Regular Paper

Strategy for Selecting Replica Server Spots on the Basis of Demand Fluctuations

Masato Asahara,[†1] Akio Shimada,[†1]
Hiroshi Yamada[†1] and Kenji Kono[†1]

Many service providers distribute various kinds of content over the Internet. They often use replica servers to provide a stable service. To position them appropriately, service providers must predict the demands for their services and provide a computing capacity sufficient for servicing the demands. Unfortunately, predicting demands is difficult because the demand for a service usually fluctuates. Our research group is developing ExaPeer, an infrastructure that apportions computing capacity to services running on hundreds or thousands of trusted machines all over the Internet. ExaPeer allocates computing capacity to the services running on top of ExaPeer. In this paper, we describe ExaPeer's approach to dynamically select candidate spots for replica servers. This approach, called EPSS (ExaPeer Server Selection), detects fluctuations in the demand and dynamically selects candidate spots on which replica servers should be placed to best meet the demand. Even if the demand on a service increases dramatically within a short term, EPSS quickly responds to the situation and rapidly selects many candidates for replica servers. To deal with short-term fluctuations, EPSS is designed to be lightweight; each machine determines whether to be a candidate spot independently of others. Experimental results demonstrate that the candidate spots selected by EPSS work better than those selected by heuristics even if the scale of the demand changes rapidly. For 90% of all client accesses, the round-trip-times (RTTs) with EPSS were 23% less than those with randomly selected machines even if the number of machines is 6.7 times larger than EPSS.

1. Introduction

Many service providers distribute various kinds of content such as web pages, streaming videos, and music over the Internet. Getting these contents over the Internet is the daily activity of many people. To provide a stable service, service providers often prepare *replica servers*, or mirror servers, and position them appropriately on the Internet. The robustness of the service is improved because the load is distributed over several servers. Even if some of the servers are not functioning or reachable, the other servers can continue to provide the service.

†1 Department of Information and Computer Science, Keio University

To position replica servers appropriately on the Internet, service providers must predict demands for their services and provide a computing capacity sufficient for meeting the demands. If the estimated capacity is lower than the capacity actually needed, they cannot provide the service in a satisfactory quality. If the estimated capacity is abundant, the computing resources prepared for the service are wasteful and often costly. If the demand for a service is almost constant, the provider can easily increase or decrease computing capacity to eventually meet the actual demand. Unfortunately, the demand for a service generally *fluctuates* on the Internet; so providers have little time to adjust the capacity.

To deal with demand fluctuations, many researchers are developing infrastructures for flexibly adjusting the amount of computing capacity. For example, Amazon.com started a service called Amazon Elastic Compute Cloud (Amazon EC2)[1], which enables users to increase or decrease the computing capacity within a few minutes. Our research group is developing ExaPeer, an infrastructure comprising a *pool* of hundreds or thousands of trusted machines all over the Internet. ExaPeer apportions a computing capacity to services running on top of itself; it detects demand fluctuations and adjusts the number and the locations of replica servers on the basis of these fluctuations. If the demand on a service A increases, the number of replica servers for A also increases. If the demand on a service B decreases, the number of replica servers for B also decreases. To use ExaPeer, the user registers a virtual machine image that can be replicated to provide the intended service anywhere in ExaPeer.

In this paper, we describe ExaPeer's approach, called *EPSS* (ExaPeer Server Selection), to dynamically select *candidate spots* for replica servers. In ExaPeer, EPSS detects fluctuations in a demand and dynamically selects candidate spots on which replica servers should be placed. EPSS targets short-term fluctuations as well as long-term ones. Even if the demand for a service dramatically increases, EPSS immediately responds to the situation and selects candidate spots for replica servers. In fact, it is quite common for the demand for a service to fluctuate dramatically in a very short term. Flash crowds[2),3)] exemplify this. In flash crowds, a web server suddenly receives an avalanche of client accesses by, for example, the "slashdot" effect; that is, many readers access a URL given in a story listed on `http://slashdot.org`[4),5)].

EPSS is designed to be lightweight and responsive to short-term fluctuations. It uses an autonomous approach to avoid complicated protocols for sharing the load information across the network; each machine in ExaPeer determines whether to be a candidate independently of the others. Each machine analyzes access paths from clients by exchanging messages with only adjacent machines and detects the areas in which the demand for a service is increasing or decreasing. It estimates the benefit of being a replica server from this access path analysis.

Simulation results demonstrate that the candidate spots for replica servers selected by EPSS work better than the ones selected by heuristics. For 90% of all client accesses, the round-trip-times (RTTs) with EPSS were 23% less than those with randomly selected machines even if the number of machines is 6.7 times larger than EPSS. The results also demonstrate that EPSS can adjust the number and the locations of candidate spots even when the demand fluctuates quickly. For 90% of all client accesses, the round-trip-times (RTTs) with EPSS were 47% less than those with randomly selected machines even when the number of machines was 33 times longer than EPSS.

The rest of the paper is organized as follows. Section 2 discusses the design issues. Section 3 explains how EPSS dynamically selects candidate spots. Section 4 describes the evaluation of EPSS. Section 5 discusses the qualitative features of EPSS. Section 6 describes related work, and Section 7 concludes with a brief summary and a look at future work.

2. EPSS Design Issues

To distribute the load from clients, it is significant to position replica servers appropriately. Since ExaPeer hosts several services at the same time, it must be able to dynamically adjust the number of replica servers for *each* service. If the demand on a particular service increases, the number of replica servers for the service is increased to distribute the load across replica servers. If the demand decreases, the number of replica servers is decreased to reduce the waste of hardware resources.

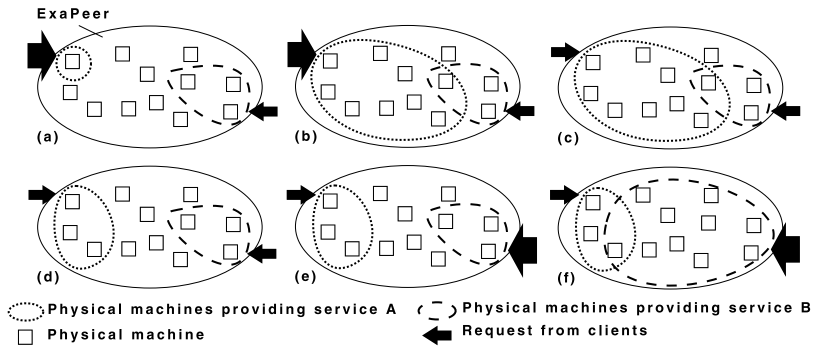

ExaPeer
(a)
(b)
(c)
(d)
(e)
(f)
Physical machines providing service A
Physical machines providing service B
Physical machine
Request from clients

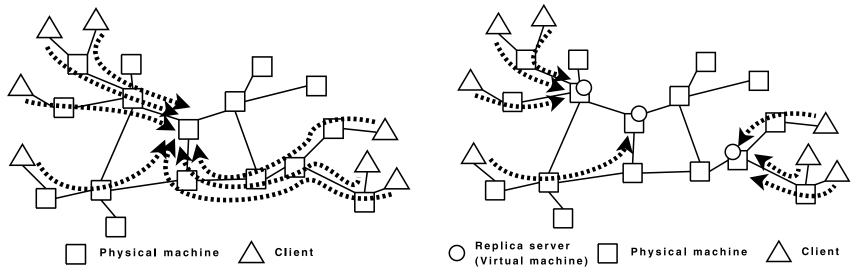
Fig. 1 Dynamically adjusting the number of replica servers.

