on the average numbers of hit peers and searched peers.

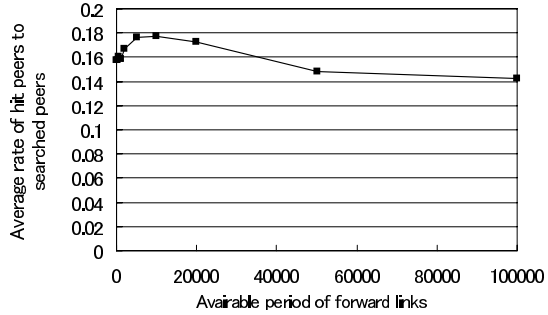

Fig. 14 Effects of the available period on the average rate of hit peers to searched peers.

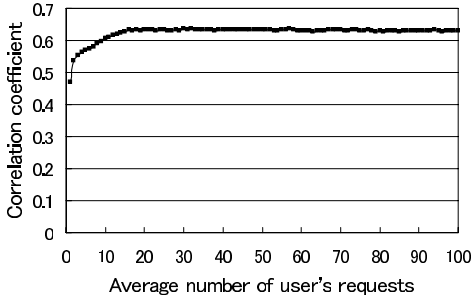

Fig. 15 Time change in correlation between usage frequency and the number of forward links.

can keep connecting only the strongly-correlated resource peers and deleting redundant links connecting weakly-correlated resource peers.

4.2.4 Automatic Adjustment of the Number of Forward Links

In the following, the relationship between the usage frequency and the number of forward links is discussed.

Figure 15 shows the time change in correlation between usage frequency and the number of forward links. In Fig. 15, the correlation between the usage frequency and the number of forward links gradually increases with time. This means that the proposed mechanism can adjust the number of forward links so that resource peers with higher usage frequencies can have more forward links. As higher-performance resource peers are generally used more frequently, these

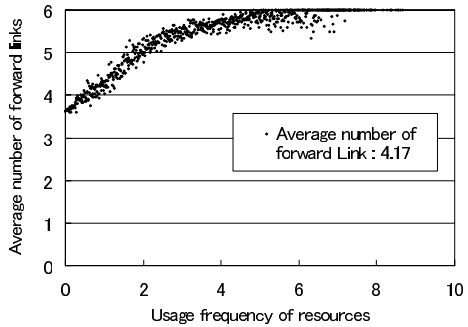

Fig. 16 Relationship between the usage frequency and the number of forward links.

results suggest that higher-performance resource peers have more forward links. **Figure 16** shows the relationship between the usage frequency and the average number of forward links during the 90th user requests to the 100th requests in the steady state. This figure also shows that the more frequently used resource peers have more forward links.

These results suggest that the proposed mechanism can properly adjust the number of forward links of each resource peer based on its usage frequency.

4.2.5 Communication Latency Optimization

In the following, the network optimization capability of the proposed mechanism is evaluated from the viewpoint of communication latency. In the evaluation, Ring Topology, Tree Topology and Inet Topology are used as the basic topologies of a physical network. The Inet Topology is generated by an Internet topology generation tool Inet-3.0 [12], which can generate a network topology similar to the actual Internet topology.

Figure 17 shows the average reduction rates at 100 requests in terms of *Traffic* and *Latency Stretch* [13] at the Ring Topology and the Tree Topology. Here, Traffic means the number of queries forwarded on the physical network and is normalized by the result of the original SORMS. Latency Stretch is defined as l_l/l_u that is the rate of discovery delay l_l to unicast delay l_u. The discovery delay l_l means the number of query hops on the physical network from a client peer to a hit peer through the resource peers that routed a discovery query. The unicast delay l_u means the number of direct hops on the physical network from a client peer

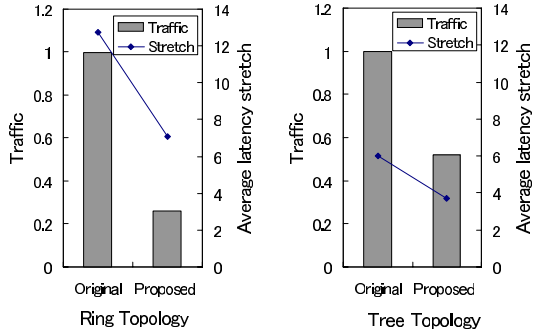

Fig. 17 Traffic and the average latency stretch (100,000 peers and 100 requests in average).

to a hit peer. If Latency Stretch is large, a query is forwarded via a redundant route that is totally different from the unicast route. This figure indicates that Traffic and Latency Stretch on Ring Topology can be reduced by 73.8% and 55.6%, respectively. Similarly, Traffic and Latency Stretch on Tree Topology can be reduced by 48.2% and 38.3%, respectively.

