Doctoral Thesis

Studies on User-Behavior Driven Content Distribution Environments

2013

Yoshio Sakurauchi

DOCTORAL PROGRAM IN INTEGRATED SCIENCE AND ENGINEERING
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING
RITSUMEIKAN UNIVERSITY

Studies on User-Behavior Driven Content Distribution Environments

abstract

In recent years, handheld terminals like smartphones have spread into our lives. Accordingly, user-generated content has become dominant in the Web. This kind of content is mainly distributed by SNS which often distributes content to outside of targeted areas, because it is hard to control the geographical extent of content distribution in SNS. Meanwhile, the author of content is one of the most important factors of UGC. Most existing services require mutual user evaluation with a voting mechanism or other reputation mechanisms which tend to increase the load of central servers. In addition, large size content like videos increases a traffic load on networks.

In this thesis, to resolve the problems of such content distribution environments, three approaches are proposed: (1) a method to create user clusters which enables the control of the geographical extent of content distribution, (2) a method to distribute content and aggregate reputation without central servers, and (3) a method to redirect content requests at layer-2. In the first method, the clustering method organizes places where content is exchanged into hierarchies so that the extent of content distribution can be controlled with its parameters to swap user records among nodes organizing clusters. The second method makes content distribution and reputation aggregation work efficiently in a P2P manner. The simulation results provide insights on trade-offs between consumed network resource and required time for content distribution and reputation aggregation. As the third method, a layer-2 redirect method is implemented on top of the OpenFlow programmable switch platform. It shows a better performance compared with a conventional redirect system working at layer-3. Simulation results conducted in random networks also show that the proposed system works collaboratively on user demands, which reduces the load of central servers and improves the user experience.

The proposed three methods resolve respective problems at a certain level. Using the three methods, a new user-behavior driven infrastructure can be constructed for content distribution environments. It enables the control of the geographical extent of content distribution, simultaneous content distribution and reputation aggregation, and seamless content delivery reducing the utilization of central servers or clouds. In other words, the distribution of content can be controlled geographically with the assurance of the quality of information under an efficient network environment.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Information Exchange in Daily Life

In recent years, handheld terminals like smartphones have spread into our lives. Accordingly, user-generated content (UGC) has become dominant in the Web [1]. Through user centric media such as Twitter, Facebook, and YouTube, users are producing and consuming UGC actively. UGC is not only useful in daily life but also in serious incidents and natural disasters. In some cases, users can get information about them earlier than traditional media like TV and radio [2]. This kind of contents is mainly distributed by SNS. Users living in a specific area value local news of their area as well as global news, while they pay little attention to local news of other areas. In other words, information has its appropriate distribution extent. Unfortunately, SNS often distributes content to outside of targeted areas, because it is hard to control the geographical extent of content distribution in SNS.

1.2 Content Distribution and Reputation Aggregation

Content a specific user creates is evaluated by other users in the process of distribution. The author of content is one of the most important factors of UGC. Even if the same content is distributed, its credibility varies with its author. Most existing services require mutual user evaluation with a voting mechanism or other reputation mechanisms.

However, those mechanisms tend to increase the load of central servers.

Generally UGC is treated by the conventional client-server architecture. This means that, with the help of cloud computing, there is no limit of scalability in a sense if anyone can provide enough funds. Although cloud computing enables both content distribution and reputation aggregation with its availability and scalability, it is better to minimize the use of clouds because of cost problems and some other reasons, such as benefiting from P2P-based systems.

Meanwhile, P2P-based content sharing systems have been used to realize scalable, fault-tolerant and sometimes anonymous content sharing [3]. In addition, P2P-based systems can exploit local networks of user terminals such as LAN and MANET. Some academic and commercial projects have tried to mix the two paradigms to take advantages of both [4]. For instance, the cloud can be used just for bootstrapping P2P networks because of its availability.

1.3 Efforts for Content Delivery

The amount of Internet traffic has been growing beyond the enhancement of its capacity. Moreover the amount of published information is also growing at an exponential rate. Large size content like videos increases a traffic load on networks. Consequently, there are the demands on performance, robustness, and low latency for a worldwide Internet population. These problems are often addressed as exaflood [5] and information explosion [6]. In Japan, the Ministry of Education, Culture, Sports, Science and Technology has supported researches related to these phenomena [7].

To meet the demand on performance, robustness, and low latency, traditional solutions have led to web proxy cache systems (hereinafter just referred to as cache systems). Cache systems process requests from clients on behalf of web servers. Currently, cache systems are generally accessed through layer 4-7 scripts and commands, such as the *route* command on Posix systems, and usually, manual configuration or JavaScript code for proxy settings. As another approach, DNS records which are used to resolve web servers' IP address are often rewritten to make clients use cache systems such as CDNs (Content Delivery Network). Thus, administrators and/or clients are forced to struggle with some tedious and error-prone operations to use these cache systems. This is far from robustness.

1.4 The Composition

In this thesis, three approaches are proposed to resolve the problems of content distribution environments. In the first method, places where content is exchanged are organized into hierarchies so that the extent of content distribution can be controlled. The second method makes content distribution and reputation aggregation work efficiently in a P2P manner to reduce the load of central servers. As the third method, a layer-2 redirect method is implemented on top of the OpenFlow programmable switch platform to deliver content to users requested seamlessly. With these methods, the distribution of content can be controlled geographically with the assurance of the quality of information under an efficient network environment.

The rest of this thesis is organized as follows. Chapter 2 introduces a model of user centric media and basics in the field. Chapter 3 describes a distance-controllable user clustering method. Chapter 4 explains a method to distribute content and aggregate reputation in a P2P manner. Chapter 5 proposes a method to deliver content to users seamlessly on user demands. Finally, Chapter 6 concludes the thesis and gives an outlook on the future.

Chapter 2

Background

2.1 User Centric Media

Ultimately, a human being is a brain which has the five senses as input and muscle as output. From this viewpoint, it can be said that how information is exchanged among people adequately affects the quality of human life because the human society is constructed by information exchange among people. Before the Web has become popular, most content is created and distributed by mass media. Common people can just distribute their content by word of mouth even if they have created valuable content.

Information technology has changed the situation. The Web has provided many opportunities for people to distribute their original content. Smart devices have enabled people to create content such as text, images, and videos wherever they are. The real-time Web which enables people to receive content as soon as it is published has become popular. Nowadays, content distribution is not only possible by mass media but also by common people.

The author analyzed and modeled user centric media as illustrated in **Figure 2.1**. The model consists of three layers. The small circles indicate user terminals such as

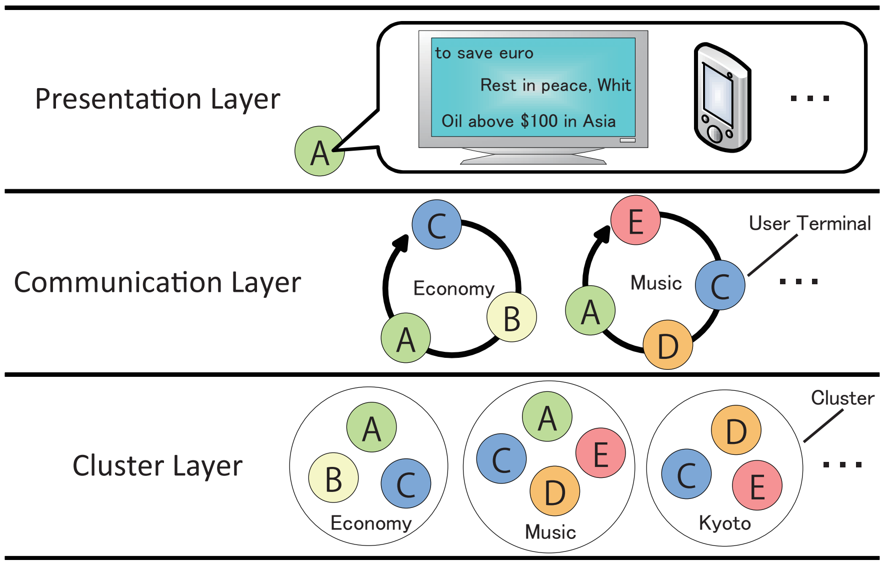


Figure 2.1: A Three-layer Model of User Centric Media

computers and smartphones. The role of each layer is described below.

2.1.1 Cluster Layer

The cluster layer manages user groups. Users are clustered in accordance with their interests. Users may belong to multiple groups at the same time. The large circles surrounding user terminals indicate clusters of user terminals. User terminals marked with the same label represent the same user terminal.

There are many well-known soft clustering methods [8]. Instead of automatic clustering, users may create, join and leave a cluster manually. This layer is essential to create user clusters which enables the control of the extent of content distribution. A method for the clustering is explained in Chapter 3.

Whenever users join/leave a cluster, its event is reported to the communication layer so that their terminals can be included in or excluded from the target of content distribution and reputation aggregation. It should be emphasized that user management mechanisms in the cluster layer are also responsible for detecting an unexpected withdrawal of user terminals by periodic check or other methods.

2.1.2 Communication Layer

The communication layer controls how to distribute content and aggregate reputations in each of clusters. As it is discussed in Chapter 4, this layer plays a vital role for the circular board method based on Chord, which achieves content distribution and reputation aggregation simultaneously.

2.1.3 Presentation Layer

The presentation layer provides methods to present content to users. Presentation methods would vary according to the difference of types of user terminals such as smartphones, tablets and desktop computers. Although it is not strictly defined how distributed content must be presented, presentation methods are expected to express which clusters content came from. Users might belong to several clusters. Content is delivered from each of the clusters. Even if the same content is posted, it will be public or foreclosed by the difference in clusters. This means that credibility of content depends on not only who posted it but also which clusters it came from. For an extreme example, if a piece of information which says "Japan did default" is distributed in a joking cluster, most users just skip it because they will never believe it. In contrast, if it is distributed in an economic cluster, users will be surprised and try to check if it is true or not in some way. How to express multiple properties of a lot of content could be an additional research issue.

2.2 Content Distribution on the Web

There is no doubt that the Web is the most common platform for information sharing. Web contents can be divided into two types. One is the surface Web and the other is the deep Web [9]. Contents in the surface Web are indexable by crawlers of conventional search engines which follow links from a list of known web contents to unknown ones. In contrast, contents in the deep Web are not indexable because of isolated links, dynamic pages like Flash, and so on.

To obtain content from the Web, URIs of contents must be known in some way. Keyword search is used to get URIs in general. Of course, with keyword search, contents in the deep Web cannot be obtained easily. Although several techniques have been proposed to solve this problem, they have not attained full solution [10].

Even for content in the surface Web, it is not easy to be obtained sometimes because results of keyword search are different depending on keywords specified by users. This problem is well known as the digital divide [11]. In short, despite the fact that useful information exists, some users cannot obtain it.

Meanwhile, the real-time Web such as Twitter which enables users to receive information as soon as it is published has become popular. Because of its promptness, it is getting indispensable for the society in these days. In particular, when serious incidents and natural disasters have happened, it is not unusual that information about them is published earlier than traditional media like TV and radio. Typically, services of the real-time Web have put restrictions on the size of content which users generate. If users need to create a large content, the content is mainly generated and located on the Web using other services. Chapter 5 proposes a layer-2 redirection engine which helps such services to distribute large contents on user demands.