Figure 1 illustrates the concept of ExaPeer. The demand for a service A increases in Fig. 1 (a). In response to this increase, several physical machines start to provide service A (Fig. 1 (b)). After that, the demand for a service A decreases (Fig. 1 (c)). Therefore, as we can see from Fig. 1 (d), some physical machines spontaneously stop providing service A. Likewise, some physical machines start to provide service B when the demand for it increases (Fig. 1 (e) and (f)).

In addition, ExaPeer tries to position replica servers appropriately to reduce the network traffic; if a replica server is placed on a physical machine that relays many requests on the service, the network traffic would be reduced because the replica server can handle the requests without sending more messages. **Figure 2** shows the difference between the cases with and without ExaPeer. In this example, with ExaPeer, the load of clients is distributed across three machines after repositioning replica servers (Fig. 2 (b)). Note that ExaPeer assumes that a replica server has already been installed on each physical machine.

ExaPeer provides a fundamental service whereby one can select candidate spots for replica servers. An ExaPeer's strategy of selecting candidate spots, *EPSS*, addresses the following issues to deal with unpredictable and rapid fluctuations in a service demand.

- *Lightweight Detection of Demand Fluctuation*:
  EPSS handles a short-term fluctuation in service demands as well as a long-term fluctuation. To deal with short-term fluctuations, EPSS must respond



(a) Without ExaPeer. A single physical machine provides a service. (b) With ExaPeer. Three machines are providing a service.

Fig. 2 ExaPeer tries to position replica servers appropriately to distribute the load and reduce the network traffic.

quickly to demand fluctuations. Thus, EPSS which collects load information must be lightweight; otherwise, a short-term fluctuation could not be detected because the load information would not be collected frequently enough to identify a short-term fluctuation. Thus, it would not be a good idea for a centralized server to collect and analyze access logs from the entire network. Since the cost of collecting the logs is not low, we cannot collect them in a timely fashion to detect short-term fluctuations.

- *Topologically-Aware Detection of Demand Fluctuation*:
EPSS takes a physical network topology into consideration when selecting candidate spots. By leveraging the topological knowledge, EPSS selects candidate spots so that the network traffic can be reduced. If a replica server is placed on a machine closer to clients, the network traffic is reduced because the distance over which messages are relayed is shorter. In addition, if a replica server is placed where many access paths from clients cross, it can naturally host many requests without having to relay them to other hosts.
- *Transparent Access to Repositioned Servers*:
If the demand for a certain service fluctuates dramatically during a short period, EPSS can reposition a large number of replica servers within a short period. Therefore, EPSS cannot rely on the traditional DNS-RR (round-robin) for finding replica servers by a user because the delay involved in updating the DNS database is too large to meet the time constraint. To ensure a transparent access to repositioned servers, ExaPeer must include a mechanism for quickly finding repositioned replica servers.

3. Selecting Candidate Spots

In ExaPeer, each physical machine detects a demand fluctuation independently of others. There is no special server dedicated to collecting or analyzing information on the demand fluctuation. EPSS confines the area to be investigated only to adjacent physical machines and thus eliminates the cost of collecting load information from the entire network. In this section we describe the basic structure and the mechanism of EPSS, which selects candidate spots for replica servers.

3.1 Overview

ExaPeer constructs a topologically-aware overlay network for use in finding physical machines (including client machines) on the Internet. This overlay network is also used to find the replica servers providing a service requested by a user. ExaPeer assumes the underlying overlay to have the following properties.

- Physical machines have d-dimensional coordinates (d is usually 6 to 8 in GNP) such that the distance calculated from the coordinates approximates the RTT between any pair of physical machines. Global Network Positioning (GNP)[6] and Vivaldi[7] are examples of these systems called network coordinate systems. An overlay network cannot control the coordinates because they are calculated from the measured RTTs.
- A query message is relayed to a replica server providing the requested service. The message is relayed so that it gets closer to the final destination. If a distributed hash table (DHT) is constructed in a topologically-aware way, the resulting overlay network usually has this property.

A straightforward approach to finding candidate spots is to collect and analyze the coordinates of all clients accessing a certain service. With this approach, we should be able to calculate nearly optimal locations of candidate spots by solving an optimal location problem. Unfortunately, this approach is not suitable for dynamically repositioning replica servers because it is a time- and memory-consuming task to collect and analyze all the coordinates. Therefore, we could not select candidate spots quickly enough to respond to short-term fluctuations in a service demand.

With EPSS, a physical machine detects the direction in which there is an area with a high demand for a particular service. Since ExaPeer assumes that a query message is relayed so that it gets closer to the final destination, a physical machine that relays a query message from one direction to another can infer that there is a high-demand area in the direction from where the query comes. If there are many queries from one direction, EPSS selects a physical machine in that direction as a candidate. If the newly selected machine also detects that many queries are coming from one direction, it also selects a candidate in that direction. In this way, the candidate spots selected by EPSS get closer to a high-demand area; EPSS continues to improve the locations of candidate spots.

3.2 ExaPeer Overlay Network

To detect the direction of a high-demand area, the current prototype of ExaPeer constructs an overlay network by using GNP[6] and a Content-Addressable Network (CAN)[8]. GNP models the Internet as a d-dimensional space (d is usually 6 to 8) and assigns a d-dimensional coordinate to each physical machine so that the Euclidean distance between any pair of coordinates approximates the RTT between the physical machines. CAN is one of the distributed hash tables (DHTs). It assigns each physical machine a virtual d-dimensional coordinate and divides the d-dimensional space into *zones*. A physical machine stores (key K, value V) pairs if the zone maintained by the machine contains the coordinate $(h_0(K), \cdots, h_{d-1}(K))$, where h_i is a hash function. If key K is given, CAN guarantees that a physical machine managing the key K can be reached in $(d/4)(N^{1/d})$ hops on average where N denotes the number of nodes on CAN.

Using GNP and CAN, ExaPeer constructs a topologically-aware overlay network. The coordinate calculated by GNP is used, in turn, to construct the CAN. As a result, adjacent machines on ExaPeer are physically closer than other machines. When a service is registered to ExaPeer, it selects a physical machine, called a *root*, to host the service. The root machine is selected in accordance with the CAN protocol.

There are some things to note. First, ExaPeer is *not* tightly coupled with CAN and GNP. For example, ExaPeer can use SCoLE[9] or Vivaldi[7] instead of GNP and eCAN[10] instead of CAN. Second, ExaPeer uses virtualization technologies[11] to deploy replica servers. Each physical machine on ExaPeer can run several virtual machines (VMs) to provide multiple services at the same time. ExaPeer assumes that the VM images have already been distributed on each machine.