The reduction in Latency Stretch means that queries reach resource peers in fewer hops on the physical network, and hence the overlay network becomes efficient to save the network traffic. The improvement by the proposed mechanism for Ring Topology is more remarkable than that for Tree Topology. This is because the average number of hops required for communication between two resource peers in Ring Topology, i.e., the network diameter of Ring Topology, is larger than that of Tree Topology.

Figure 18 shows the reduction rates achieved by the proposed mechanism for various network diameters of Tree Topology. In this figure, as the network diameter grows, the reduction rates in both Traffic and Latency Stretch tend to rise. The results indicate that the proposed mechanism can reduce communication latencies very much compared with the original SORMS in which logical links are reconnected at random, especially in a large scale network environment. Therefore, the proposed mechanism works more effectively for a large-scale network with a long diameter that consists of many peers.

Finally, the performance gain of the proposed mechanism in a realistic network environment is evaluated with Inet Topology. To evaluate the performance gain,

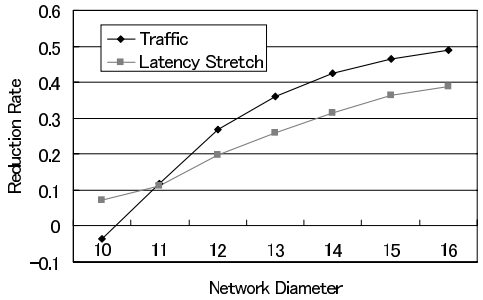

Fig. 18 Network diameter vs. reduction rate (tree, 100 requests in average).

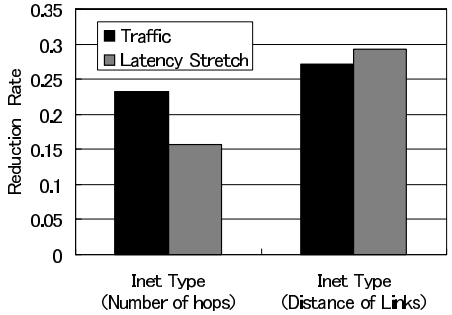

Fig. 19 Reduction rates in Traffic and Latency Stretch for the Inet Topology (10,000 peers and 100 requests in average).

two parameters are defined. One is the $h/h_{\max}$, which means the reduction in Latency Stretch, when the number of hops on the physical network is considered the communication delay. The other is the $L/L_{\max}$, which means reduction in Latency Stretch, when the distance between two peers on the physical network is considered their communication delay. Here, h is the numbers of hops on the physical network and $h_{\max}$ is the end-to-end hops on the physical network. In addition, L is the distance of a path on the physical network between two resource peers, and $L_{\max}$ is the distance of the path between the uttermost resource peers.

Figure 19 shows the reduction rates of Traffic and Latency Stretch at the Inet Topology with 10,000 peers. From Fig. 19, the proposed mechanism reduces Traffic by 23.2% and Latency Stretch by 15.6% on Inet Topology if the delay is

evaluated with the number of hops. Moreover, the proposed mechanism reduces Traffic by 27.1% and Latency Stretch by 29.4% on Inet Topology if the delay is evaluated with the distance between two peers. Thus, if the delay is evaluated with the distance, the reduction rates of Traffic and Latency Stretch are 3.9% and 13.8% higher than those with the communication delay caused by the number of hops. In the case where there are two resource peers that satisfy the user's requests, if the communication delay to one resource peer is smaller than that to the other, the proposed mechanism generates a logical link to the former resource peer without considering the number of hops. As the result of this, each user's locality space of interests is optimized not only in an overlay network but also in the physical network from a viewpoint of communication latency. Therefore, those results clearly demonstrate that the proposed mechanism is effective to make the logical route on the overlay network similar to the unicast route on the physical network, resulting in a reduction in the communication latency.

