Vol.2010-MBL-52 No.15
2010/1/29

An Implementation of a Framework for Integrating Churn Managements among Structured Overlay Networks

Kimihiro Mizutani,[†1] Satoshi Matsuura,[†1,∗1]
Shinichi Doi,[†1] Kazutoshi Fujikawa [†1,∗2]
and Hideki Sunahara[†1,∗3]

In this paper, we propose a framework which can share states of the nodes and spread churn information to neighbor nodes among different overlay networks. Using our framework, we achieve maintaining each overlay network effectively and improve the reachability of messages under churn. We evaluate our framework with Chord, Kademlia, and Pastry. Our framework improves by approximately 30% in the detection speed of churn and decreases the stabilization messages by approximately 30-40%. We discuss the relationship between improvement of detection time and the reachability of messages.

1. Introduction

Structured overlay networks have many famous service primitives and are applied to many peer-to-peer (P2P) applications. For example, distributed hash tables (DHT)[1)], decentralized object location and routing (DOLR)[2)] and anycast and multicast (CAST)[3)] are famous primitives. The popularity of P2P applications ranges from file sharing to conferencing and content distribution. When a user starts a P2P application, the user's node joins an overlay network. The node contributes some resources while making use of the resources provided by others. The node leaves an overlay network when the user exits the application.

Churn is the action of a node joining or leaving an overlay network. If churn occurs frequently, an overlay network will not maintain its structure and service availability. Consistently, a user of a P2P application cannot get target content. Churn Handling is one of the issues in implementing an overlay network. In typical churn handling, a source node sends a stabilization message to some target node and confirms whether the target node is alive or dead, by checking its response. When the target node does not reply to a message within timeout, the source node can detect the target node leaving. That is, the detection time of churn depends on the interval of stabilization messages.

These days, a lot of overlay networks work concurrently under high churn rates on a real network. In such a situation, the overlay networks cannot maintain its structure because there is a limitation on churn handling by a single overlay network. To improve service availability of overlay networks, we focus attention on mechanisms of cooperation among overlay networks.

Mechanisms of cooperation among overlay networks are used widely to improve the quality of P2P services[4)]. A traffic management mechanism is a famous example[5),6)]. The management mechanism provides effective bandwidth management and loss assurance for overlay networks. As a result, the mechanism enables overlay networks to share network resources effectively and prevents network traffic from concentrating at a particular node. However, many mechanisms of cooperation among overlay networks do not consider improving service availability of overlay networks against a churn situation. This is because each overlay network has its own message protocol and ID-space for managing churn information. For these reasons, there are many redundancies of churn detection among different overlay networks.

In this paper, we propose a new framework integrating ID-space and churn handling functions among overlay networks for eliminating the churn detection redundancies. Our framework introduces *Identifier Space Manager* to general overlay networks. *Identifier Space Manager* links each ID-space to an Internet Protocol (IP) address as a common ID and provides functions of churn handling for overlay networks as common functions. Such functions can share states of the nodes among different overlay networks. As a result, our framework can help overlay networks cooperate efficiently in churn detection and improve their service availability.

The rest of this paper is organized as follows. In section 2, we explain our proposed framework, which has mechanisms to share states of the nodes and spread

†1 Graduate School of Information Science, Nara Institute of Science and Technology
∗1 Presently with National Institute of Information and Communications Technology
∗2 Presently with Graduate School of Information Science, Osaka University
∗3 Presently with Graduate School of Media Design, Keio University