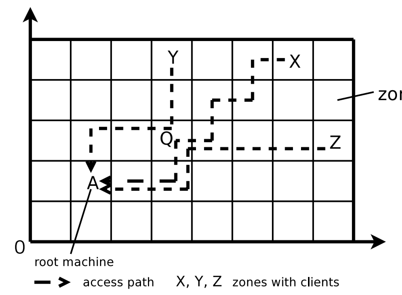


Fig. 3 Selecting a candidate spot on the basis of the degree of access path convergence.

3.3 EPSS Basic Mechanism

Figure 3 illustrates the EPSS mechanism, using a 2-dimensional CAN. In this figure, physical machine A is selected as a root. Client X sends out a service query, which is relayed according to the CAN protocol until it reaches the root. Then, clients Y and Z also send out the query.

EPSS selects candidate spots for replica servers on the basis of the *degree of access path convergence*. An *access path* is a network path on the overlay along which a query is relayed from a client to a target server. The degree of convergence is the number of access paths that cross the physical machine of interest. Each physical machine in ExaPeer maintains the degree of convergence for each service. When a physical machine Q relays a query message for service A, it increments the degree for service A. In Fig. 3, the degree of convergence for service A is three on the physical machine Q.

To estimate the degree of access path convergence, each physical machine records the number of requests for *each service* that it relays to another machine. If the demand for a service is increasing in some areas, the number of relayed requests also increases along the access paths. A physical machine is considered a candidate if its degree of access path convergence reaches a pre-defined threshold, *upper*. If the degree drops below another threshold, *lower*, the machine ceases to be considered a candidate. In Fig. 3, the physical machine Q is a

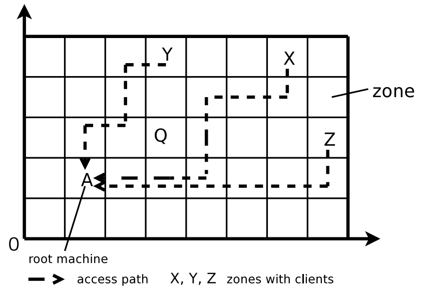


Fig. 4 Failure to select candidate spots.

candidate spot for replica servers because there are three access paths across Q (if *upper* is 3).

EPSS uses a sliding window to count the number of relayed messages and periodically checks the degree of convergence. To avoid the situation where a virtual machine starts and stops frequently, *lower* is set sufficiently lower than *upper*.

3.4 Improving EPSS Mechanism

The basic mechanism of EPSS sometimes fails to select candidate spots appropriately. In the example shown in **Fig. 4**, three clients, X, Y, and Z, are accessing root A. Note that these are the same clients as in the example in Fig. 3, but the access paths are more distributed. As a result, all machines cannot detect the increasing demand.

There are two reasons for failing to detect an increasing demand. First, CAN can choose different access paths even if the same client is sending out query messages because CAN has some flexibility in choosing access paths. Second, if the high-demand area is wide and far from the root machine, the root machine is accessed from many more directions, reducing the likelihood that the access paths converge on some machine. ExaPeer solves this problem by using two techniques: virtual path expanding (VPE) and virtual demand raising (VDR).

3.4.1 Virtual Path Expanding

Virtual path expanding (VPE) expands the of an access path. It increments the convergence degrees of the physical machines adjacent to an access path, thereby virtually expanding the breadth of an access path. In **Fig. 5**, VPE increments

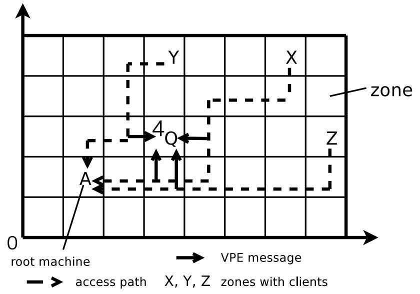


Fig. 5 Virtual path expanding (VPE). The convergence degrees of physical machines adjacent to an access path are incremented.

the convergence degrees of the physical machines adjacent to the paths from X, Y, and Z to A. For example, Q is adjacent to all access paths, so the convergence degree of Q becomes four although no access paths cross Q.

With VPE, each physical machine along an access path sends a VPE message to its neighbors. On receipt of a VPE message, each machine increments by one its degree of convergence. Since VPE messages are sent to only neighbor machines, the number of VPE messages is not so large, and the physical machines are not burdened.

3.4.2 Virtual Demand Raising

If a root machine is accessed from many different directions, the increasing demand cannot be effectively detected even if VPE is used. To more aggressively detect the direction in which there is a high-demand area, ExaPeer uses (VDR). When a query message is relayed to the physical machine that actually provides the requested service, VDR increments the convergence degree of the relaying machine by two, thereby raising the probability of selecting candidate spots in the direction of the high-demand area. As illustrated in **Fig. 6**, R is adjacent to the physical machine (A) that actually provides the requested service and is on the paths from X and Z. Its convergence degree thus becomes 5 ($= 2 + 2 + 1$). Note that the degree is incremented by one since R is adjacent to the path from Y to A.

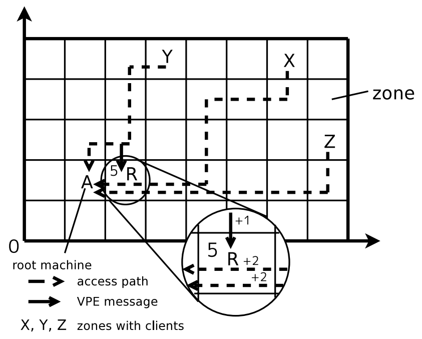


Fig. 6 Virtual demand raising (VDR). The convergence degrees of physical machines adjacent to the servicing machine are incremented by two.

4. Experiments

In this section we report simulation results and analysis. This section shows that EPSS can detect demand fluctuations and select candidate spots for replica servers appropriately. We also exhibit that candidates selected by EPSS have a higher probability of decreasing the amount of traffic than have heuristic strategies.

4.1 Evaluation Methodology

To evaluate the effectiveness of EPSS, we built an ExaPeer prototype with EPSS on Overlay Weaver [12], a toolkit that enables a structured overlay network to be easily constructed and its behavior to be emulated. We tested ExaPeer with EPSS under different load fluctuations conditions on Overlay Weaver, and we measured three key indicators.

- *The number of candidate spots*, which shows how well EPSS selects candidate spots for replica servers on the basis of clients demands. If the number of spots changes with the fluctuations in the demand, we can say that EPSS selects candidate spots on the basis of demand fluctuations.
- *The number of hops between accessing clients and accessed spots*, which shows how well EPSS selects candidate spots that are close to high-demand areas on the overlay network. The fewer the hops, the "logically" closer the spots are to the high-demand areas.
- *The RTT between accessing clients and accessed spots*, which shows how well EPSS selects candidate spots that are close to high-demand areas on the physical network. The lower the RTT, the "physically" closer the spots are to the high-demand areas.

For comparison, we measured the number of hops and RTTs when no replica servers are placed, called *ROOT_ONLY*; that is, only the root machine services all the requests.

To confirm the behavior of EPSS in realistic environments, we emulated the Internet environment by introducing the Internet topology derived from a virtual topology and the Internet measurement data. The Internet measurement data was collected by the King measurement technique [13]. In our experiments, we used two datasets to construct the Internet topologies.