5. Related Work

A discovery mechanism based on an overlay network can be classified into two types [14]: *Structured Overlay* and *Non-Structured Overlay*. Structured Overlays such as Chord [15], Pastry [16] and CAN [17] are not appropriate for shared resource discovery; because it is so expensive to keep consistency of routing tables distributed over resource peers especially if resource peers repeatedly join and leave, or if the internal state of each resource peer such as the available memory size is always changing. Meanwhile, Non-Structured Overlays do not need to manage routing tables to adapt to the changes in participating resource peers and their internal states, because they assume that a user sends a query directly to resource peers that may satisfy his/her requirements. However, it tends to cause a query flooding problem. To solve this issue, self-organizing mechanisms based on monitoring a relationship with neighbor peers [18)–21)] have been proposed.

TellaGate [18] reconnects logical links and adjusts the number of those links so that resource peers repeating similar requirements are closely connected based on the information obtained from peers within two hops. TellaGate is efficient when a client peer requests other peers that have similar attributes to the client peer. However, it becomes ineffective if a client peer requires other peers whose attributes are totally different from those of the client peer. Moreover, TellaGate uses a *peer digest* that represents the information used to assess the similarity among peers. A peer digest for a peer is extracted from some words in documents and file names provided by the peer. In the resource discovery mechanism, however, there are many relations among providing services, even if the service names differ from each other. Therefore, it is difficult to get a meaningful relation for resource discovery only from the similarities in service names.

The self-organizing mechanism proposed in Ref. 19) dynamically reconstructs an overlay network by monitoring the route of forwarding responses. In this mechanism, a peer constantly observes responses passing over the peer itself. Then, logical links are reconfigured based on the importance of each forwarding route evaluated by the frequency of response forwarding. However, this mechanism requires a large amount of calculation with an increase in the number of forward links, even if the destination of a reconnected forward link is limited to the peers within two hops. Discovery efficiency is decreased by the computational complexity, especially in an environment where peers repeatedly join and leave the network.

In file sharing systems based on a P2P overlay network, some systems such as Freenet [20] and Winny [21] achieve high discovery efficiency by dynamically changing the query-forwarding destinations. However, these dynamic network configuration mechanisms are used under the assumption that shared files are distributed in advance. In the case of file sharing, the discovery efficiency can be improved by simply replicating popular files. On the other hand, in the case of service resource sharing, replication is usually impossible. Hence, dynamic link reconnection techniques are required to restrict the area of query flooding. The purpose of service resource sharing systems discussed in this paper is clearly different from that of file sharing.

The above discussions suggest that the proposed SORMS is effective for mutual use of general public resources whose behaviors frequently change. Discovery efficiency of the original SORMS might decrease in a heterogeneous environment with usage frequencies and different network latencies. However, the evaluation results indicate that the proposed mechanism can cover the drawbacks and hence enable SORMS to achieve a high discovery efficiency even in such an environment.

6. Conclusions

This paper proposed an adaptive overlay network management mechanism for heterogeneous environments, in which computing resources have different usage frequencies and communication capabilities, to efficiently limit the area of query flooding. The proposed mechanism is composed of two functions. One is a function to regulate the number of forward links of each resource according to its usage frequency. The other is a function to reduce the communication latency.

The simulation results indicate that the proposed mechanism can reduce both latencies and communication traffics for resource discovery by effective management of logical links. In addition, the proposed mechanism is especially effective for a large scale network with an Internet-like topology.

Currently, we are developing a mechanism to retry resource discovery when no required resource is found within a localized search area. We will also discuss logical link reconnection when a peer rejoins the overlay network. In addition, we will apply SORMS to clustering of users in Social Networking Service (SNS). This will be discussed in our future work.