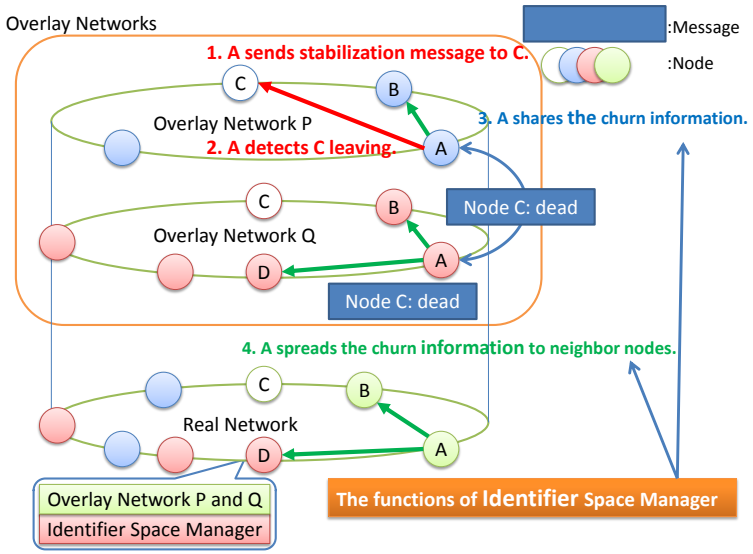

Fig. 1 Our framework overview

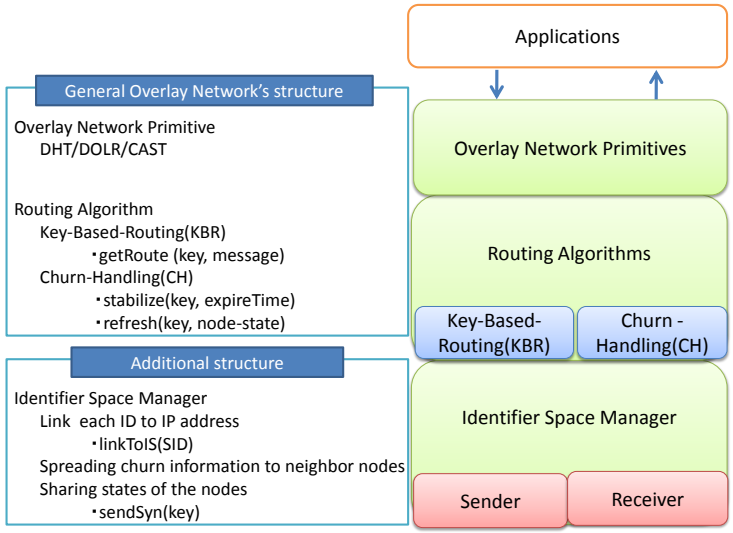

Fig. 2 Framework functions

churn information to neighbor nodes. In section 3, we evaluate our framework using Chord[7], Kademlia[8], and Pastry[9]. Section 4 discusses the effectiveness of our framework. Section 5 provides an overview of several overlay network frameworks and those sharing and spreading mechanisms. Section 6 concludes the paper.

2. Framework for managing churn information

In this section, we explain the proposed framework linking ID-spaces of overlay networks and managing states of the nodes. Our framework introduces the *Identifier Space Manager* to general overlay networks. Different overlay networks cooperate and communicate through the *Identifier Space Manager*.

The *Identifier Space Manager* aims to eliminate redundancies of churn detecting operation and improve churn detection effectively with common churn handling functions. Each overlay network has its own ID-space, resources and routing information. The *Identifier Space Manager* is used for sharing information about overlay networks and provides efficient interaction among different overlay networks. Figure 1 illustrates an overview of our framework.

1 When *Node-A* sends a stabilization message to *Node-C* in *overlay network-P*, *Node-C* does not reply to *Node-A*.
2 *Node-A* detects *Node-C* leaving.
3 *Node-A* shares the churn information with *overlay network-Q*.
4 *Node-A* spreads the churn information to neighbor nodes *Node-B* and *Node-D* on each overlay network

The *Identifier Space Manager* supports the processes **1,2** and **3** by a function called "Sharing states of the nodes". Moreover, the *Identifier Space Manager* supports processes **1,2** and **4** by a function called "Spreading churn information to neighbor nodes". The *Identifier Space Manager* provides these functions for overlay networks as common functions. These functions can eliminate redundancies in detecting churn operation and improve churn detection effectively.