(1) *TS*: A Transit-Stub model of the Internet that has a two-level hierarchy of routing domains, i.e., transit domains interconnect lower-level stub domains [14]. To generate a transit-stub topology, we used the GT-ITM topology generator. To generate the topology, we assigned the parameters as follows: 228 transit domains, 5 transit nodes per transit domain, 4 stub domains attached to each transit node, and 2 nodes in each stub domain. We selected about 3,000 nodes with unique GNP coordinates from about 10,000 nodes.

(2) *Meridian*: The dataset used by the Meridian project [15]. It contains pairwise RTT measurements between 2500 DNS servers whose IP addresses are unique, spanning 6.25 million node pairs. Data was collected on May 5-13, 2004. We selected about 2,200 nodes whose GNP coordinates are unique.

The number of services was one. The number of replica servers on a physical machine was one. The number of physical machines is equal to the number of nodes in each simulation. The dimension of GNP was six.

We set the size of the sliding window to 300 seconds and checked the degree of convergence every 60 seconds. Jung, et al. [2] reported that in some flash crowds the access rate increased in 40 seconds to 15 minutes intervals. So, we set the interval of convergence checking to 60 seconds.

4.2 Changing Demand Scale

EPSS selects candidate spots for replica servers so as to effectively handle

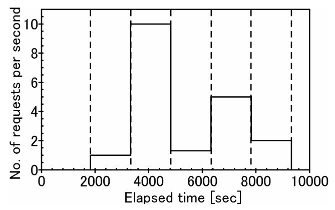


Fig. 7 Frequency of requests.

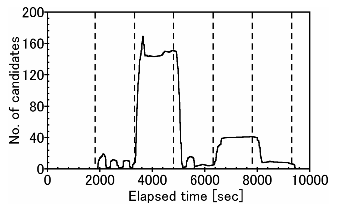


Fig. 8 No. of candidate spots with EPSS.

client demands. EPSS can also find the spots quickly even if the scale of a demand changes frequently. If the scale of a demand increases, EPSS selects more candidate spots, and thus the number of hops and RTTs decrease. If the scale of a demand decreases, EPSS eliminates wasteful candidate spots. Despite the decrease of candidate spots, RTTs are still lower than accessing to the root machine because EPSS maintains the number of candidate spots to efficiently handle the demand.

To evaluate how well it does this, we tested ExaPeer with EPSS when the demand for a service fluctuated over the entire network. To generate the fluctuation, we changed the frequency of accessing the service. **Figure 7** shows the access frequency. We used in the TS model as clients. A terminal node is a node connected to only one other node in the TS topology. A terminal node corresponds to a gateway in a local area network to which clients belong. The number of terminal nodes was 1,815. *Upper* was set to 300, and *lower* was set to 150.

4.2.1 Performance of EPSS Mechanism

Figure 8 shows the number of candidate spots for replica servers. The vertical dashed lines denote the times when the access frequency changed. EPSS dynamically increased/decreased the number of candidates in accordance with the demand fluctuation. When the access frequency increased, EPSS increased the number of candidates. When the access frequency decreased, it reduced the number of candidates. This demonstrates that EPSS quickly adjusts the number of candidates even when the number of client accesses suddenly increase or decrease (compare the number of candidates at 3,200 and 6,400 sec in Fig. 8). For a given demand fluctuation, EPSS took at most 340 sec to adjust the number of candidates. Reducing the checking interval would make EPSS even more sensitive to demand fluctuations, enabling it to select a candidate even more quickly.

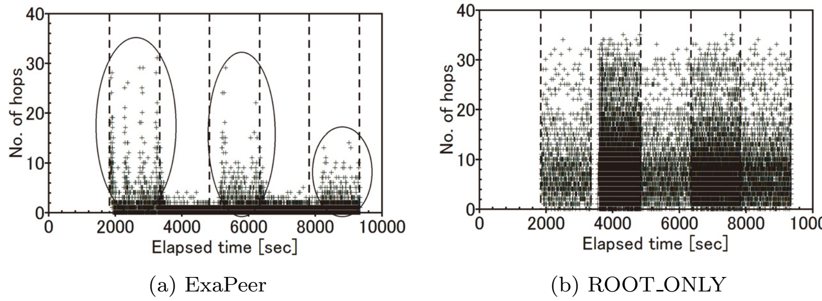


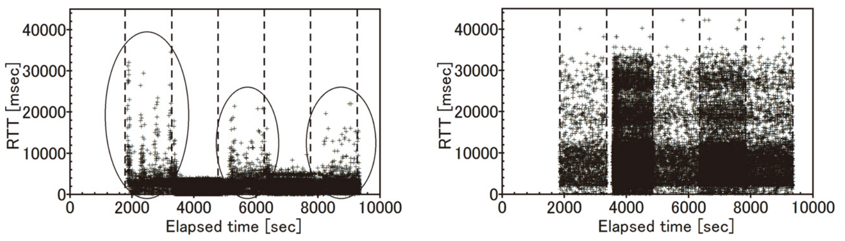
Fig. 9 Comparison of number of hops between EPSS and ROOT_ONLY.

Figure 9 shows the number of hops between clients and accessed candidate spots. Figures 9 (a) and (b) show the number of hops with EPSS and ROOT_ONLY, respectively. The vertical dashed lines denote the times when the access frequency changed. The tendency towards fewer hops suggests that EPSS can find candidate spots on the overlay network near the accessing clients. Compared with Fig. 9 (b), the average number of hops with EPSS is lower than that of ROOT_ONLY. When the access frequency was high, the average number of hops was close to four. When the access frequency was low, the average number was higher (circled regions in Fig. 9 (a)) because EPSS did not increase the number to avoid booting unnecessary replica servers.

Figure 10 shows the RTTs between clients and accessed spots. Figure 10 (a) and (b) show RTTs with EPSS and ROOT_ONLY, respectively. The tendency towards lower RTTs suggests that EPSS can select candidate spots which are close to the accessing clients on the physical network. Compared with Fig. 10 (b), the average RTTs with EPSS is lower than that of ROOT_ONLY. When the access frequency was low, the average RTTs were longer (circled regions in Fig. 10 (a)) because, once again, EPSS did not increase the number to avoid booting unnecessary replica servers.

4.2.2 Comparison with Other Strategies

To determine whether the candidate spots selected by EPSS have a higher



(a) ExaPeer (b) ROOT_ONLY

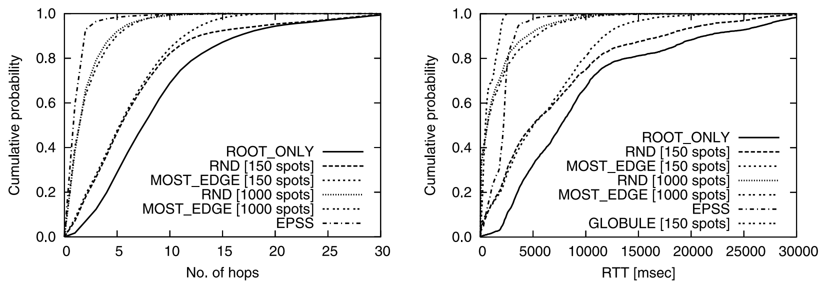
Fig. 10 Comparison of RTT between EPSS and ROOT_ONLY.

probability of decreasing the amount of traffic than other strategies, we compared EPSS with four strategies as mentioned below.