Chapter 3

Distance-controllable User Clustering

3.1 Purpose

One of the simplest models of content distribution in daily life is a person-to-person propagation model. In a person-to-person propagation model, users exchange information when they meet each other.

Figure 3.1 shows relations among users, places, and time in the person-to-person information exchange. Although user A and user B have visited the same place, they never exchange information with each other because the time is different. Similarly, user C never exchanges information with user A and user B, even if they share the same time, because places are different. From a viewpoint of content distribution, these geographical distance and temporal distance should be eliminated. For example, if a geographical distance of 30 km is eliminated, users can exchange information beyond a distance of 30 km. If a temporal distance of 1 hour is eliminated, users can exchange information beyond a distance of 1 hour.

More generally, this kind of problem can be treated as user clustering. This chapter presents a method to create user clusters which enables the control of the extent of content distribution in the cluster layer.

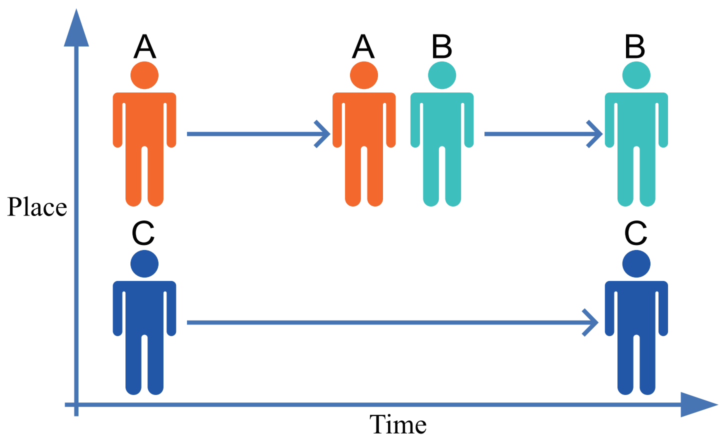


Figure 3.1: Distance between Users

3.2 Related Work

Several information exchange (or sharing) systems which use user locations have been proposed and implemented. These systems can be compared with each other from 3 perspectives described below.

1. Locality

   Locality is a linkage among locations which a system targeted at. If locality is high, information exchange among users is only happen in the same place. If locality is low, information exchange among users happens between remote locations.

2. Connectivity

   Connectivity is divided into 2 types; permanent connection and non-permanent connection. If connectivity is non-permanent, users can exchange information only in the same place and the same time. If connectivity is permanent, users can exchange information even they visit the same place at different time.

3. Activity

   Activity is whether users are required or not to take some actions to exchange information. For example, sending an e-mail message is active but receiving it on a cellular phone is inactive (passive).

iCAMS [12] is a communication tool utilizing user location and schedule information. In iCAMS, a suitable communication means to each user is suggested from an office phone, a mobile phone, an e-mail, and a special short messaging service of iCAMS according to its user context. iCAMS has low locality and users can exchange information in a remote place. Regarding connectivity, it has permanent connection and users can exchange information whenever they want. Since iCAMS has been implemented for existing communities, users cannot exchange information across communities. Activity of iCAMS depends on a communication means which it has suggested.

Alipes [13] is an architecture for mobile applications utilizing user location information. The architecture consists of 4 sections: the positioning platform which provides user location for applications with abstraction, the privacy and security handler which grants or denies any location information requests from application, the map service which provides methods for obtaining map information, and the service infobase which provides information for applications from the Web and an XML database. As an application example, *FriendFinder* which indicates positional relationships between users on a map and *GeoNotes* which enables users to leave/read a note on a particular place are presented in the paper. If an application is implemented just on the architecture, the locality will be high and the connectivity will be non-permanent connection. The activity depends on each application.

Kontti [14] utilizes user context and enable users to exchange a message each other. For example, a user can leave a message at a particular place for another user who is in a particular context. Although Kontti has permanent connection, the locality is high and users are required to take some active operations to put/get a message.

As of 2008 in Japan, there are several information services which focus places which users visit at such as *Kiseki* [15] and *PiTaPa goopas* [16]. *Kiseki* enables users to write a diary regarding places where they have visited that day using GPS location information on a mobile phone. Recommended content is also provided for users based on geographical action history. *PiTaPa goopas* delivers information to users via e-mail when users touch an IC card called *PiTaPa* at a ticket gate of railway stations. Both of them has high locality. *Kiseki* has permanent connection, while *PiTaPa goopas* does not have permanent connection. Regarding the activity, *Kiseki* is active and *PiTaPa goopas* could be passive. Meanwhile, all of the content provided by both services are commoditized information by content providers. They do not support content exchange between users.

3.3 Distance-controllable User Clustering

3.3.1 User Similarity

The proposed clustering method focuses on places as user location. Each place has attributes such as *theater* and/or *restaurant*. It can be said that places with the same attributes are similar and users visiting such places are also similar even if users are visiting different places as long as those places have the same attributes. Considering the geographical distance, users who visited the same place are more similar than other users who visited faraway places which have the same attributes. The same holds true for the temporal distance; users who visited places at the same time are more similar than other users who visited the places at different time.

3.3.2 The Overlay Location Network

The overlay location network is the core of the proposed user clustering method. It consists of the following factors.

First, places which have the same attributes are defined as

$$P = \{p_1, p_2, p_3, \cdots, p_m\} \tag{3.1}$$

and users who have been to $\forall p \in P$ are defined as

$$U = \{u_1, u_2, u_3, \cdots, u_n\} \tag{3.2}$$

Each user $u \in U$ has a set of tuples; each tuple consists of a place p user visited and its time t.

$$record(u) = \{(p, t)\} \tag{3.3}$$

The current time is defined as t_c. Note that time t is a temporal interval $[t, t + \delta)$ defined by a short period of time δ in a precise sense.

A set of users $u \in U$ who visited p at t is defined as local cluster $L_{p,t}$.

$$L_{p,t} = \{u \mid u \in U, (p, t) \in record(u)\} \tag{3.4}$$

Since users share the same place and time, users who belong to the same local cluster resemble each other. The node N_p is defined as a set of local clusters $L_{p,t}$ in the same place p during $[t_c - t_u, t_c]$ where t_c is the current time and t_u is a unit time.

$$N_p = \bigcup_{t=t_c-t_u}^{t_c} L_{p,t} = \{u \mid u \in U, (p, t) \in record(u), t \in [t_c - t_u, t_c]\} \tag{3.5}$$

Each local cluster holds records of users who have visited its place at its time. Consequently, a node holds records of users who have visited its place during a time unit (hereinafter called user records). Content is exchanged among users in the same node under the similarity of the fact that those users have visited the same place during the same unit time. The temporal distance between users in the same node can be controlled by changing the length of unit time. In other words, the temporal distance between users can be eliminated up to a length of unit time.

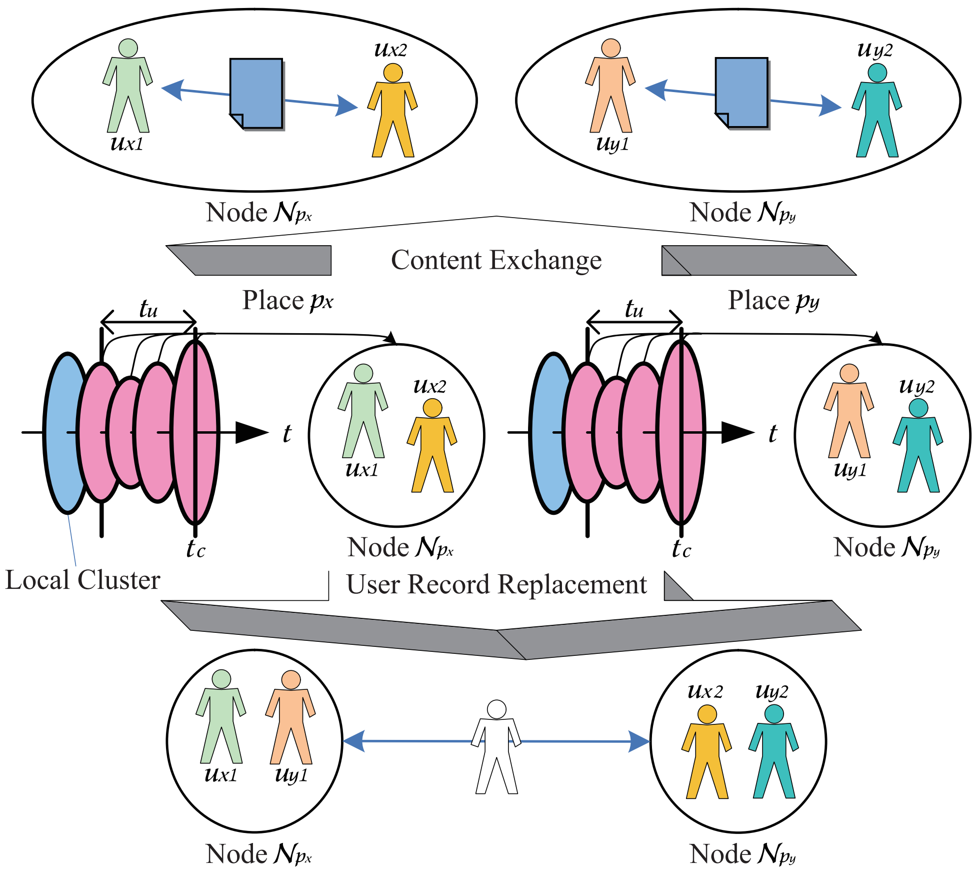


Figure 3.2: Content Exchange between Users and User Records Replacement between Nodes

A set of all nodes N_p is defined as the cluster C.

$$C = \bigcup_{p=1}^{m} N_p = \{u \mid u \in U, (p, t) \in record(u), p \in P, t \in [t_c - t_u, t_c]\} \tag{3.6}$$

Users in the same cluster has a resemblance to each other, that they have visited places which have the same attributes during the same time unit. Under the similarity, to eliminate geographical distance between users, user records in a node are exchanged with other node in the same cluster. Due to this, the geographical distance can be eliminated up to a distance between actual places which correspond to a pair of nodes that user records are exchanged with each other.

Figure 3.2 represents a conceptual diagram of content exchange between users and user record replacement between nodes. First, at place p_x and place p_y, node N_{p_x} and node N_{p_y} are constructed by respective local clusters from time $t_c - t_u$ to time t_c. In node N_{p_x}, user u_{x1} and user u_{x2} are enrolled. Similarly, in node N_{p_y}, user u_{y1} and user u_{y2} are enrolled. Content exchange between users happens automatically in the same node. For example, user u_{x1} and user u_{x2} may exchange content each other in node N_{p_x} or user u_{y1} and user u_{y2} may exchange content each other in node N_{p_y}. User record replacement also happens automatically between nodes. For example, user u_{x2} in node N_{p_x} and u_{y1} in node N_{p_x} may be swapped so that users who visited different places can exchange content.

3.3.3 Organization of the Overlay Location Network

When the number of nodes in a cluster is large, it is not efficient for each node to exchange user records with all other nodes. For this reason, nodes in a cluster are organized in a hierarchy as shown in **Figure 3.3**. Each node exchanges user records with other nodes using the links. When making a link between nodes, geographically close nodes are chosen so that the similarity between users is not to be lost; a user would be more similar to geographically close users than geographically distant users. At the same time, content exchange between users that are geographically dispersed may be valuable in some cases. In order to achieve this kind of content exchange, nodes in a cluster are divided into several layers. Each layer contains at least one hierarchical cluster which promotes content exchange among distant users at its correspondent layer level.