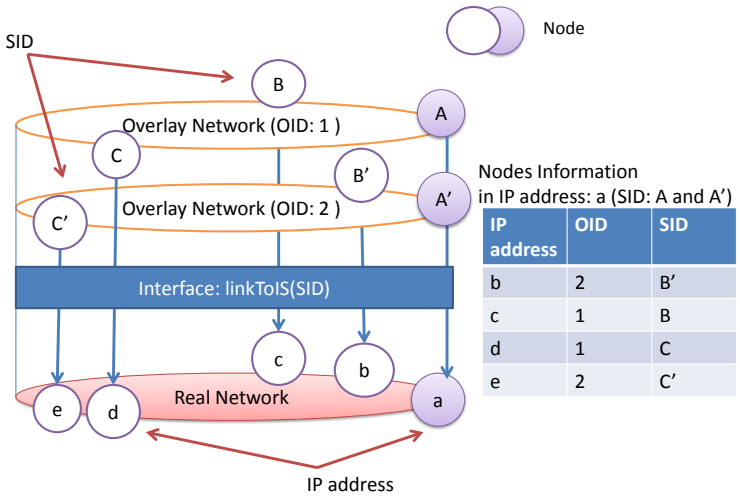

Fig. 3 Node-ID management

2.1 Details of our framework

Figure 2 illustrates our framework functions. Our framework can be roughly divided into three categories: *Overlay Network Primitive*, *Routing Algorithm*, and *Identifier Space Manager*.

A general overlay network can be composed of *Overlay Network Primitive* and *Routing Algorithm*. *Overlay Network Primitive* achieves basic overlay network services: DHT, CAST, and/or DOLR. These services have common operations such as **GET** and **PUT**. For example, DHT provides a function to find a node responsible for a certain piece of data with **GET**. However, these operations are abstracted from *Routing Algorithm* operations. The *Routing Algorithm* has two functions, Key-Based-Routing (KBR)[10] and Churn-Handling (CH). KBR provides a method to find the nearest host related with paticular data, according to some defined metric of hop counts[10]. CH provides a stabilization function that confirms whether a node is alive or dead. The *Overlay Network Primitive* calls *Routing Algorithm* functions and executes the operations. This hierarchical structure has broad utility and can be applied to a wide range of applications.

The *Identifier Space Manager* provides three functions: **linkToIS(SID)**, "Sharing states of the nodes," and "Spreading churn information to neighbor nodes." **LinkToIS(SID)** API achieves a function of linking an SID to an IP address. An SID is an ID that is specific to an overlay network and is assigned to manage the ID-space of the overlay network. The *Identifier Space Manager* manages information of the nodes that contains an IP address and SID. When a node specifies an SID to send a message to a certain node, the *Identifier Space Manager* of the node converts the SID to an IP address and sends the message by using the converted IP address. This function can link SIDs of the nodes to IP addresses. Figure 3 illustrates how to manage each SID in the *Identifier Space Manager*. *Node-A* joins two overlay networks whose overlay network IDs (OIDs) are 1 and 2. *Node-A* links two SIDs: A and A' with their IP address by using **linkToIS(SID)**. Each node links its SIDs into an IP address in the same way. Then *Node-A* has the IP address, OID and SID of the other node on the routing table. By linking each SID to an IP address, we aim to resolve compatibility issues among different overlay networks.

2.2 Mechanism for sharing churn information

When a source node sends a stabilization message to a target node, the source node uses **Stabilization (key, expire-Time)** API. The *key* is a node SID. Then, an operation of the API sends a *Syn* message through the *Identifier Space Manager* and waits for the *expire-Time*. Receiving the message, the target node replies with an *Ack* message to the source node. Receiving the *Ack* message from the target node, the *Identifier Space Manager* of the source node finds SIDs that belong to the IP address of the target node and calls **refresh(SID, node-state)** for each SID. *Node-state* indicates alive or dead. If the source node receives an *Ack* message, the target node is alive. Otherwise the target node is dead. A source node can share states of the target nodes through the *Identifier Space Manager* among different overlay networks and reuse states of the nodes that are confirmed by the source node. This process can improve the detection time of churn and reuse states of the nodes.