- *ROOT_ONLY*: Only the root machine services all requests.
- *RND*: Candidate spots are chosen randomly from all machines.
- *MOST_EDGE*: Candidates are selected from machines that have many connections to terminal nodes.
- *GLOBULE*: Candidates are selected with the method used by Globule [16)–19)]. Globule deals with long-term fluctuations only because it relies on global information such as access logs on all servers and exact locations of all physical machines. Since Globule selects nearly optimal spots based on global information, EPSS finds candidate spots close to optimal candidates with local information only if RTTs with EPSS are close to those with GLOBULE. Since Globule cannot deal with short-term fluctuations used in our experiment, we calculated the candidate spots in off-line. Note that we cannot measure the number of hops on GLOBULE because Globule uses an overlay network different from ExaPeer.

RND, MOST_EDGE and GLOBULE selected 150 candidate spots because the largest number of candidates selected by EPSS was 150 in this experiment.

Figure 11 shows the cumulative probabilities of the number of hops and the RTTs. The performance of EPSS was better than that of RND, MOST_EDGE and ROOT_ONLY. With EPSS, 90% of all client accesses were relayed to the servicing machine within two hops. With RND and MOST_EDGE, 13 and 12 hops respectively were required to process the client accesses, with ROOT_ONLY, 17 were required. The RTTs with EPSS were less than 3,300 msec for 90% of all client accesses, while those with RND and MOST_EDGE were less than 17,200 and 11,800 msec, respectively. With ROOT_ONLY, the RTTs were less than 21,300 msec.



(a) No. of hops (b) RTT

Fig. 11 Cumulative probabilities of No. of hops and RTT.

We also measured the number of hops and the RTTs when the number of physical machines available to RND and MOST_EDGE for use as candidates was increased from 150 to 1000. As shown in Fig. 11, for 80% or less of all client accesses, the RTTs were better than those with EPSS. This is because the number of candidate spots were 6.7 times more than with EPSS. However, to process 90% of all client accesses, RND and MOST_EDGE required five hops and more RTTs than EPSS.

The performance of candidates selected by EPSS were the closest to GLOBULE. Since the spots selected by GLOBULE are nearly optimal, this result says that EPSS can select good candidates. Note that GLOBULE collects various kinds of information from the entire network and thus is not suitable for short-term fluctuations.

4.3 Moving Localized Demand

EPSS finds candidate spots as close to a high-demand area as possible even if the area is moving. A high-demand area moves, for example, due to time differences around the world. To confirm this feature of EPSS, we demonstrate that EPSS finds candidate spots located around the high-demand areas. To generate a high, localized demand area, we built two types of client clusters on

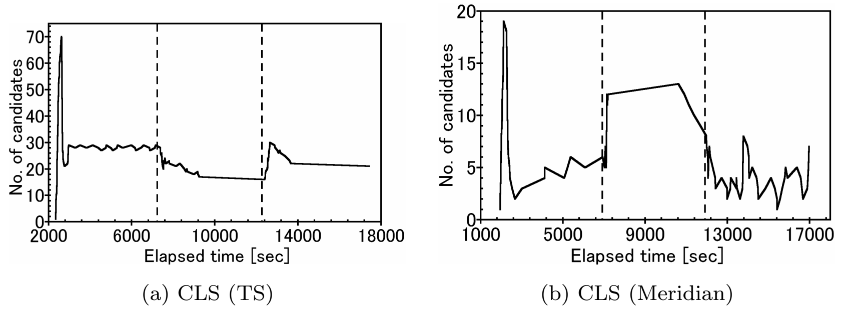


Fig. 12 The number of candidate spots with EPSS when the high-demand area moves.

each topology, based on the following schemes.

- CLS (TS): The machines were selected from terminal nodes in TS. We selected center machines, the RTTs between which are over 5,000 msec. Also, we selected ten machines, including the center machine, for each cluster that had an RTT from the center machine of less than 2,000 msec.
- CLS (Meridian): The machines were selected from all nodes in Meridian. RTTs between center machines are over 250 msec. Also, we selected ten machines, including the center machine, for each cluster that had an RTT from the center machine of less than 100 msec.

The parameters used for creating clusters differ in CLS (TS) and CLS (Meridian) because the average RTT differs in CLS (TS) and CLS (Meridian). Note that the difference between CLS (TS) and CLS (Meridian) is the base's topologies. CLS (TS) is based on the TS model and CLS (Meridian) is based on the Meridian model. We switched the cluster requesting a service to another one every 5,000 sec. *Upper* was set to 300, and *lower* was set to 200.

4.3.1 Performance of EPSS Mechanism

Figure 12 shows the number of candidate spots found by EPSS in this experiment. The vertical dashed lines denote the times when the requesting cluster was switched. EPSS changed the number of candidates when the cluster was switched. The number of candidates tended to increase excessively just immediately after the cluster was switched. This is because EPSS temporarily selected candidates along the client access path. It then quickly eliminated the unneeded

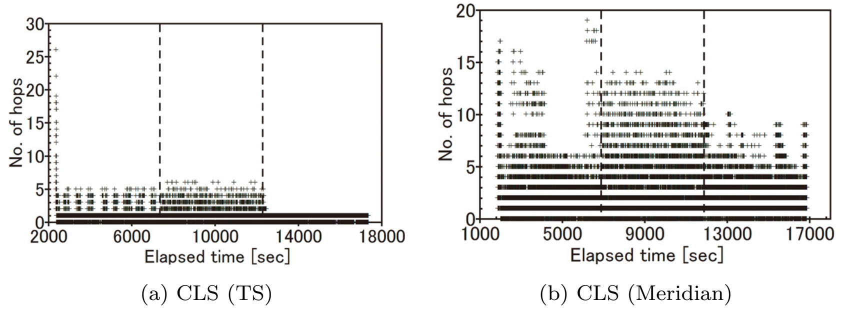


Fig. 13 The number of hops with EPSS when the high-demand area moves.

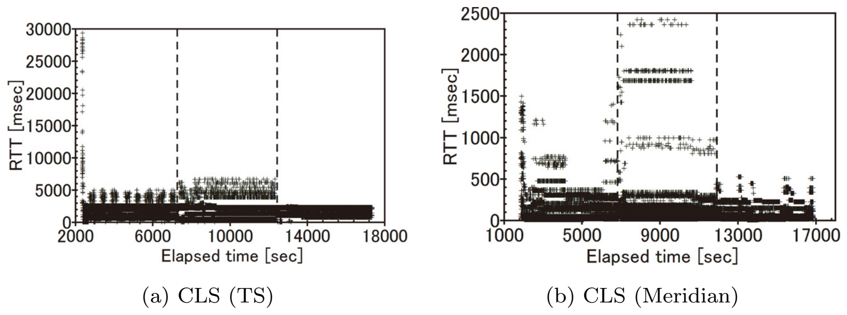


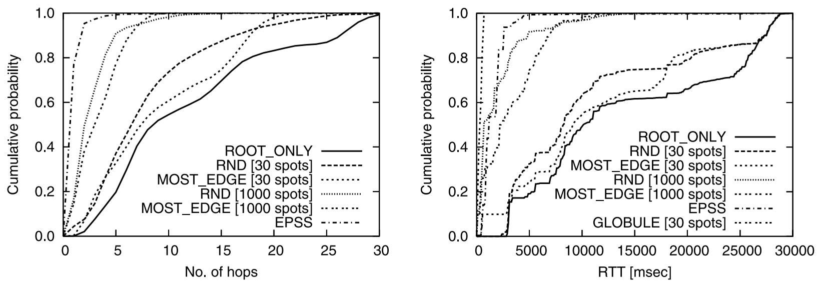
Fig. 14 RTTs with EPSS when the high-demand area moves.

spots far from accessing clients. **Figure 13** shows the number of hops between clients and candidates, and **Fig. 14** shows the RTTs between clients and candidates. EPSS selected candidates closer to the high demand areas.