The steps for a method to organize an overlay location network are indicated below.

1. According to intended places, the method defines a unit distance for geographical distance. It also determines the number of overlay layers.

2. According to intended similarities of places and/or users, the method defines an

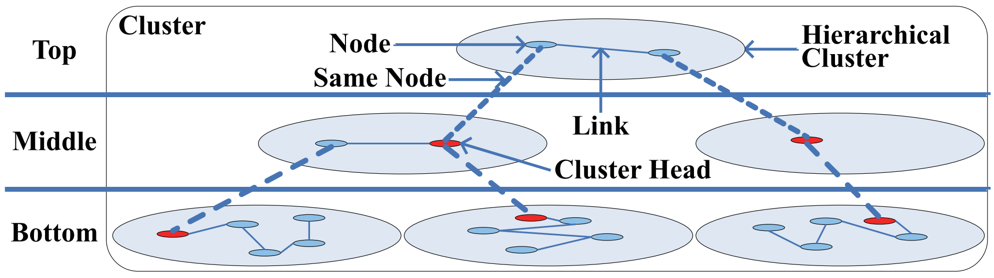


Figure 3.3: A Three-layer Overlay Location Network

estimator which determines whether a link should be established between nodes.

3. From the bottom layer to the top layer, (a) and (b) are repeated until the all layers have been organized. All nodes must be jointed in the bottom layer.
    (a) Links among nodes are created using the estimator.
    (b) A cluster head is selected and made to join the next higher layer.

Actual overlay location networks are organized in the simulations later, which will make each step easy to be understood.

3.3.4 Controlling the Extent of Content Distribution

In the overlay location network, content is exchanged among users who belong to the same node. Thus, user record replacement among nodes represents that those users have visited corresponding places virtually. This means that geographical distance between the corresponding places is eliminated at the same time. Temporal distance is also eliminated by content exchange of users in a node up to a unit time of its node definition.

How much user records should be replaced is controlled by 3 parameters shown below.

The number of replacement is the number of user record replacements executed in a time unit. This parameter controls how far a user can move virtually from its original node. For example, if the number of replacement is 50, a user can virtually get to a node which is up to 50 hops far away from an original node of the user.

Replacement ratio is a ratio of user records which are going to be replaced. This parameter value should be in 0-100%. If the number of user recodes in each node is not equal to another, the smaller number is regard as 100% so that the number of user recodes in each node is not changed. Replacement ratio controls how many users should go to other places virtually.

Replacement order is an order of user record replacement in each node and each layer. This parameter controls virtual pathways of users in the user record replacement phase. For example, if a cluster is organized by the linked-list structure, an order could be top down, bottom up, or random.

In the meantime, although user records are possible to be shared by nodes, the proposed clustering method treats user records as things which are expected to be replaced to represent user behavior indicating one user visits one place at the same time. The reason is to make the extent of content distribution easy to be controlled. Sharing user records represents that a user exists in multiple places at the same time virtually and increases complexity of the control. For instance, if node N_a and node N_b share a user record of user x, eliminated geographical distance of content exchange which happened via user x between any user u_a in node N_a and any user u_b in node N_b is difficult to be measured. As the number of intermediate users for content exchange grows, it is almost impossible to control and estimate the extent of content distribution. On the contrary, if user records are replaced, even if there is an intermediate user x for content exchange, the extent of content distribution can be estimated by controlling a replacement behavior of user x.

3.4 Simulation 1: Post Office

The first simulation is targeted at post offices across all over Japan as places where content exchange is performed. Eliminated geographical distance and temporal distance are investigated by shifting the values of the 3 parameters: the number of replacement, replacement ratio, and replacement order.

3.4.1 Simulation Setting

A set of places P (Formula 3.1) is a collection of 19,603 post offices across Japan. The number of users in each post office is determined by one-hundredth of a normal distribution that a mean is each prefecture's population divided by the number of post offices in its prefecture and a standard division is 20% of the mean. The final number of users in each post office is divided by 100, because it is unlikely to think that all of them visit a post office within a time unit. Accordingly, a set of users (Formula 3.2) contains 1,268,551 users in total. A unit time is fixed to one hour and each recorded time t of user record $record(u)$ (Formula 3.3) is determined by a uniform distribution which represents that users visit its place during 0 to 59 minutes ($\delta = 1$ minute) uniformly. Using the above, each local cluster $L_{p,t}$ (Formula 3.4), each node N_p (Formula 3.5), and the gross cluster C (Formula 3.6) are determined consequently.

A unit distance is *kilometer*, which is calculated from latitude and longitude in the equidistant cylindrical projection. Although there is a margin of error between the real distance kilometer and the calculated kilometer in the equidistant cylindrical projection, it would be admissible because its range is small. Latitude and longitude of each post office is obtained from *CSV Geocoding Service* [17] managed by center for spatial information science at the University of Tokyo. The number of overlay layers in the overlay location network is fixed to 3 to represent municipalities as the bottom layer, prefectures as the middle layer, and whole of Japan as the top layer. The linked-list is selected as an organizing topology of nodes in each layer. An estimator which determines whether a link should be established between nodes is the number of users in each nodes; that is, nodes are organized into a line from the head node, which has the largest number of users, to the tail node, which has the smallest number of users. The head node in each local cluster becomes a cluster head. In the content exchange phase, 2 users are randomly selected within each node and exchange content each other.

Possible replacement order in the linked-list is top down or bottom up. In the top down replacement, user records are replaced from the top layer to the bottom layer and from the head node to the tail node within each hierarchical cluster. In the bottom up replacement, user records are replaced from the bottom layer to the top layer and from the tail node to the head node within each hierarchical cluster. To investigate the impact of replacement order on controlling geographical distance and temporal distance, 4 basic combinations are considered as follows.

- Top down - Top down (hereinafter called TT)
- Top down - Bottom up (hereinafter called TB)
- Bottom up - Top down (hereinafter called BT)
- Bottom up - Bottom up (hereinafter called BB)

For example, if replacement order is TB, user records are replaced in bottom up after replaced in top down. Another possible replacement order is random.

- Random - Random (hereinafter called RD)

The number of replacement varies by 1 from 0 to 100 step by step. Note that each step (one replacement) indicates one replacement of user records in the 5 basic combinations. Replacement ratio varies from 0% to 100% by 1%. All content exchange will

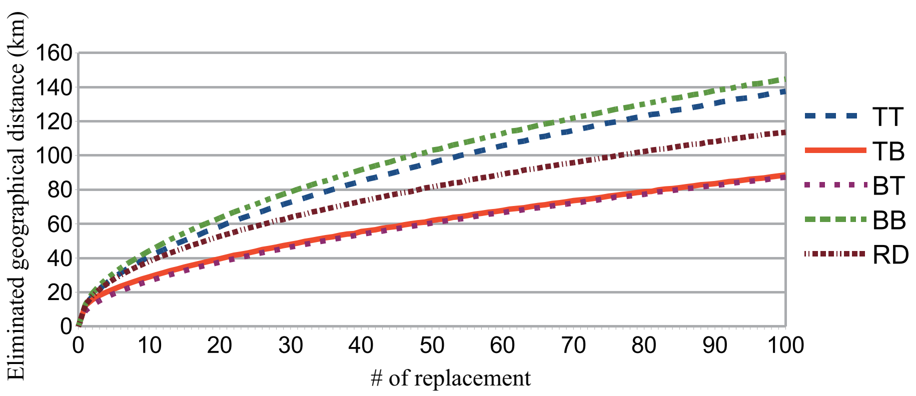


Figure 3.4: Relationship between the Number of Replacement and Eliminated Geographical Distance

be performed randomly; 2 users are randomly selected within each node and exchange content with each other.

3.4.2 Result

The mean of temporal distance and geographical distance between users who have exchanged content is investigated for each combination of the replacement orders.

First, any parameter does not have a substantial impact on eliminated temporal distance. This means that eliminated temporal distance is independent from user recored replacement. In any results, eliminated temporal distance distributes around 20 minutes which is nearly equal to an expectation value of margins between recorded times in user records. Remember that all users visit places during 0 to 59 minutes ($\delta = 1$ minute) uniformly. To control temporal distance in content exchange phase, another method will be needed such as user selection methods considering recorded times in each user record which determine a pair of users who exchange content each other.

Next, relationship between the number of replacement and eliminated geographical distance is shown in **Figure 3.4**. Eliminated geographical distance is monotonically increasing for any replacement order. Hence, geographical distance can be controlled by the number of replacement. Meanwhile, averages of eliminated geographical distance

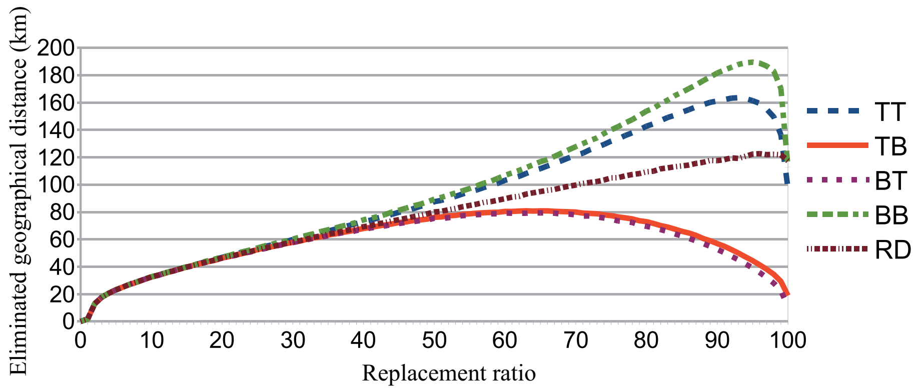


Figure 3.5: Relationship between Replacement Ratio and Eliminated Geographical Distance

converges to about 461 km with the sample data of this simulation. This indicates that eliminated geographical distance is limited when all content exchanges are performed randomly. If this limitation is required to be overcome, another method will be necessary in the same reason for the temporal distance.

Finally, relationship between replacement ratio and eliminated geographical distance is shown in **Figure 3.5**. There is an apparent local maximum point for each replacement order. In other words, eliminating geographical distance does not simply increase as replacement ratio increases. If too many user records are replaced, the probability of content exchange among users who originally enrolled in the same node becomes higher. One of the reasons why a local maximum of TB and BT is around 63% would be because replaced user records tend to go back to its original node frequently, even if user records are replaced between nodes layering up-and-down.

These simulation results have revealed that eliminating temporal distance only depends on a unit time when users visit places temporally-uniformly and content exchange is performed randomly. Besides, eliminating geographical distance converges to a particular value and the increase of eliminated geographical distance does not monotonically increase; there is an apparent local maximum point.

3.5 Simulation 2: Railway Station

The second simulation is targeted at railway stations as places where content exchange is performed. In this simulation, a target of the extent of content distribution is fixed to 5 stations; users exchange content with other users who visit a station which is 5 stations away from a station of their daily use on average. Replacement order is also fixed to make the simulation simpler. The impact of remaining 2 parameters, the number of replacement and replacement ratio, to eliminate geographical distance is investigated.