References

1) Next-Generation Services Joint-Development Forum. http://www.ngs-forum.jp/
2) Soma, S., Furuya, S., et al.: Services by home gateway for home appliance control, *Proc. Society Conference of IEICE*, B, No.2, p.129 (Sep. 2008). (in Japanese)
3) Korpela, E., Werthimer, D., Anderson, D., Cobb, J. and Lebofsky, M.: SETI@home: Massively distributed computing for SETI, *Computing in Science and Engineering*, Vol.3, pp.78–83 (Jan./Feb. 2001).
4) Anderson, D.P.: BOINC: A system for public-resource computing and storage, *5th International Workshop of Grid Computing* (*GRID 2004*), pp.4–10 (Nov. 2004).
5) Kobayashi, H., Takizawa, H., Okawa, T. and Inaba, T.: An Effect Control Mechanism for Self-Organizing Overlay Networks of Large-Scale P2P Systems, *Interdisciplinary Information Science*, Vol.13, pp.227–238 (2007).
6) Kobayashi, H., Takizawa, H., Inaba, T. and Takizawa, Y.: A Self-Organizing Overlay Network to Exploit the Locality of Interests for Effective Resource Discovery in P2P Systems, *The 2005 International Symposium on Applications and the Internet* (*SAINT2005*), pp.246–255 (Jan. 2005).
7) Patterson, D.A. and Hennessy, J.L.: *Computer Organization & Design*, Morgan Kaufmann Publishers Inc, 2nd (1998).
8) Bauch, P.: *Amazon Hacks – 100 Industrial-Strength Tips & Tools*, O'Reilly & Associates, Inc. (2003).
9) Okawa, T., Takizawa, H. and Kobayashi, H.: An Evaluation and Modeling of Resource Discovery in Large Scale P2P Systems, *Inf. Technol. Lett.*, Vol.4, pp.21–24 (2005). (in Japanese)
10) Inaba, T., Murata, Y., Takizawa, H. and Kobayashi, H.: Consideration of Resource Access History for Optimizing Overlay Networks in P2P-based Resource Discovery, *the 2008 International Symposium on Applications and the Internet*, pp.269–272 (2008).
11) Kleinert, H.: *Path Integrals in Quantum Mechanics, Statistics, Polymer Physics, and Financial Markets*, 4th edition, World Scientific, Singapore (2004).
12) Winick, J. and Jamin, S.: *Inet-3.0: Internet Topology Generator*, University of Michigan Tech Report, CSE-TR-456-02 (2002).
13) Zhang, H., Goel, A. and Govindan, R.: Improving Lookup Latency in Distributed Hash Table Systems Using Random Sampling, *IEEE/ACM Trans. Networking*, Vol.13, No.5 (Oct. 2005).
14) Ezaki, H., et al.: *P2P textbook*, Impress Japan Corporation R&D (2008).
15) Stoica, I., Morris, R., Karger, D., et al.: Chord: A scalable peer-to-peer lookup service for internet applications, *Proc. ACM SIGCOMM 2001*, pp.149–160 (2001).
16) Rowstron, A.I.T. and Druschel, P.: Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," *IFIP/ACM International Conference on Distributed Systems Platforms* (*Middleware 2001*), pp.329–350 (2001).
17) Ratnasamy, S., Francis, P., Handley, M., et al.: A Scalable Content-Addressable Network, *Proc. ACM SIGCOMM*, pp.161–172 (2001).
18) Kojima, K.: Tellagete: Self-Organization Approach of Communities for P2P Networks, *Lecture Notes in computer Science*, Vol.3144, pp.108–119 (2004).
19) Nakano, H., Harumoto, K. and Nishio, S.: A Topology Re-formation Algorithm for P2P Networks Based on Response Statistics, *The Database Society of Japan* (*DBSJ*) Letters, Vol.5, No.1, pp.41–44 (June 2006).
20) Clarke, I., Sandberg, O., Wiley, B., et al.: Freenet: A distributed anonymous information storage and retrieval system, *Proc. ICSI Workshop Design Issues in Anonymity and Unobservability*, pp.311–320 (July 2000).
21) Kaneko, I.: *The Thechnology of Winny*, ASCII MEDIA WORKS Inc. (2005). (in Japanese)

(Received June 4, 2010)
(Accepted November 5, 2010)
(Released February 9, 2011)

Tsutomu Inaba is currently a system engineer of NTT EAST-Miyagi. He is mainly in charge of business-academia collaboration projects. His research interests include P2P network computing and its applications. He received his B.E. degree in Mechanical Engineering, and M.S. and Ph.D. degrees in Information Sciences from Tohoku University in 1993, 1995 and 2010, respectively. He is a member of the IPSJ.

Hiroyuki Takizawa is currently an associate professor in Graduate School of Information Sciences, Tohoku University. His research interests include high-performance computing systems and their applications. He received his B.E. degree in Mechanical Engineering, and M.S. and Ph.D. degrees in Information Sciences from Tohoku University in 1995, 1997 and 1999, respectively. He is a member of the IEEE CS, the IEICE, and the IPSJ.

Hiroaki Kobayashi is currently a director and professor of Cyberscience Center and a professor of Graduate School of Information Sciences, Tohoku University. His research interests include high-performance computer architectures and their applications. He received his B.E. degree in Communication Engineering, and M.E. and D.E. degrees in Information Engineering from Tohoku University in 1983, 1985, and 1988 respectively. He is a senior member of IEEE CS, and a member of ACM, IEICE and IPSJ.