In CLS (Meridian), the clients in some clusters still got the larger number of hops and the longer RTTs than those in others. This is because the access paths of these clients were anomalously longer than those of others. Since there were long access paths, EPSS took a longer time to improve the location of candidate spots. In fact, the number of hops and RTTs were improved at the end of the second period (see Fig. 13 (b) and Fig. 14 (b)).

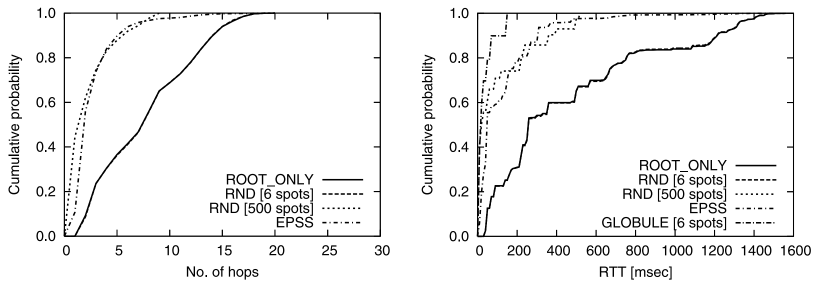
4.3.2 Comparison with Other Strategies

To determine whether the candidate spots on EPSS are closer to the



(a) No. of hops (b) RTT

Fig. 15 Cumulative probabilities of number of hops and RTT in CLS (TS).



(a) No. of hops (b) RTT

Fig. 16 Cumulative probabilities of number of hops and RTT in CLS (Meridian).

high-demand areas than in the case of heuristic strategies, we again used ROOT_ONLY, RND and MOST_EDGE for comparison. RND and MOST_EDGE chose 30 physical machines as candidates since EPSS selected 30 candidates in this experiment. Note that CLS (Meridian) uses RND only because the network topology of Meridian is unknown.

Figure 15 shows the cumulative probability of the number of hops and RTTs in CLS (TS) when the first cluster appeared. EPSS performed better than RND, MOST_EDGE and ROOT_ONLY. With EPSS in CLS (TS), 90% of all client accesses were relayed to the servicing machine within two hops. With RND and MOST_EDGE, 17 and 18 hops were required, respectively. With ROOT_ONLY, 27 hops were required. For 90% of all client accesses, the RTTs with EPSS were less than 2,600 msec, while those with RND and MOST_EDGE were less than 27,300 msec. With ROOT_ONLY, the RTTs were less than 27,400 msec. **Figure 16** shows the cumulative probability of the number of hops and RTTs in CLS (Meridian). EPSS performed better than RND and ROOT_ONLY.

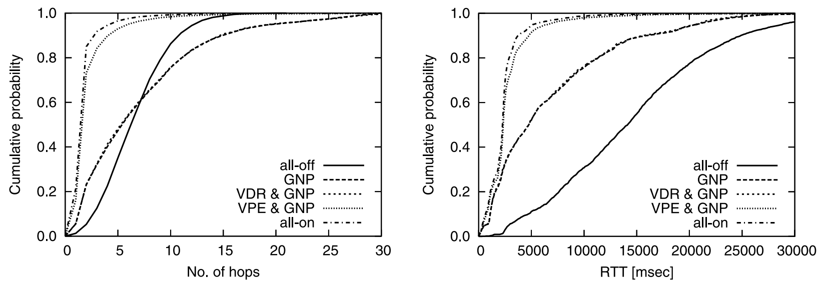
We also measured the number of hops and the RTTs when the number of physical machines which RND and MOST_EDGE could use as candidate spots was increased from 30 to 1,000 in CLS (TS) and from 6 to 500 in CLS (Meridian). The experimental results are shown in Figs. 15 and 16. Even if the number of candidates were increased, RND and MOST_EDGE could not find candidates better than EPSS.

To confirm that EPSS selects good candidates, we compared EPSS with GLOBULE, which selects nearly optimal candidates. In this experiment, EPSS and GLOBULE selected 30 candidates in CLS (TS) and 6 candidates in CLS (Meridian). As shown in Figs. 15 and 16, the performance of EPSS was the closest to GLOBULE among all the strategies. We again mention that GLOBULE is not suitable for short-term fluctuations because it requires the information from the entire network.

4.4 Effectiveness of EPSS Mechanism Components

To determine whether all EPSS mechanism components contribute to selecting candidate spots for replica servers, we prepared five different configurations of the EPSS mechanism: all-off, GNP, VDR & GNP, VPE & GNP, and all-on. In this experiment, a randomly selected terminal node sent a request to a service every 500 msec. We measured the number of hops and RTTs between clients and accessed spots. *Upper* was set to 200, and *lower* was set to 150.

The cumulative probabilities of the number of hops and RTTs are shown in **Fig. 17**. Although the number of hops with all-off was better than the one with GNP and VDR & GNP, the RTTs with all-off were the largest of all these configurations. The main reason for this is that all-off does not take positional relations into consideration because GNP is not used in all-off; adjacent zones in the constructed CAN may be physically far from each other. With GNP, EPSS takes positional relations between physical machines into consideration. Therefore, the RTTs with GNP were better than those with all-off. When VPE was added to EPSS, along with GNP, the RTTs were lower than with GNP



(a) No. of hops (b) RTT

Fig. 17 Comparison of the effectiveness of VPE, VDR, and GNP.

only. We also simulated EPSS with VDR & GNP. The results were very similar to those with GNP. This is because the VDR technique enhances the effect of VPE. VPE facilitates finding access paths from clients, and then VDR facilitates finding the direction in which replica servers should be deployed. Therefore, VDR is useless without VPE. Combining VDR with VPE facilitates finding candidate spots more precisely. When all the features of EPSS were turned on, the number of hops and the RTTs between clients and accessed spots were lower than with GNP & VPE.

5. Discussion

5.1 Scalability

EPSS should be scalable with the number of clients because the communication traffic it adds is negligible. The total communication traffic with EPSS that uses 6-dimensional GNP and CAN, including query messages and EPSS control messages, is no more than $52 * 12(N^{1/6})$ bytes, where N denotes the number of available physical machines. If there are one million machines, the total communication traffic is at most 6,240 bytes per client request. However, if the coordinates calculated by GNP are not distributed uniformly, it may be a bit higher.

The amount of data recorded at each physical machine increases with the number of hosting services. EPSS should be scalable with the number of services because EPSS handles only a simple array of integers for each service. If each physical machine records the number of relayed queries for the last 300 seconds (i.e., the length of the array is 300), the amount of data recorded on each machine is only 1.2 Kbytes for each service.