3.5.1 Simulation Setting

A set of places P (Formula 3.1) includes all 42 stations of the *Keihan Main Line* which is a straight-route railway system (between Yodoyabashi and Demachiyanagi) operated by *Keihan Electric Railway Co., Ltd.* A unit time is fixed to one day and the number of users in each station is 10% of outgoing passengers at each station [18]. Accordingly, a set of users (Formula 3.2) contains 9,211 users in total. Note that current time t_c and record time t in user record $record(u)$ (Formula 3.3) are out of consideration, because eliminating temporal distance is not considered in this simulation. Using the above, each local cluster $L_{p,t}$ (Formula 3.4), each node N_p (Formula 3.5), and the gross cluster C (Formula 3.6) are determined consequently.

A unit distance is a *station* which represents how far two stations are apart from each other. For example, if there is no station between two stations, its geographical distance is 1 station. If there is one station between two stations, its geographical distance is 2 stations.

The number of overlay layers in the overlay location network is fixed to 3 to represent stations which local trains stop at (hereinafter called local stations) as the bottom layer, stations which express trains stop at (hereinafter called express stations) as the middle layer, and stations which limited express trains stop at (hereinafter called limited express stations) as the top layer. Note that local trains stop at all stations but express trains and limited express trains just stop at several stations. Express trains stop at all limited express stations. An organizing topology of nodes in each layer is just as the railway system; neighboring stations are simply linked on each layer. In the content exchange phase, every pair of users is selected within each node and exchanges content with each other.

Replacement order is top down; user record replacement is performed from the top layer to the bottom layer and to and from Yodoyabashi via Demachiyanagi within each hierarchical cluster. The number of replacement varies by 1 from 0 to 100 step by step. Replacement ratio varies from 0% to 100% by 1%.

3.5.2 Result

Geographical distance of all content exchanges is investigated for each combination of the number of replacement and replacement ratio. From the all cases that geographical distance becomes 5 stations on average, 3 cases are chosen to look into details:

- 1% - 66 times

  A case where replacement ratio is low and the number of replacement is large

- 8% - 8 times

  A case where replacement ratio is middle and the number of replacement is middle

- 53% - 1 time

  A case where replacement ratio is high and the number of replacement is small

The mean and the maximum of eliminated geographical distance for each case is shown in **Figure 3.6** and **Figure 3.7** respectively.

According to Figure 3.6, the 3 cases are similar to each other and eliminated geographical distance is less than 5 stations on average in more than half stations. In

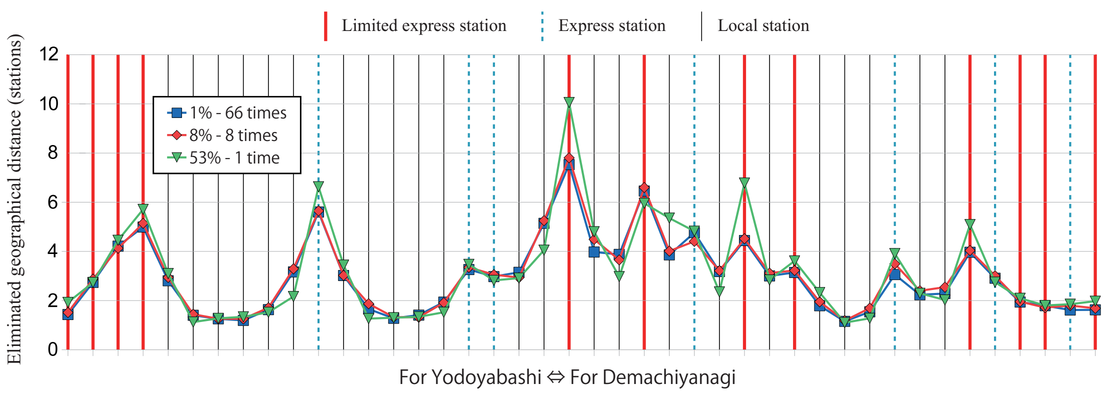


Figure 3.6: Eliminated Geographical Distance - Mean

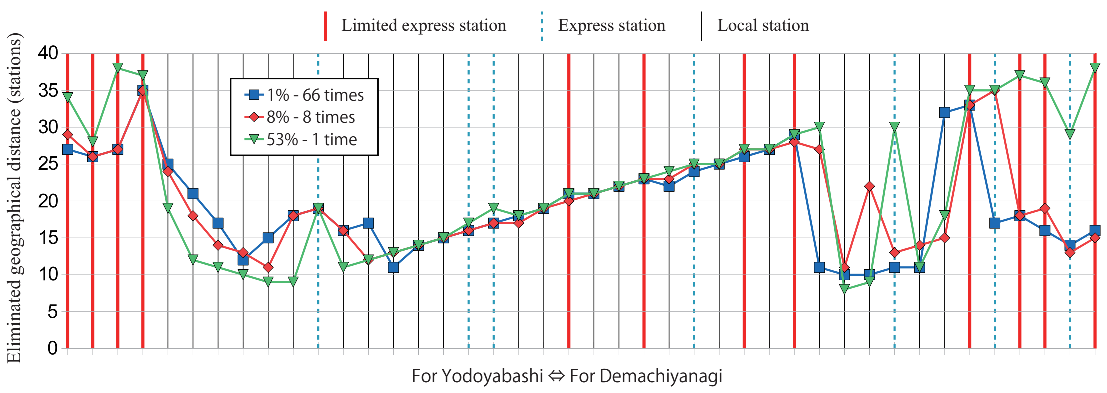


Figure 3.7: Eliminated Geographical Distance - Maximum

addition, eliminated geographical distance increases in the middle part of the railway system and decreases in the each end part of the railway system.

According to Figure 3.6 and Figure 3.7, mean and maximum of eliminated geographical distance in the case of 1%-66 times tends to be higher than that in the case of 53%-1 time. In other words, replacement ratio has a greater impact on the extent of content distribution for limited express stations and express stations. By contrast, the number of replacement has a greater impact on the extent of content distribution for local stations as mean and maximum of eliminated geographical distance in the case of 53%-1 time tends to be higher than that in the case of 1%-66 times.

These simulation results have revealed that eliminating geographical distance increases in the middle part of the railway system and decreases in the each end part of the railway system when the railway system is a straight route. Additionally, the extent of content distribution is mainly affected by the number of replacement in the bottom layer, while the replacement ratio in other layers.

3.6 Discussion

3.6.1 About Simulation 1

The simulation 1 has been targeted at post offices across all over Japan as places where content exchange is performed and a unit time is fixed to one hour. The simulation results can be applied to real scenarios that content is exchanged by users who use post offices in a particular time. For example, elderly people may use post office in the morning, salaried employee may use in lunch time, and students may use in the evening.

Meanwhile, the results of TB and BT indicate that replaced user records tend to go back to its original node frequently. This reversion means that only temporal distance has been eliminated.

Finally, it is clear that the extent of content distribution can be controlled by the 3 parameters: the number of replacement, replacement ratio, and replacement order. For instance, if eliminating geographical distance needs to be increased, replacement order should be BB and replacement ratio should be a local maximum of BB in Figure 3.5. Under these setting, the extent of content distribution can be simply controlled by the number of replacement on demand.

3.6.2 About Simulation 2

The simulation 2 has been targeted at railway stations as places where content are exchanged with the unit time fixed to one day. The simulation results can be applied to a real scenario that content are exchanged by common daily users. The users may exchange content regarding an area. Note that the extent of content distribution is fixed to 5 stations on average.

Additionally, depending on a layer nodes belong to, the extent of content distribution is strongly affected by the number of replacement or the replacement ratio. This means that eliminating geographical distance for a user depends on where he/she has visited. If this difference should be abolished, the number of replacement and the replacement ratio should be customized for each layer; in the top and middle layer, the number of replacement should be small and the replacement ratio should be high; in the bottom layer, the number of replacement should be large and the replacement ratio should be low. Although, just 3 cases–1%-66 times, 8%-8 times, and 53%-1 time–are shown in Section 3.5.2, there are a lot of other possible combinations. From the combinations, it would be easy to select parameters which make an overall average of the extent of content distribution to be 5 stations like r_t%-n_t times for the top layer, r_m%-n_m times for the middle layer, and r_b%-n_b times for the bottom layer.

On the other hand, content gathers where people gather in the real world. This fact implies that it is natural to think that eliminating geographical distance for a user depends on where he/she has visited in the real world. Therefore, it would be reasonable to accept the difference.

3.6.3 About the Distance-controllable User Clustering

If the unit time is fixed to short one, content exchange between users who visit a place in particular time would be promoted. If the unit time is fixed to long one, content exchange between users who visit a place in general time would be promoted.

Additionally, depending on a layer which nodes belong to and its estimator which determines whether a link should be established between nodes, the extent of content distribution is strongly affected by the number of replacement or replacement ratio.

The setting of parameters should be modified according to whether optimizing a layer is enough or optimizing all layer is required.

Finally, one place has been corresponded to one node in the overlay location network (Formula 3.5). Owing to this, it will be easy to implement the overlay location network as a distributed system. A node corresponds to a server.

3.7 Summary

In this chapter, a method to create user clusters which enables the control of the extent of content distribution has been proposed. The method uses the overlay location network which focuses on geographical distance and temporal distance between users. Simulation results have revealed that the geographical extent of content distribution can be controlled by the parameters of the proposed method.

Chapter 4

Content Distribution and Reputation Aggregation

4.1 Purpose

This chapter proposes to apply the circular board method based on Chord [19] to user centric media. The application enables to distribute content and aggregate reputation at the same time efficiently in a P2P manner. The circular board is a part of Japanese culture and used in communities to share information. If someone wants to distribute content to their community, he/she puts the content on a (physical) circular board and the community members pass around the circular board in order (physically). The community members may take some actions such as signing, making comments, and so on, when they get the circular board and before they pass it to the next member. This circular board method would also work well for (digital) user centric media especially in a P2P manner. Chord is employed to determine the order of passing the circular board because Chord is the most common algorithm to organize a P2P cyclic topology. Its cyclic topology makes it possible to effectively collect the reputation from users at the same time when each piece of UGC passes through user terminals. The results of simulations provide insights about trade-offs between consumed network resource and required time for content distribution and reputation aggregation.

4.2 Related Work

Several system models have been proposed to distribute content to interested users. While the circular board model can achieve content distribution and reputation aggregation simultaneously, these models only set a goal to content distribution. The same thing can be said about reputation aggregation algorithms.

4.2.1 Content Distribution

The publish/subscribe model, the gossip model, and the P2P streaming model are three widely used system models for content distribution.

Publish/Subscribe Model

The publish/subscribe model divides users into publishers and subscribers. Publishers publish information without specific receivers. Subscribers can subscribe to any publishers and get information published by them.

Twitter is a typical example of the publish/subscribe model and one of the most popular microblogging services. Users can send posts of up to 140 characters, called "tweets," and subscribe to other users' tweets. This action to subscribe is called "follow" and subscribing users are called "followers." Generally Twitter is also regarded as an online social networking service because of these following-followed relationships. Users of Twitter are not limited to only individuals but also organizations such as companies and universities [20]. Automatic programs are also handled as users. That kind of programs post popular tweets, tweets in specific fields, and so on. In addition, Twitter has a function, called "retweet," which enables users to diffuse information they credited to their followers. These facts mean that users can get information from other users whom they are not directly following.