Figure 4 illustrates the mechanism of sharing node states. In this figure, *Node-C* sends stabilization messages to *Node-A* and *Node-B* in each overlay network at

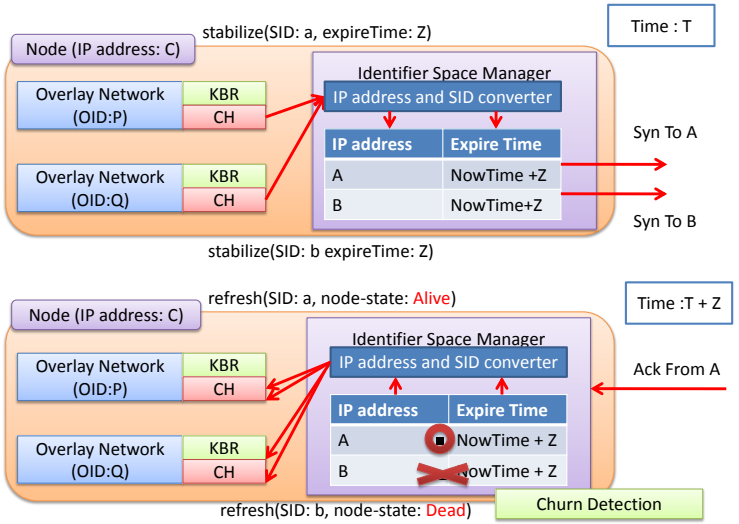

Fig. 4 Sharing states of the nodes

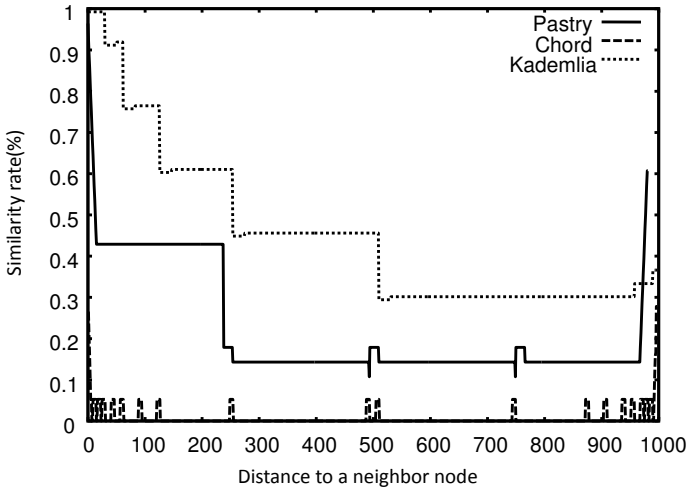

Fig. 5 Similarity of neighbor's routing table

the same time with *expire-Time: Z*. After *Z* seconds, *Node-C* receives an ACK message from *Node-A*. Then, the *Identifier Space Manager* of *Node-C* finds out the states which show *Node-A* is alive and *Node-B* is dead. The *Identifier Space Manager* conveys these nodes' states to overlay networks, which manage these states of the nodes by calling **refresh(SID, node-state)**. This function can share states of the nodes between overlay network (OID: P) and overlay network (OID: Q).

2.3 Mechanism for spreading churn information to neighbor nodes

Our proposed mechanism for spreading churn information considers similarity of routing table among nodes[7]. The proposal mentioned that a routing table of a node is similar with routing table of

the neighbor nodes. Figure 5 illustrates a similarity of a routing table for metric distances. In Chord, a node 's routing table is similar to neighbor nodes by approximately 20%. In Kademlia and Pastry, the similarity is by approximately 90-99%. By these features, a node 's neighbor probably reuses the information which the node detects churns.

However, the mechanism cannot reuse churn information effectively. This is because the mechanism does not consider network latency of spreading churn information. When a node detects a churn and spreads the churn information to neighbor nodes, the neighbor nodes probably have detected the churn by network latency. In this situation, spread churn information is not reused effectively.