5.2 Consistency

ExaPeer is designed to primarily support services that provide read-only contents. Unlike typical content distribution networks (CDNs), ExaPeer allows us to provide dynamic contents, which generate the contents dynamically by executing server-side scripting code. This is because ExaPeer enables the entire image of virtual machines to be transferred. This feature of ExaPeer widens its applicability. Since each virtual machine can contain read-only databases, ExaPeer can host a catalog site, for example, for on-line shopping. A catalog site typically consists of dynamic web pages generated by queries from the users. Since the database is not frequently updated, ExaPeer can host a catalog site if combined with techniques for virtual machine migration[20)–22)].

To provide a service on ExaPeer that requires frequent updates to databases, the service designer must divide the database records into two parts: read-intensive and update-intensive ones. The read-intensive part of the database can be incorporated into a virtual machine supported by ExaPeer. The update-intensive part should be located on a special machine to avoid running a complicated protocol for consistency. A load-balancing technique, such as GlobeTP[23)], for database queries should be usable in conjunction with ExaPeer.

5.3 Fairness of Services

EPSS increases the number of candidate spots for a highly demanded service. At first glance, EPSS favors the highly demanded service and degrades the quality of service for lightly demanded services. A closer look reveals that EPSS does not degrade the overall performance of lightly demanded services even if there is an avalanche of accesses to a highly demanded service.

EPSS selects candidate spots from the areas in which there are a large number of clients requesting a specific service. As a result, replica servers are placed on physical machines close to or within the high-demand areas. Thus, the overall load on the Internet is reduced, so the responsiveness of lightly demanded services should improve.

5.4 Security

Unlike typical peer-to-peer (P2P) systems, ExaPeer is assumed to be built on top of trusted physical machines. In normal P2P systems, any user can add a machine to the system without any process of authentication. In ExaPeer, a machine is added to the system only when it is successfully authenticated, for example, by a challenge and response protocol; ExaPeer overlay is closed. Thus, ExaPeer does not have to beware of malicious physical machines. Of course, physical machines may be cracked if ExaPeer is vulnerable in its implementation.

ExaPeer is highly robust against denial of service (DoS) attacks, which are inherently the same as flash crowds[2]. EPSS can handle DoS attacks as well as it can handle flash crowds because it can handle extreme short-term demand fluctuations. However, if an attacker accesses a physical machine directly, ExaPeer is susceptible to a DoS attack. To improve ExaPeer's robustness against DoS attacks, we must incorporate another mechanism, such as dFence[24].

6. Related Work

6.1 Content Distribution Networks

CDNs handle high demand by replication of *content*. Akamai[25] and other commercial CDNs use DNS redirection to reroute client requests to local clusters of physical machines by building detailed maps of the Internet through a combination of BGP feeds and their own measurements. These systems do not control the amount of computing capacity in accordance with demand fluctuations.

Globule[17] is a web server module that replicates documents on the basis of observed access patterns in order to spontaneously create cache copies on peer servers close to clients. It experiences difficulties to respond to short-term demand fluctuations because it uses a simulation to decide where to place cache copies[18].

CoralCDN[26] is a peer-to-peer content distribution network that relies on a hierarchical structure, called "distributed sloppy hashtables," to reduce the load on web servers. It has trouble detecting certain types of demand distributions, such as far and wide but low density demand. To detect such distributions, EPSS mechanism uses VPE and VDR.

Basically, ExaPeer has more power than CDNs. CDNs mainly target a web service, while ExaPeer can provide other services because it repositions replica servers including all of the server system. ExaPeer enables provision of dynamically generated content because a complete image of virtual machines can be distributed. In addition, a service provider can configure its own image of virtual machines; the provider can select operating systems, libraries, and applications truly.

Amazon.com recently released Amazon Elastic Compute Cloud (Amazon EC2) service[1]. The purpose of the service is similar to that of ExaPeer. When a user calls the provided API manually, Amazon EC2 increases or decreases the number of replica servers in a local cluster of physical machines. Amazon EC2 does not reposition the location of replica servers on the Internet. Without any manual operations, EPSS dynamically adjusts the number and the locations of replica servers.

6.2 Server Selection

From the viewpoint of clients, ExaPeer is an anycast service. IP anycast is a network-level solution to a server selection[27]–[29]. However, IP anycast is not widely used or available. ExaPeer can serve as a flexible, demand-distribution-based server-selection mechanism for clients.

Akamai also supports a mechanism for selecting a server. Akamai traceroutes the IP address space from multiple vantage points to detect the route convergence and pings the common router found by the traceroute to select a server close to clients[30]. Unlike ExaPeer, Akamai does not handle demand fluctuations.

Meridian[15] (used for DNS redirection by ClosestNode.com[31]) creates an overlay network with neighbors chosen from a particular distribution; routing to closer nodes is guaranteed to find a minimum RTT given a growth-restricted metric space[32]. EPSS does not need on-demand probes to select candidate spots.

OASIS[33] is a globally distributed anycast system that enables legacy clients to find nearby or unloaded replica servers for distributed services. OASIS provides only an anycast service; it does not adjust the computing capacity for each service.

7. Conclusion

EPSS selects candidate spots for replica servers to handle short-term demand fluctuations (such as flash crowds) as well as long-term fluctuations. From the degree of access path convergence, EPSS detects demand fluctuations and adjusts the number and the locations of candidate spots for replica servers. Even if the demand for a service increases in the short term, EPSS quickly responds and selects candidates for replica servers. Simulation results demonstrated that the candidate spots with EPSS work better than the ones selected by heuristic methods no matter how much the scale of the demand changes.

The next step in this research is twofold. First, we have to develop a technique for repositioning replica servers in accordance with physical machines loads. Second, we plan to evaluate EPSS on the Internet using PlanetLab[34].

Acknowledgments This work was partially supported by MEXT Grant-in-Aid for Scientific Research on Priority Areas (New IT Infrastructure for the Information-explosion Era).