Some publish/subscribe services are built on top of a distributed hash table (DHT) unlike Twitter which is built on the cloud. There are several ways to maintain or generate dissemination tree on top of DHT [21]. Ferry [22] proposes an architecture that extensively yet wisely exploits the underlying DHT overlay structure to build an efficient and scalable platform for content-based publish/subscribe services. Moreover, some approaches which enable subscribers to get information from unforeseen publishers have been tried to provide users more opportunity to get unknown information. For instance, keyword-based content dissemination has been tried instead of publisher-based content dissemination [23].

Gossip Model

The gossip model mimics word of mouth in the real world. Information is disseminated by users who think it is worth spreading. In case of using wireless ad-hoc networks between mobile terminals, users who did not encounter anyone cannot obtain content. Using online social networks [24], on the other hand, users can get content wherever they are in the real world.

The gossip model can be also regarded as an epidemic model especially when gossip systems work without users' actions such as sharing with someone. In this case, the peer sampling is a very important matter. A peer sampling mechanism determines which peers should share content with which peers. Some gossip peer sampling mechanisms postulate the cloud and physical network [25]. [4] has proposed a hybrid architecture of the cloud and P2P based on Cyclon [26]. It has also shown that the economic cost can be reduced effectively in commercial services, the hourly News Update podcast from CNN and the Dilbert's comic strips.

P2P Streaming Model

The P2P streaming model uses the tree and/or mesh topology to distribute content among a large number of users [27]. Content is delivered from a server to clients in a continuous fashion. This model has been used in commercial services such as P2P TV and displayed its scalability [28].

Basically this model assumes that content is delivered from a few users to a large number of other users. This means that, if a large number of users tries to disseminate content simultaneously, it is very hard to handle them and sometimes it will take them out of service.

4.2.2 Reputation Aggregation

Needless to say, if web servers can be used, it is easy to aggregate reputations from users. However, it is not easy in P2P-based systems, because it is difficult to determine which peer should maintain aggregated reputations.

In GossipTrust [29], each peer has local scores representing reputations for all other peers and shares the local scores with randomly selected peers. PTrust [30] has tried to combat malicious peers, enabling peers to send occasional messages to lower the global scores of malicious peers as soon as they make mischief. These algorithms are effective only for reputation aggregation and do not have a function to distribute content. The circular board method, on the other hand, can work as reputation aggregation mechanism at the same time with content distribution mechanism. The details are described in Section 4.3.1.

Meanwhile, Aggregation Skip Graph [31] has enabled efficient execution of range query for aggregation in Skip Graph [32] which is a kind of distributed data structure based on skip lists. In particular, minimum and maximum values can be computed with fewer messages as the query range becomes wider. Range query in Skip Graph can also be used to distribute content. Section 4.5.5 discusses this point.

4.3 System Model

This section proposes the circular board method based on Chord. It provides an example of how it works to categorize transmissions of circular boards.

4.3.1 Chord-based Circular Board Method

Chord

In the communication layer, user terminals (hereinafter called peers) in the same cluster are connected with each other by Chord. Chord is a DHT algorithm and organizes a ring topology of peers. The simple ring topology is realized by only the successor list which each peer has. If the length of a successor list is 1, a peer only knows the next peer. If the length is 2, a peer knows the next peer and the peer after the next peer. The same goes for the following. Peers maintained by Chord also have special shortcuts to other peers in the *finger table* which enable a peer to communicate with distant peers. Successor lists and finger tables can be generated by the typical algorithm of Chord easily [19].

Besides, it is assumed that reorganization of Chord happens only when users join or leave the cluster. In other words, even if an online/offline status of peers has changed, reorganization of Chord does not happen as long as the user belongs to the cluster. Connection failure, which means peers tried and failed to communicate with offline peers, is ordinary and admissible in the proposed system. When a peer left permanently from a Chord network for some reason, its situation is detected by user management mechanisms in the cluster layer. At that time, reorganization of Chord will be prompted based on the notice from the cluster layer.

Circular Board Method

In the proposed system, UGC is treated by a circular board. The basic format of circular boards is as follows.

- Board ID
- UGC (text, image, and so on)
- Author information

  - General profile (user name and etc.)
  - Latest reputation
  - Peer ID

- Collection of reputations

A circular board has a board ID, UGC, its author information, and collection of reputations. The author information provides a general profile of the author and its latest reputation. A latest reputation would be presented by numerical scores or user ranks so that other users can estimate credibility of content. The peer ID of the author is used to determine the pathway of the circular board. The collection of reputations contains reputations from other peers the circular board has passed through.

When a user generates content, a circular board starts out from and arrives back in its peer. While the circular board flows among other peers along the topology of Chord, other users can watch and/or listen to them. The ring topology makes it possible to collect the reputation from other users at the same time, during the circular board passes through other peers. Note that users do not have to evaluate all UGCs.

The algorithm for consumer peers, the peers which receive circular boards, is indicated in **Algorithm 1**. Once a circular board arrives from other peers, onReceive() is called. In the procedure, the circular board which is already received is not processed but one which is not received before is passed to notice() so that presentation methods in the presentation layer can provide it to users. Users may give a reputation to the UGC in some cases using evaluateInfo(); the parameter will be a score or comments. Subsequently, the peer determines receivers using the peer ID of the circular board. getReceiver() returns a receiver list based on the settings, whether the finger table is enabled or disabled and how to sort receivers. The peer ID is used not to transmit the circular board to receivers across the producer peer (the author peer).

Algorithm 1 Procedures for Consumer Peers

```
 1: procedure onReceive(info)
 2:     if info is not already received then
 3:         notice(info)
 4:         receivers ← getReceiver(info)
 5:         for i ← 0, receivers.length do at prescribed intervals
 6:             send(receivers[i], info)
 7:         end for
 8:     end if
 9: end procedure
10:
11: procedure evaluateInfo(info, eval)
12:     info.add(eval)
13: end procedure
14:
15: procedure getReceiver(info)
16:     successor ← peers in the successor list of Chord
17:     if the finger table is enabled then
18:         finger ← peers in the finger table of Chord
19:         receivers ← successor + finger
20:     else
21:         receivers ← successor
22:     end if
23:     for i ← 0, receivers.length do
24:         if receivers[i] jumps over info.peer then
25:             Remove receivers[i] from receivers
26:         end if
27:     end for
28:     sort(receivers)                                ▷ Farthest first in principle
29:     return receivers
30: end procedure
```

Once receiver peers are determined, the peer starts passing the circular board to them at prescribed intervals. This interval will vary according to applied applications. Note that user evaluation of UGC can occur simultaneously when the passing is ongoing. In that case, the reputation is only transmitted to the remaining receivers which have not received the circular board yet.

The algorithm for producer peers, the peers which generate circular boards, is indicated in **Algorithm 2**. Once a user produces content, the peer determines receivers based on the settings, whether the finger table is enabled or disabled and how to sort receivers. It starts passing the circular board to them at prescribed intervals. When

Algorithm 2 Procedures for Producer Peers

```
1: procedure postInfo(info)
2:     receivers ← getReceiver()
3:     for i ← 0, receivers.length do at prescribed intervals
4:         send(receivers[i], info)
5:     end for
6: end procedure
7:
8: procedure getReceiver( )
9:     successor ← peers in the successor list of Chord
10:    if the finger table is enabled then
11:        finger ← peers in the finger table of Chord
12:        receivers ← successor + finger
13:    else
14:        receivers ← successor
15:    end if
16:    sort(receivers)                          ▷ Farthest first in principle
17:    return receivers
18: end procedure
19:
20: procedure onReceive(info)
21:    tallyReputation(info)
22: end procedure
```

the circular board arrives back in the peer, the reputations from users are tallied using tallyReputation(), which will calculate a score and so on. Note that a circular board may arrive back in the producer peer with some replicas which are duplicated in the passing process. One peer transmits a circular board to multiple receivers in the passing process. Each transmitted circular board can be regarded as replicas of the original circular board. In that case, all circular boards containing the same board ID are merged to one in the tallying process.

Example and Effective/Ineffective Transmission

An example of content distribution and reputation collection by the proposed method is illustrated in **Figure 4.1**. The small circles indicate user terminals. In the example, the length of a successor list is 3 and the finger table is not enabled. The user of peer S produces content at $t = 0$, and the peer S starts passing the circular board at $t = 1$.

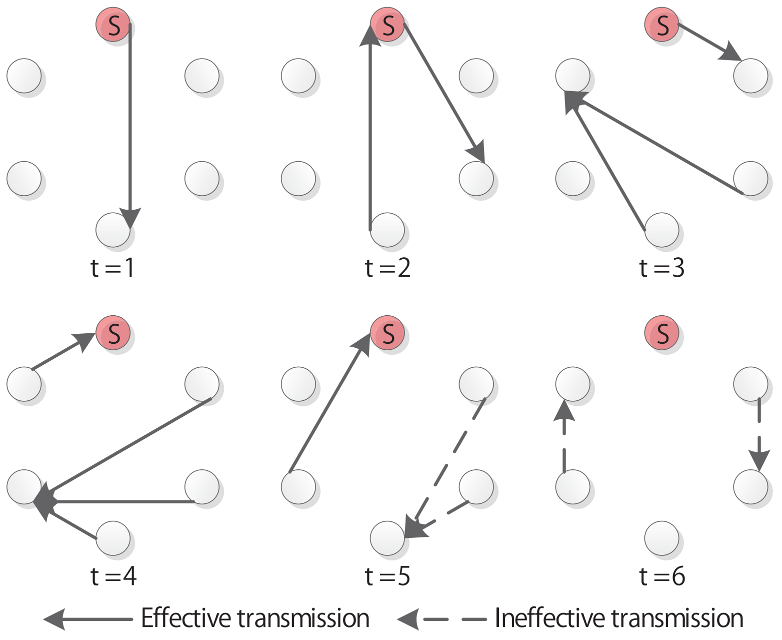


Figure 4.1: A Pathway of Circular Board

Every peer tries to transmit the circular board to the farthest peer in the receiver list for each interval. During the passing phase, reputations are collected as attachments of the circular board if users give evaluation on the content. When the circular board comes back to the peer S, the collected reputations are tallied.

Passings of circular boards can be categorized into two types, the effective transmission and the ineffective transmission. The effective transmission is defined as transmissions to uninformed peers or transmissions which finally lead to the producer peer. The former contributes to content distribution, while the latter contributes to reputation. The other transmissions are defined as the ineffective transmission. Note that if some effective transmissions (in a fail-proof condition) are failed for any reason, those transmissions are treated as just connection failures. The ineffective transmissions (in a fail-proof condition) would be treated as effective ones.

4.4 Simulation

First indices are defined to be used to observe and compare the network efficiency for content distribution and reputation aggregation. Through the use of the indices, the relationship between the connection failure rate and other settings is investigated so that the proposed system can be applied to a practical application with proper settings; the efficiency is largely dependent on an average connection failure rate which will be estimated from features of the underlying networks and/or user behaviors. Since it is difficult to prepare real environments with different connection failure rates, a simulator has been implemented.