In this paper, we implement our mechanism into the *Identifier Space manager*. Our proposal mechanism can spread the churn information effectively by considering similarity of neighbor nodes and RTT(Round Trip Time) between a node and the node neighbor nodes. For considering similarity of routing table and RTT, the *Identifier Space manager* records an access-log which has accessed IP addresses, the count of accesses and RTT for an accessed node. When a node detects a churn, the *Identifier Space manager* of the node selects some nodes and spreads churn information to the nodes. To select the nodes, the *Identifier Space manager* estimates the frequency of stabilization in a certain short interval.

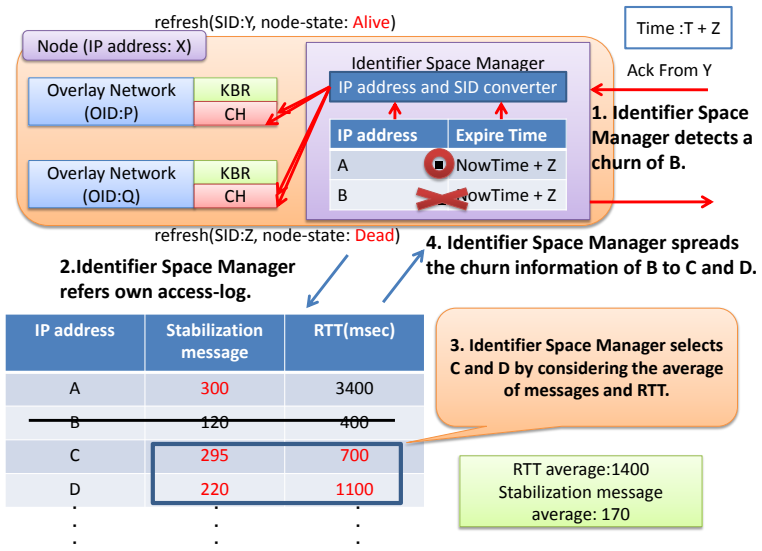

Fig. 6 Spreading churn information to neighbor nodes

When the frequency is higher than average of total stabilization frequency, the *Identifier Space manager* realizes the stabilized node as a neighbor node. Moreover the *Identifier Space manager* selects nodes from the neighbor nodes which have shorter RTT than the average and spreads the churn information for the neighbor nodes.

Figure 6 illustrates a mechanism of spreading churn information to neighbor nodes. A process of our mechanism is as following.

1 *Identifier Space Manager* detects a churn of *Node-B*.
2 *Identifier Space Manager* refers own access-log.
3 *Identifier Space Manager* selects neighbor nodes whose RTT are shorter than the average of RTT.
4 *Identifier Space Manager* spreads the churn information to neighbor nodes *Node-C* and *Node-D* on all overlay network.

The neighbor nodes need not confirm a state of *Node-C* because of sharing churn information. This is because a node can spread quickly by considering RTT. Our mechanism can improve detection time of churn effectively.

3. Evaluation

In this section, we evaluate our framework, and estimate the detection time of churns and sharing stabilization messages. In the experiments, we assume an environment in which churn will occur frequently. We implement three structured overlay networks, Chord, Kademlia, and Pastry, by using our framework. Structured overlay networks conform to a specific graph structure that allows them to locate objects by exchanging *O(logN)* messages where *N* nodes exist in each overlay network. We describe the characteristics of each overlay network.

Chord

On a Chord network, each node manages *finger-table* and *successor-list*. *Finger-table* has node addresses that are determined by the distance from its node. The distance is calculated away from a node by the n-th power in circular ID-space. *Successor-list* has neighbor node addresses and a capacity for binary logarithm of the total nodes.

Kademlia

Kademlia has *k-buckets*. *K-buckets* has node addresses by matching k-bits (k:natural number) with a node, using XOR calculation. Each bucket can store up to 20 node addresses.

Pastry

Pastry has *prefix-map* and *leafset*. *Prefix-map* forms *address-blocks*. Each *address-block* stores up to 16 node IDs. To form the *address blocks*, the node ID is divided into digits with each digit being 4 bits long. When a node ID matches 4k (k:natural number) bit with some other node ID, the node stores the other node ID in k *address block*. *Leafset* stores neighbor node addresses of a node in a circular ID-space, moving clockwise/counter-clockwise. We don't use *neighborset* in our experiment.