References

1) Amazon: Amazon Elastic Compute Cloud. http://aws.amazon.com/ec2.
2) Jung, J., Krishnamurthy, B. and Rabinovich, M.: Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites, *Proc. ACM Int'l Conf. on World Wide Web*, pp.293–304 (2002).
3) Lorenz, S.: Is your Web site ready for the flash crowd? (2000). http://www.serverworldmagazine.com/sunserver/2000/11/flash.shtml.
4) Slashdot: Slashdot FAQ. http://slashdot.org/faq/.
5) Wikipedia, the free encyclopedia: Slashdot effect. http://en.wikipedia.org/wiki/Slashdot_effect.
6) Eugene Ng, T.S. and Zhang, H.: Predicting Internet Network Distance with Coordinates-Based Approaches, *Proc. IEEE Infocom*, pp.170–179 (2002).
7) Dabek, F., Cox, R., Kaashoek, F. and Morris, R.: Vivaldi: A Decentralized Network Coordinate System, *Proc. ACM SIGCOMM*, pp.15–26 (2004).
8) Ratnasamy, S., Francis, P., Handley, M., Karp, R.M. and Shenker, S.: A Scalable Content-Addressable Network, *Proc. ACM SIGCOMM*, pp.161–172 (2001).
9) Szymaniak, M., Pierre, G. and van Steen, M.: Scalable Cooperative Latency Estimation, *Proc. IEEE Int'l Conf. on Parallel and Distributed Systems*, pp.367–376 (2004).
10) Xu, Z., Tang, C. and Zhang, Z.: Building Topology-Aware Overlays Using Global Soft-Stat, *Proc. IEEE Int'l Conf. on Distributed Computing Systems*, pp.500–508 (2003).
11) Goldberg., R.P.: Survey of Virtual Machine Research, *IEEE Computer Magazine*, Vol.7, No.6, pp.34–45 (1974).
12) Shudo, K., Tanaka, Y. and Sekiguchi, S.: Overlay Weaver: An Overlay Construction Toolkit, *IPSJ Trans. Advanced Computing Systems*, Vol.46, No.SIG 4 (ACS 9), pp.33–44 (2005).
13) Gummadi, P.K., Saroiu, S. and Gribble, S.D.: King: estimating latency between arbitrary internet end hosts, *Proc. ACM SIGCOMM Internet Measurement Workshop*, pp.5–18 (2002).
14) Zegura, E.W., Calvert, K.L. and Bhattacharjee, S.: How to Model an Internetwork, *Proc. IEEE Infocom*, pp.594–602 (1996).
15) Wong, B., Slivkins, A. and Sirer, E.G.: Meridian: A lightweight network location service without virtual coordinates, *Proc. ACM SIGCOMM*, pp.85–96 (2005).
16) Pierre, G., van Steen, M. and Tanenbaum, A.S.: Dynamically Selecting Optimal Distribution Strategies for Web Documents, *IEEE Trans. Computers*, Vol.51, No.6, pp.637–651 (2002).
17) Pierre, G. and van Steen, M.: Globule: A Platform for Self-Replicating Web Documents, *Proc. IEEE Int'l Conf. on Protocols for Multimedia Systems*, Lecture Notes in Computer Science, Vol.2213, Springer, pp.1–11 (2001).
18) Szymaniak, M., Pierre, G. and van Steen, M.: Latency-Driven Replica Placement, *Proc. IEEE Symp. on Applications and the Internet*, pp.399–405 (2005).
19) Pierre, G. and van Steen, M.: Globule: a Collaborative Content Delivery Network, *IEEE Communications Magazine*, Vol.44, No.8, pp.127–133 (2006).
20) Clark, C., Fraser, K., Hand, S., Hansen, J.G., Jul, E., Limpach, C., Pratt, I. and Warfield, A.: Live Migration of Virtual Machines, *Proc. USENIX Symp. on Networked Systems Design and Implementation*, pp.273–286 (2005).
21) Sapuntzakis, C.P., Chandra, R., Pfaff, B., Chow, J., Lam, M.S. and Rosenblum, M.: Optimizing the Migration of Virtual Computers, *Proc. USENIX Symp. on Operating Systems Design and Implementation* (2002).
22) VMware Inc.: Building Virtual Infrastructure with VMware VirtualCenter, White paper. http://www.vmware.com/pdf/vi_wp.pdf.
23) Groothuyse, T., Sivasubramanian, S. and Pierre, G.: GlobeTP: Template-Based Database Replication for Scalable Web Applications, *Proc. ACM Int'l Conf. on World Wide Web*, pp.301–310 (2007).
24) Mahimkar, A., Dange, J., Shmatikov, V., Vin, H. and Zhang, Y.: dFence: Transparent Network-based Denial of Service Mitigation, *Proc. USENIX Symp. on Networked Systems Design and Implementation*, pp.327–340 (2007).
25) Dilley, J., Maggs, B.M., Parikh, J., Prokop, H., Sitaraman, R.K. and Weihl, W.E.: Globally Distributed Content Delivery, *IEEE Internet Computing*, Vol.6, No.5, pp. 50–58 (2002).
26) Freedman, M.J., Freudenthal, E. and Mazières, D.: Democratizing Content Publication with Coral, *Proc. USENIX Symp. on Networked Systems Design and Implementation*, pp.239–252 (2004).
27) Ballani, H. and Francis, P.: Towards a global IP anycast service, *Proc. ACM SIGCOMM*, pp.301–312 (2005).
28) Katabi, D. and Wroclawski, J.: A framework for scalable global IP-anycast (GIA),

Proc. ACM SIGCOMM, pp.3–15 (2000).
29) Partridge, C., Mendez, T. and Milliken, W.: Host Anycasting Service, RFC 1546 (1993).
30) Chen, Y., Lim, K.H., Katz, R.H. and Overton, C.: On the stability of network distance estimation, *ACM SIGMETRICS Performance Evaluation Review*, Vol.30, No.2, pp.21–30 (2002).
31) Wong, B. and Sirer, E.G.: ClosestNode.com: an open access, scalable, shared geocast service for distributed systems, *ACM Operating Systems Review*, Vol.40, No.1, pp.62–64 (2006).
32) Karger, D.R. and Ruhl, M.: Finding nearest neighbors in growth-restricted metrics, *Proc. ACM Symp. on Theory of Computing*, pp.741–750 (2002).
33) Freedman, M.J., Lakshminarayanan, K. and Mazières, D.: OASIS: Anycast for Any Service, *Proc. USENIX Symp. on Networked Systems Design and Implementation*, pp.129–142 (2006).
34) Bavier, A.C., Bowman, M., Chun, B.N., Culler, D.E., Karlin, S., Muir, S., Peterson, L.L., Roscoe, T., Spalink, T. and Wawrzoniak, M.: Operating Systems Support for Planetary-Scale Network Services, *Proc. USENIX Symp. on Networked Systems Design and Implementation*, pp.253–266 (2004).

(Received October 9, 2007)
(Accepted March 20, 2008)

Masato Asahara was born in 1982. He received his B.E. degree from the University of Electro-communications in 2005 and M.E. degree from Keio University in 2007, respectively. He is currently a Ph.D. candidate in School of Science for Open and Environmental Systems at Keio University, since 2007. His research focuses on large-scale peer-to-peer systems for repositioning replica servers in the Internet. He is a member of the IPSJ, IEEE/CS, ACM and USENIX.

Akio Shimada was born in 1983. He received his B.E. and M.E. degrees from Keio University in 2006 and 2008, respectively. Since 2008 he is working for Systems Development Laboratory, Hitachi, Ltd. His research interests include overlay networks and network simulators. He is a member of the IPSJ.

Hiroshi Yamada was born in 1981. He received his B.E. and M.E. degrees from the University of Electro-communications in 2004 and 2006, respectively. He is currently a Ph.D. candidate in School of Science for Open and Environmental Systems at Keio University, since 2006. His research interests include virtual machine technology, operating systems, and system software. He is a member of the IPSJ, ACM, USENIX and IEEE/CS.

Kenji Kono received the BSc degree in 1993, MSc degree in 1995, and Ph.D. degree in 2000, all in computer science from the University of Tokyo. He is an associate professor of the Department of Information and Computer Science at Keio University. His research interests include operating systems, system software, and Internet security. He is a member of the IEEE/CS, ACM and USENIX.