4.4.1 Indices of Efficiency

The network efficiency at the time interval t can be computed as below.

$$E(t) = \frac{w_{ip} \cdot ip_t + w_{tr} \cdot tr_t}{p} \cdot \frac{w_{et} \cdot et_t}{w_{et} \cdot et_t + w_{it} \cdot it_t + w_{cf} \cdot cf_t} \tag{4.1}$$

p is the number of peers in the cluster. ip_t stands for the number of informed peers (peers which received a circular board), tr_t the number of tallied reputations, et_t the number of effective transmissions, it_t the number of ineffective transmissions, and cf_t the number of connection failures at the time interval t, respectively. If the terms are expressed without the subscript t like ip hereafter, that means the final number of each term like the total number of finally informed peers. w_{ip}, w_{tr}, w_{et}, w_{it} and w_{cf} represent a weight for ip, tr, et, it and cf, respectively. The early part of the formula represents the coverage of informed peers and tallied reputations. The more peers are informed and reputations are tallied, the higher the network efficiency is. The latter part of the formula means that it is better to decrease the number of ineffective transmissions and connection failures.

The network efficiency is only comparable in the same sequence of content distribution and reputation aggregation, because if any settings such as the length of a successor list is changed, the number of effective transmissions changes, for instance. To figure out which setting is the most efficient in total, the relative total efficiency is proposed as below.

$$E = C \cdot E_{net} \cdot E_{time} \tag{4.2}$$

Each term is formulated as below.

$$C = \frac{w_{ip} \cdot ip + w_{tr} \cdot tr}{p} \tag{4.3}$$

C is the coverage of informed peers and tallied reputations.

$$\begin{aligned} E_{net} = w_{et}(1 - \frac{et}{et_{max} + 1}) + w_{it}(1 - \frac{it}{it_{max} + 1}) \\ + w_{cf}(1 - \frac{cf}{cf_{max} + 1}) \end{aligned} \tag{4.4}$$

E_{net} is the relative total network efficiency. et_{max}, it_{max} and cf_{max} represent the maximum number of et, it, and cf, respectively, from results which need to be compared. This formula means that the fewer the number of transmissions becomes, the higher the network efficiency is.

$$E_{time} = 1 - \frac{tm}{tm_{max} + 1} \tag{4.5}$$

Likewise tm_{max} represents the maximum number of tm from results which need to be compared. This formula means that it is better if time required for content distribution and reputation aggregation gets faster.

4.4.2 Simulation Setting

Settings for the simulation are as follows.

Number of peers

The simulations have been done in the case when the number of peers is 2^{10}, 2^{13}, and 2^{16}. One of the peers works as a producer peer, while others work as consumer peers. Although the number of peers does not change in a simulation dynamically, the offline status of peers is treated as the connection failure as mentioned later.

Chord

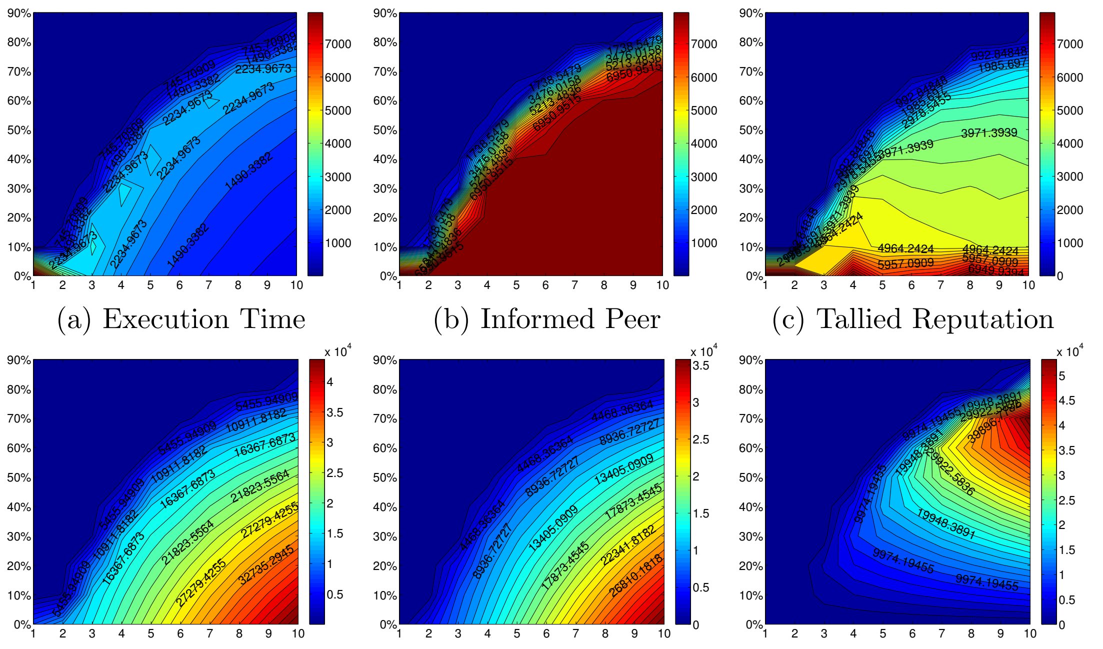
The length of a successor list ranges from 1 to 10. When the finger table is disabled, only the successor list is used to determine receivers. If the finger table is enabled, peers in a finger table are also treated as receivers. Hence, some peers in a successor list may overlap with peers in a finger table. In that case, peers in the finger table which also appear in the successor list are not used. Besides, it is assumed that peers are uniformly distributed in the hash space of Chord (160 bits) for making a finger table operate efficiently; Chord has a technique called the virtual peer to achieve this assumption [19].

Connection Failure Rate

The connection failure rate ranges from 0% to 90%. This parameter directly affects the probability of connection failure when each peer tries any transmission. The assumed causes of the connection failure contain the offline status of peers and network failures.

4.4.3 Simulation Scenario

At the beginning of a simulation, peers are generated with the ring topology of Chord organized according to the settings. Next, a circular board starts flowing from the producer peer. At this moment, an elapsed-time counter starts. When each peer receives the circular board, the content is given an evaluation immediately within an interval. In the passing phase, each peer tries to transmit the circular board in order, from the farthest peer in candidates, to receivers which the peer has not transmitted the circular board yet. The reason for the farthest-first transmission is because the results of preliminary experiments show that the farthest-first transmission is the most suitable from the view point of time required for content distribution and reputation



(a) Execution Time (b) Informed Peer (c) Tallied Reputation

(d) Effective Transmission (e) Ineffective Transmission (f) Connection Failure

Figure 4.2: Successor Only - 2^{13} Peers

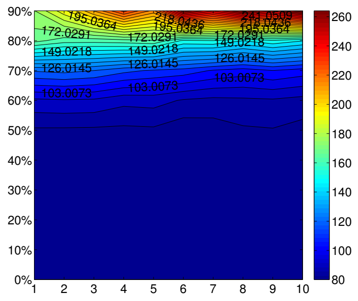
aggregation. Additionally, for each unit time (interval), the number of informed peers, tallied reputations, effective transmissions, ineffective transmissions, and connection failures are recorded.

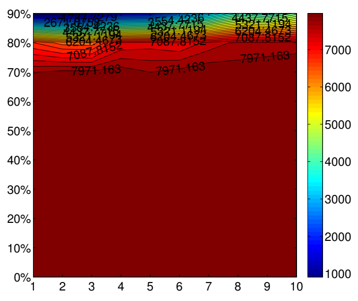
Simulations stop when every peer finishes all transmissions for its receivers or becomes static in some conditions. Each combination of the settings is repeated 100 times and the averages are calculated.

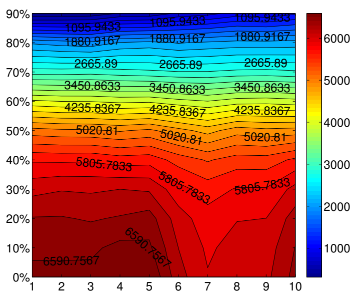
4.4.4 Result

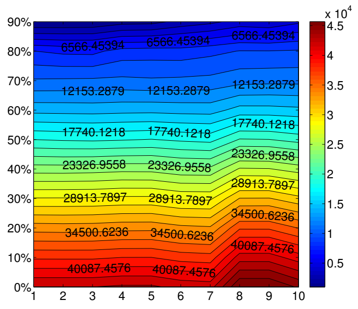
The results of the simulation for 2^{13} peers are illustrated in **Figure 4.2** and **Figure 4.3**. The finger table has been disabled in Figure 4.2, while enabled in Figure 4.3. In each contour plot, the horizontal axis, the vertical axis, and the contour represent the length of a successor list, the connection failure rate, and the counts for each index, respectively.

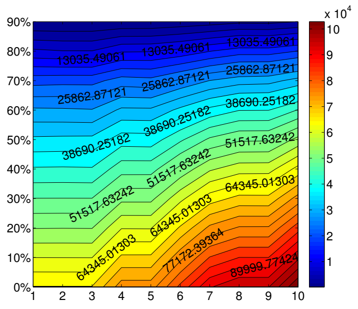
Focusing on simulation time, although the upper left part of Figure 4.2 (a) indicates


(a) Execution Time


(b) Informed Peer


(c) Tallied Reputation


(d) Effective Transmission


(e) Ineffective Transmission

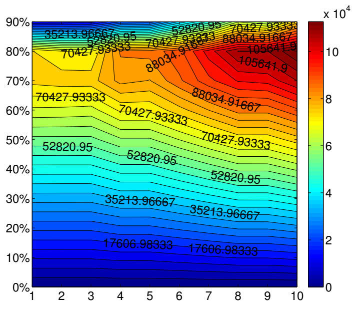

(f) Connection Failure

Figure 4.3: Successor with Finger Table - 2^{13} Peers

that the simulations have finished in a short time under a condition where the length of a successor list is short and the connection failure rate is high, this indication does not mean that content distribution and reputation aggregation have finished in a short time as shown in Figure 4.2 (b) and (c). It seems that circular boards cannot be transmitted to the receivers during the early stage and the all peers become static under the condition.

Figure 4.2 (b) and (c) show that the number of tallied reputations is influenced more strongly than the number of informed peers by the connection failure rate. In contrast, Figure 4.2 (d) and (e) indicate that the distribution of the effective transmission count is almost similar to that of the ineffective transmission count. The number of connection failures is simply controlled by the connection failure rate as seen in Figure 4.2 (f).

In Figure 4.3 (c), the number of tallied reputations decreases when the connection failure rate is low and the length of a successor list is around 7. This type of phenomenon is also seen when the number of peers is 2^{10} and 2^{16}. As the number of peers increases, the phenomenon area moves in a direction from the right to the left. This phenomenon may be caused by a combination of the number of peers and the finger table which organizes a specific topology. This phenomenon is not seen when the connection failure rate is high. A high connection failure rate seems to destroy a topology of peers virtually.

In Figure 4.3 (e), the lower right part is higher than the lower left part, because the number of transmission trials increase as the length of a successor list is getting long when the connection failure rate is the same. In Figure 4.3 (f), the same holds true for the upper left part which is lower than the upper right part.

In comparison with Figure 4.2, Figure 4.3 shows that the finger table enables very fast content distribution and reputation aggregation and provides a high tolerance for the connection failure. However, it requires a lot of transmissions which are several times higher.