All experiments are performed on an emulator. The execution environment of the experiments is as follows; CPU: Core2Duo TS100 2.1GHz, MEMORY: 2GB, OS: Windows Vista, and PROGRAM LANGUAGE: Java SE Development Kit 6. The emulator can control joining and leaving of nodes and manage a lot of scenarios consisting of tasks defined by users. The emulator provides an ideal

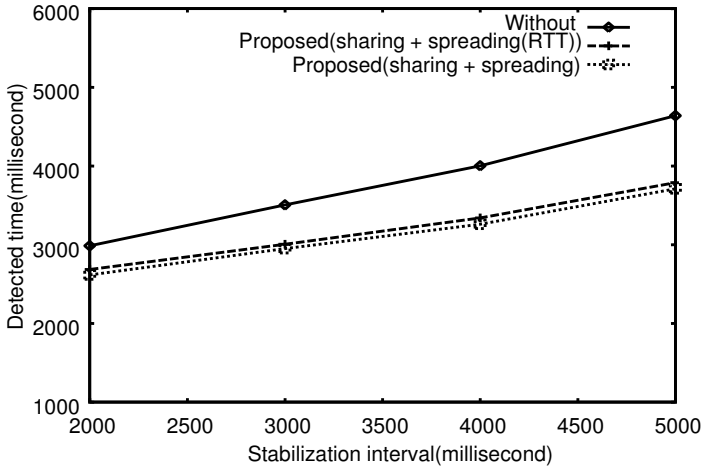

Fig. 7 Detection time of churn

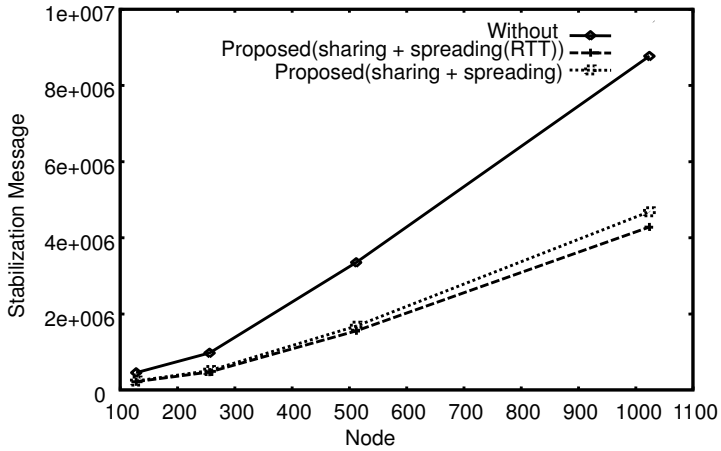

Fig. 8 Stabilization messages for detecting churn

communication environment. In this environment, nodes communicate with each other through UDP and these communications are measured by this emulator. Packet loss don't occur because all communications are generated inside the emulator.

3.1 Impacts of our framework

The purpose of sharing states of the nodes and spreading churn information are to improve detection time of churn and decrease the number of messages. An evaluation scenario is as follows.

(1) The emulator creates N nodes and these nodes join all overlay networks.
(2) Each node generates 1,000 keys that have certain random SIDs as content ID. Each node constantly puts a key in a node that has the responsibility to manage the key on the overlay networks.
(3) Each node constantly gets keys every 1,000 milliseconds.

A node executes GET and PUT functions randomly. In the experimental scenario, when a node joins an overlay network, another node leaves. In other words, the total number of nodes is constant. This evaluation method is proposed by[11]. The frequency of churn is per one second and all nodes have certain RTT(Round Trip Time). RTT is determined by Euclidean distance between a pair of nodes coordination which is assigned by the emulator. The range of RTT is 0-1,000 milliseconds. A time-out period of communication between nodes is defined as 10,000 milliseconds.

At first, we measure improvement of the detection time with our framework. In this experiment, we set the stabilization interval as 2,000, 3,000, 4,000 and 5,000 milliseconds. Figure 7 illustrates the result of the detection time of churn.