The relative total efficiency for Figure 4.2 and Figure 4.3 is calculated as **Figure 4.4** and **Figure 4.5**, respectively. Again, the finger table has been disabled in Figure 4.4, while enabled in Figure 4.5. A weight for the number of connection failures is fixed to 0.5, and the others to 1. These weights are chosen under the assumption that a cost of the connection failure is about half of the effective/ineffective transmission, and the other concerns have the same level of importance. Moreover, in order to make the values of efficiency easier to understand, each value is normalized under the condition which makes the maximum 1.

When the finger table is disabled, the efficiency becomes the maximum under the condition that the connection failure rate is 0% and the length of a successor list is 4, according to Figure 4.4. This tendency does not change even if the number of peers is 2^{10} or 2^{16}. In contrast, when the finger table is enabled, a maximum area, as seen in

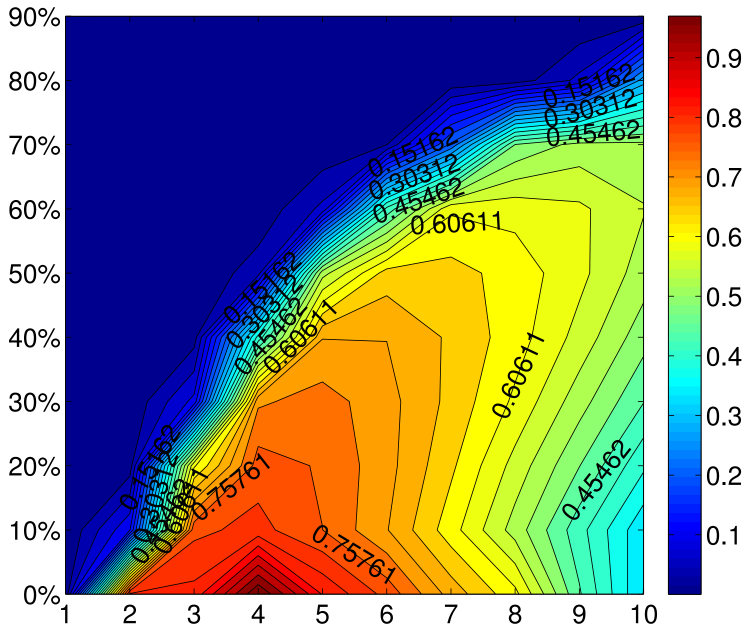


Figure 4.4: Total Efficiency - Successor Only - 2^{13} Peers

Figure 4.5, has moved in a direction from the bottom to the top with the increase of the number of peers.

Although the other results are omitted, they have revealed the same tendency of Figure 4.2 through Figure 4.5. Characteristic results have been comparable to the ones stated above. All of the results are shown in Appendix.

4.5 Discussion

4.5.1 Employment in Real Environment

According to a survey [33], from January 15th to March 19th in 2013, a success rate of packet communication between smartphones and web servers is about 96% in Japan. In other words, a success rate of packet communication between a pair of smartphones will be about 92%, which can be calculated by 96% $\times$ 96%. It corresponds to the connection failure rate of 8%.

If the Japanese mobile telephone network is used as the underlying network of the

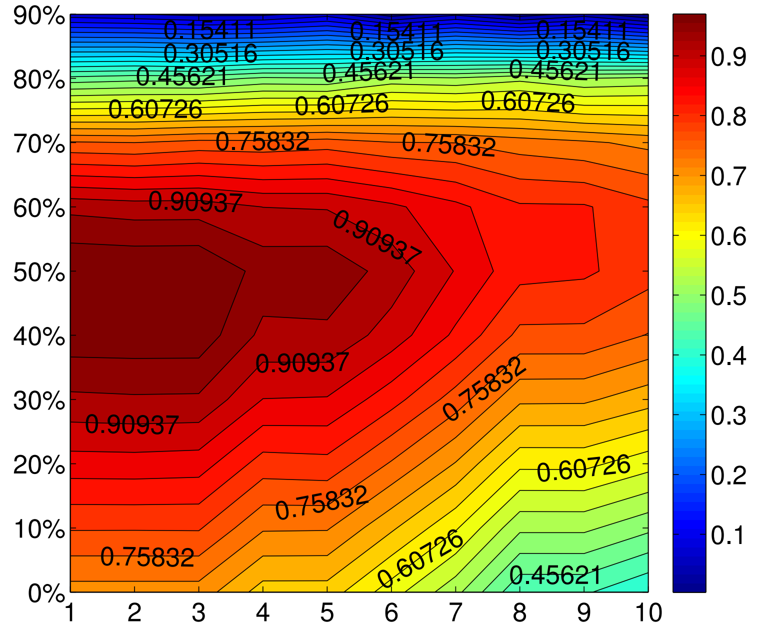


Figure 4.5: Total Efficiency - Successor with Finger Table - 2^{13} Peers

proposed system, with 2^{13} peers, the proper length of a successor list is 4 when the finger table is disabled, or 3 when the finger table is enabled. **Figure 4.6** represents the number of informed peers, tallied reputations, and the network efficiency for each of the conditions. To calculate the network efficiency, a weight for the number of connection failures is fixed to 0.5, and the others to 1. Note that the network efficiency is not comparable with a different sequence as mentioned before. It is normalized under the condition which makes the maximum 1. In addition, the horizontal axis for time is logarithmic.

When the finger table is enabled, the number of informed peers and tallied reputations increases in a similar way as the time goes. On the other hand, when the finger table is disabled, although the number of informed peers increases as the time goes, the number of tallied reputations does not increase until just before the last part of the graph.

From the view point of the network efficiency, an apparent peak can be seen when

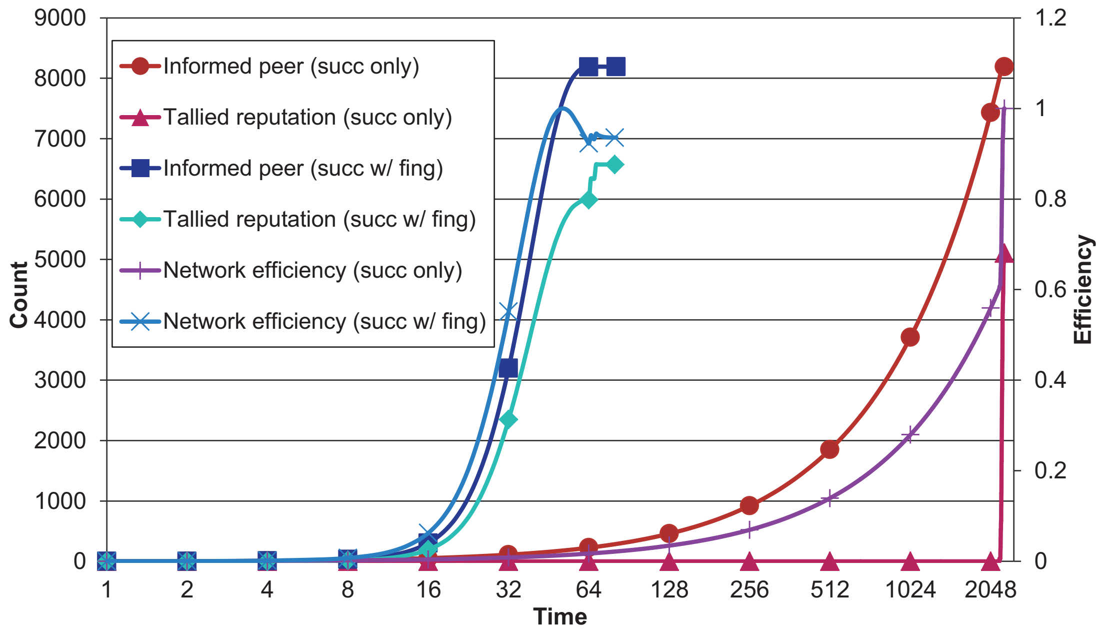


Figure 4.6: Effect of the Finger Table

the finger table is enabled. This means that many ineffective transmissions and/or connection failures occur in the last part of the simulations. In contrast, such peak cannot be seen when the finger table is disabled.

Obviously, the finger table greatly reduces the time required for content distribution and reputation aggregation. For instance, if the unit time (interval) is 10 minutes, it takes 13.5 hours with the finger table, while 384.0 hours without the finger table to distribute content and aggregate reputations in 2^{13} peers on average. From a viewpoint of network resources, the finger table causes 3.38 times transmissions including effective transmissions, ineffective transmissions, and connection failures. It depends on each operation policy of individual services whether the increase is allowable or not in the network cost. In Twitter, 75% of retweets (the user operation to diffuse information to other users) occurs within one day and about 10% take place a month later [34]. Retweeted information can be regarded as meaningful information at that time. In other words, if the proposed method is applied to a Twitter-like service, most information should be diffused within one day. This could be a trade-off between the required time and network resource which is consumed.

It should be also mentioned about the feasibility of the proposed method with poor terminals such as smartphones. One of the possible implementations would use WAP Push API [35] which is typically used by service providers to push information to consumers like e-mail and flash news. Using this API, smartphones can send/receive information through HTTP POST connections. According to a performance survey [36], the realistic maximum number of HTTP connections is 17 in Android 4.x terminals, 23 in iPhone 5, and more than 60 in Windows Phone 7/8 terminals. In the proposed method, a peer transmits a circular board in series not in parallel as seen in lines 5-7 of Algorithm 1. This means that only one HTTP connection is required for each user cluster to pass around circular boards. Consequently, it is believed that the proposed method works practically with smartphones even if a user belongs to multiple—but not so many—clusters under the reasonable data size of content and the properly selected unit time (interval). In regard to the network cost of maintaining Chord, the influence of a joining/leaving terminal is limited to several associated terminals due to a feature of Chord [19]. Moreover, reorganization of Chord only happens when users join or leave the cluster and do not happen when an online/offline status of peers just has changed as mentioned in Section 4.3.1. For these reasons, although it is hard to estimate how often reorganization of Chord is required, it is predicted that its frequency is very low compared with the transmission of circular boards and smartphones can handle it adequately.

4.5.2 Collection of Reputations

In the simulation, it is assumed that all users give evaluation on the content immediately within an interval when each peer receives a circular board. This assumption would not be realistic.

		# of peers		
		2^{10}	2^{13}	2^{16}
Maximum error	0 %	1024.00	8192.00	65536.00
	1 %	964.70	5489.81	13271.33
	2 %	821.90	2759.30	3911.98
	3 %	659.25	1508.67	1798.29
	4 %	516.23	922.99	1023.83
	5 %	403.65	615.69	658.96

Table 4.1: Number of Required Reputations

Table 4.1 shows the thresholds of required reputations to guarantee a level of statistical significance. The thresholds are calculated under the common assumption in statistics that the confidence level is 99% and the population proportion is 50%. If the number of tallied reputations exceeds the thresholds, the reputations can be treated as a unified reputation of all users with each confidence level.

As the number of peers increases, the number of required reputations also increases. However, its amount of increase gets smaller. In other words, the number of required reputations converges to a particular value with the same maximum error rate. For example, if a maximum error rate is 5%, the number of required reputations converges to about 664.