At each interval, the shortest detection time is 2,000 milliseconds. The longest is about each interval plus 2,000 milliseconds. The result shows the detection time is improved approximately 30% regardless of the stabilization intervals.

Second, we measure how many stabilization messages are reused. In the experiment, we configure the stabilization interval in each overlay network. In Chord, we configure the interval of *successor-list* as 2,000 milliseconds and the interval

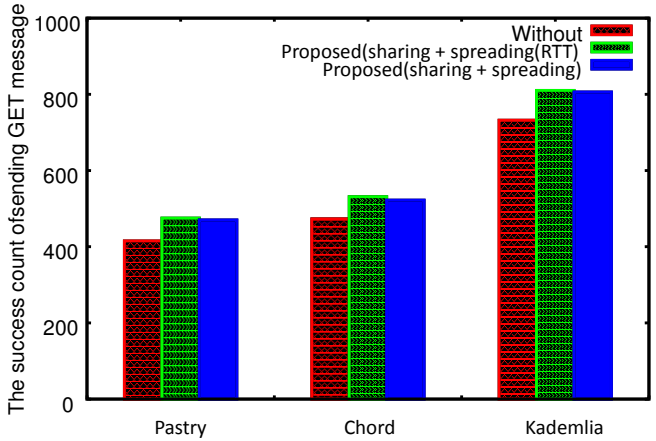

Fig. 9 The success counts of sending GET message

of *finger-table* as 4,000 milliseconds. In Kademlia, stabilization processes are not executed because of adopting a hard state management. In Pastry, we configure the interval of *leafset* as 2,000 milliseconds and the interval of *prefix-map* as 4,000 milliseconds. Figure 8 illustrates the result, which shows stabilization messages are decreased approximately 30-40% while maintaining the service availability, regardless of the number of nodes.

4. Discussion

We discuss improving service availability. To evaluate the improvement of service availability, we execute additional evaluation. In the additional evaluation, we measure success rate of getting a contents in overlay networks. Basically, a environment of additional evaluation is similar with the environment of Section 3. An evaluation scenario is as the following.

(1) The emulator creates 1000 nodes and these nodes join all overlay networks.
(2) A node generates keys which have certain random SIDs as contents ID. The node puts the keys in each node which has the responsibility to manage the key on the overlay networks.
(3) The emulator selects 10 nodes randomly and the selected nodes get keys.

This scenario is repeated 100 times and we measure the average of response time in each overlay network. A root node which manages the contents ID does not leave in overlay networks until the contents are got. An interval of a node leaving is 1000 milliseconds and churn occurs constantly after contents were put. Figure 9illustrates the success counts of sending GET message in each overlay network. In Pastry, the success rate of sending GET message is improved by approximately 14% with our framework. In Chord and Kademlia, the success rate is also improved by approximately 12% and 10% with our framework. There is not so much of a difference between our proposed two mechanisms. Moreover, the mechanism of considering RTT can decrease the number of stabilization message and improve the maintenance costs of the overlay networks under churn situation.

5. Related work

Many researchers have already studied how to integrate operations of overlay networks.

The first approach is defining common functions of overlay networks with APIs and frameworks[4]. This work proposes cooperative mechanisms for managing overlay networks. In the proposal, the mechanisms provide efficient and secure interactions among nodes, by controlling the scope of spreading various messages. The mechanisms can separate overlay networks from their own ID-space, resources and message routing information. A special overlay is used for sharing routing information among different overlay networks. A lot of the researchers propose only mechanism designs. The same can be said for churn management among different overlay networks. It is difficult to implement the management mechanisms for overlay networks because an overlay network has its own ID-space and it is difficult to link different ID-spaces to a single ID-space.

Our framework implements the ID-space management mechanisms. Moreover, our framework provides a function of sharing states of the nodes with IP addresses in *Identifier Space Manager*, which can manage different overlay network IDs. In our framework, different overlay networks can cooperate and share churn

information to improve the service availability of their applications.

The second approach is managing overlay networks by using new protocols and network providers[12]. The protocol aims to reduce network traffic by using network topology data. The protocol selects peers intelligently at random and achieves efficient routing. In this approach, it is easy to achieve consensus comparatively so that ISPs provide APIs. However, there is a possibility that the APIs regulate the operation of overlay networks. In addition, a node must execute the various operations through a tracker server. When the server implements churn handling APIs, all nodes must confirm churn information through the server. Then the flood of stabilization messages must concentrate on the server.

Our framework can manage the churn information decentrally. We implement a mechanism of spreading churn information to neighbor nodes autonomously. This is because the neighbor nodes can reuse states of the nodes. Then our framework prevents stabilization messages from concentrating at a node and improves the maintenance costs of the complex networks under a churn situation.

6. Conclusion

Under a churn situation, it is difficult for a structured overlay network to maintain its structure and service availability. In this paper, we aimed to improve service availability. In concrete terms, we proposed a framework that shares states of the nodes and spreads churn information to neighbor nodes among different overlay networks. Our framework can improve the detection time of churn by about 30% and decrease the stabilization messages by approximately 30-40%. Moreover, We discussed the effect of our framework and recognized the improvement of success rate of sending GET message by approximately 10%-14% by using our framework. These results show our framework can reuse states of the nodes and improve the availability of overlay networks.

Acknowledgment

This work was supported in part by Research and Development Program of 'Ubiquitous Service Platform' (2009), The Ministry of Internal Affairs and Communications, Japan and 'MITOH Program' (2008), Information-Technology Promotion Agency, Japan.

References

1) Dabek.F,Kaashoek.M.F,Karger.D.Morirs.R, and Stoika. I, "wIde-area cooperative storage with CFS," inProc. ACM Symposium on Operating Systems Principles 2001, pp.202 - 215(2001).
2) Hildrum.K,Kubiatowcz.J,D,Rao. S, and Zhao.B.Y, "Distributed object location in a dynamic network," inProc. ACM Symposium on Parallelism in Algorithms and Architectures 2002, pp.41-52(2002).
3) Castro.M,Druschel.P,Keramarrec.A.M, and Rowstron.A, "SCRIBE: A large-scale and decentralized application level multicast infrastructure," IEEE Journal on Selected Areas in Communications 2002, vol.20, issue.20, pp.1489-1499(2002).
4) Koskela. T, Harjula. E, and Ylianttila. M, "Mechanism for Peer-to-Peer Group Management using Multiple Overlays draft-kassinen-p2prg-group-management," Internet-Draft(2009).
5) Lakshminarayanan.S,Ion.S,Hari.B, and Randy .H.Katz, "OverQoS: An Overlay based Architecture for Enhancing Internet QoS," SIGCOMM Computer Communication Review 2003, vol.33, pp.11-16(2003).
6) Akira.I, "A Hierarchical Multilayer QoS Routing System with Dynamic SLA Management," IEEE Journal on Selected Areas in Communications 2000, vol.18, issue.12, pp.2603-2616(2000).
7) Stoica.I,Morris.R,Karger.D,Kaashoek.M.F, and Balakrishnan.H, "Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications," inProc. SIGCOMM 2001, pp.149-160(2001).
8) Maymounkov.P, and Mazieres.D, "Kademlia:A Peer-to-peer Information System Based on the XOR Metric," inProc. International workshop on Peer-To-Peer Systems 2002, pp.53-65(2002).
9) Rowstron.A, and Druschel.P, "Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems," inProc. IFIP/ACM International Conference on Distributed Systems Platforms 2001, pp.329-350(2001).
10) Dabek. F, Zhao. B, Druschel. P, Kubiatowicz. J, and Stoica. I, "Towards a common API for Structured Peer-to-Peer Overlays," International workshop on Peer-To-Peer Systems 2002, p.7-13(2003).
11) Sean. R, Dennis. G,Timothy. R, and John. K, "Handling Churn in a DHT," inProc.USENIX Annual Technical Conference 2004, pp.10-22(2004).
12) Haiyong.X,Richard.Y,Arvind.K,Yanbin.L, and Avi.S, "P4P: Provider Portal for Applicatioion P2P," inProc. SIGCOMM 2008, pp.351-362(2008).