How many users evaluate content and how much accuracy is necessary depend on an application. For instance, in the example of Section 4.5.1, up to 6572.81 reputations can be tallied with the finger table and 5101.68 without the finger table on average. If 25% of users evaluate within an interval, these numbers turn out to be about 1643 and 1275 which guarantee a maximum error rate of 3% and 4%, respectively. This could be another trade-off.

4.5.3 Surrogate for Producer Peer

Although the offline status of producer peers is treated as the connection failure in the simulation, in actual situations, some producer peers may go offline for a very extended period of time after the users generate content. One simple solution is to prepare surrogates such as super peers and web servers. If transmissions to producer peers fail, the transmitters can simply transfer them to the surrogates with ID of the producer peer. When the producer peers get back online, they can get their reputations from the surrogates.

The super peer can be elected from peers based on the performance of CPU, the capacity of network and the duration time of online. Multiple super peers would be required to deal with network failures and sudden defection of some super peers. Alternatively the cloud can be used as mention in the introduction. It can also solve the bootstrap problem, how to find an existing peer, and enable the user management like expulsion of immoral users easily which often cause trouble in P2P-based systems.

4.5.4 Content Distribution Network

In some cases, it is not realistic to transmit a vast amount of content like high quality videos directly by content distribution systems including the proposed system in this chapter. For handling that kind of content, a combination of a content delivery network (CDN) and content distribution systems is one of the most potent solutions. CDN enables a large number of consumers to download a vast amount of content simultaneously [37].

For instance, a pair of a content summary and a URI link to the body of the content may be distributed and the body may be located in CDN-assisted hosts. If web servers can be used, CDN postulating the Web such as OpenWeb which is presented in Chapter 5 would be proper. Alternatively, CDN postulating P2P [38] would be proper if web servers cannot be used or are not allowed to be used for some reason.

In the future, if the next generation network (NGN) has been well-wired, it may bear a chance of transmitting a vast amount of content directly.

4.5.5 Using Chord

To my knowledge, there is no versatile approach to enable content distribution and reputation aggregation simultaneously in distributed systems. If content distribution is simply required, application layer multicast or flooding would be the most common and efficient means. For that purpose, Pastry [39] which has a mesh topology and/or Kademlia [40] which has a tree topology must be appropriate. Additionally, in some structured overlay networks such as Skip Graph [32] and Chord# [41], range query is available for content distribution; efficient range query for aggregation is also proposed as mentioned in Section 4.2.2. However, when and who should invoke aggregation query could be another difficult issue.

The most important reason of employing Chord is its simple ring topology. Using Chord, a pathway of a circular board is directly corresponding to its topology. This means that content is distributed to consumer peers and reputations are collected at producer peers just by forwarding circular boards along the topology once. If non-ring topologies are used, at least pathway control mechanisms are absolutely necessary. It would be a thorny issue to collect distributed circular boards in a P2P manner. Again, the author believes that the most significant part of this work is to enable content distribution and reputation aggregation simultaneously. Performing content distribution and reputation aggregation in different phases is no more than a combination of existing works introduced in Section 4.2.1 and 4.2.2 or the well-known approaches of overlay networks described above.

4.6 Summary

This chapter has proposed to apply the circular board method based on Chord to user centric media to distribute content and aggregate reputation efficiently in a P2P manner. The proposed system makes it possible to effectively collect the reputation from users at the same time when each piece of UGC passes through user terminals. The results of simulations have provided insights about trade-offs between network resource consumed and time required for content distribution and reputation aggregation. As future work, it should be considered to customize the original topology of Chord and explore other topology such as layered P2P networks [42] to make the efficiency higher.

Chapter 5

Seamless Content Delivery

5.1 Purpose

If cache systems work at the switching layer (layer-2), administrators can introduce the system just with its insertion into the network so that clients can use the system transparently. It is considerably robust. Using the *OpenFlow* [43] switch architecture, it is possible to implement a cache system which works at the switching layer. The OpenFlow is a new programmable switch abstraction defined by the OpenFlow Switch consortium centered at Stanford University. The overall intent of OpenFlow is to separate the control plane of a switch from its data plane, which makes the control plane programmable.

This chapter proposes to use new advances at the switching layer to redirect requests of clients to cache systems seamlessly and effortlessly. Technically, it introduces OpenWeb, an overlay transfer application of the OpenFlow programmable switching layer. OpenWeb can provide good user experience and server load balancing without any troublesome user configuration and difficult-frail administrator management on user demands.

5.2 Intended Content and Related Work

This section provides a brief overview of web content and cache systems.

5.2.1 Web Contents Handled by Cache Systems

Generally, there are two types of Web content handled in the field. One is static content and the other is dynamic content. The difference between these two types of content is whether the same content is delivered to all users by the same URL overtime. In the case of static content, the same content is delivered to all the users exactly as it is stored. For example, HTML documents, cascading style sheets, and image files are typical static content. On the other hand, in the case of dynamic content, content is generated and delivered to the users in accordance with the interaction of each user, the time, and so on. For example, personalized pages and frequently-updated pages such as the top pages of news portals are typical dynamic content.

Although cache systems mainly handle static content, some dynamic content can be handled by cache systems considering a certain unit time. For instance, news articles and blog entries can be cached for a certain time. Moreover, even if a web page needs to be generated dynamically, the page will contain some static content such as template HTML documents, image files, and so on. These kinds of content also can be handled by cache systems.

5.2.2 Content Delivery Systems

The past decade has seen the gradual introduction of content delivery systems. From the viewpoint of using data cache, these systems can be seen as a type of cache system. They are divided into 3 types.

Content Delivery Network

CDNs seek to provide scalable and high-performance delivery of block data, notably including web pages, saving load on both clients and original hosts, namely a web server which has original content. Examples of CDNs are Akamai [37], CoDeen [44] [45] [46], and Coral [47].

Some CDNs require several configurations of clients to use the systems. This means that the effect of the systems depends on client behavior. Although other CDNs do not require any configuration of clients, server administrators need to rewrite DNS records which are used to resolve web servers' IP address. This means that the servers are inseparably connected with the cache systems. These kinds of cache systems are expected to be independent from web servers for robustness.

Peer-to-Peer Service

P2P services seek to provide high-performance and robustness in content distribution/sharing among many collaborating clients. Examples of P2P services are BitTorrent [48], Antfarm [49], and Squirrel [50].

Some P2P services achieve very good performance. To use P2P services, however, clients need to install special softwares. Moreover, cache management in P2P services is difficult to be implemented because clients with data cache may join and leave whenever they like. Although there are some cache management methods such as distributed-hash-table algorithms and simple management by a central server, these methods sacrifice performance and scalability. Note that some methods even have a single point of failure. For these reasons, it implies that clients should not do anything special to benefit from the cache systems, assuring the cache systems should keep its high performance and scalability.

Storage System

Storage systems seek to provide distributed storage services for high-performance content distribution. Examples of storage systems are Amazon S3 (Simple Storage Service) [51], OceanStore [52], and the Internet Logistics Service [53]. Storage systems can be regarded as a storage-specific CDNs.

Obviously, to benefit from these storage systems, data must be stored in these systems, besides the same problems with CDNs.

5.2.3 Other Related Works

As other existing works which are similar to the proposed method in this chapter, there are at least two efforts at the application layer, and one transport-layer protocol.

Ocala [54] is an overlay convergence layer sitting below the transport layer and above the IP layer in the host networking stack. Its fundamental purpose is to intercept networking API calls to redirect them to the appropriate service. It consists of an *overlay-independent* layer, which services all registered overlay services, and an overlay-dependent layer, which formats specific and appropriate requests for the specific service. Ocala was demonstrated on a NAT traversal service and a secure connection service.

Oasis [55] is, similarly, a modified networking layer below the transport layer on the end-host IP stack. Oasis concentrates heavily on adaptive, rule-based routing of end-user packets onto a number of overlays; the rules are sufficiently general to support a wide variety of application protocols.

The WCCP (Web Cache Communication Protocol) [56] is a Cisco-systems inspired protocol offered on a number of Cisco routers. It redirects traffic for the web to a group of web servers, utilizing load balancing. However, this is available only on router products, not on switches definitely.

To best knowledge of the author, the work described in this chapter is the first effort to offer web redirection at the switching layer, with neither of use of modification to end-host code, nor requirement of IP redirection. Of course it is not necessary to rewrite DNS records.

5.3 OpenFlow

The OpenFlow programmable switch abstraction is used to implement a cache system which works at the switching layer. This section introduces the OpenFlow framework based on the OpenFlow white paper [43].

5.3.1 Feature

An OpenFlow switch adds a *controller* to a network of switches, which is a single commodity server connected to each switch in the network over a secure connection. The controller acts as the control plane for the switches, directing packets on a network-wide basis. Of course, the controller cannot make packet decisions at line speed. The switches provide the controller with a data abstraction which permits the controller to program the switches on a coarser grain than the individual packet.

The basic abstraction in OpenFlow is a *flow*, regarded as a single TCP session or a stream of UDP packets from a single source to a single destination, over a single TCP/UDP port. The OpenFlow switch maintains a *flow table*, which contains the directives to route and manage flows. The controller sends the OpenFlow switch rules for processing flows. This instruction is very powerful, as it permits the controller not only to make a packet-by-packet decision but also to update the switch flow table for subsequent processing of further packets in this flow or packets in similar flows.

5.3.2 Implementation

OpenFlow has been implemented by a number of vendors on their switch lines, notably including NEC, Cisco Systems, Juniper Networks, and Hewlett-Packard. The level of support for OpenFlow and the efficacy of the implementation varies from vendor to vendor; NEC has announced full support for OpenFlow as a product offering, and HP has made a significant effort achieving a robust OpenFlow implementation as a research project with high bandwidth and extensive support.

In addition, NetFPGA [57] can be used as an OpenFlow switch. NetFPGA is a kind of layer-2 programmable switch based on FPGA (Field-Programmable Gate Array). Note that NetFPGA is made not only for OpenFlow but also all network modules that use hardware rather than software to forward packets.

5.3.3 Example Application

OpenFlow has been the platform for a wide variety of innovative networking projects. Some examples follow.

- Network Management and Access Control

  A canonical example of OpenFlow is Ethane [58]. Indeed, OpenFlow is essentially the network portion of Ethane. Ethane enabled the central definition of network-wide policy on access control and quality of service. The controller checks a new flow against a set of rules, such as:

  - Guests on the network can communicate using HTTP, but only via a specific web proxy;
  - Only IP addresses belonging to human resources can access the personnel database;
  - VOIP phones are not allowed to communicate with laptops.

- Multipathing in Wireless Networks

  It enables clients to achieve good quality of service such as bandwidth and fault tolerance in wireless networks.

- Virtual Machine Migration

  It enables server administrators to migrate virtual machines from an old server to a new server while keeping IP addresses of virtual machines constant.

There are a lot of other examples. OpenFlow has been used for a lot of advanced network researches.

5.4 OpenWeb

In this section, OpenWeb is proposed as a new open protocol at the switching layer (layer-2). This protocol enables far more robust and seamless packet redirection without configuration alternation or unreliable scripts by users. OpenWeb is a layer-2 redirection engine implemented as an application of the OpenFlow switch architecture.

5.4.1 Concept

The goal of OpenWeb is to enable users to use the cache system without any configurations, and administrators to manage the cache system easily and robustly. In order to achieve this goal, three things described below should be kept in mind.

Independency

The system should be independent from web servers and can handle unexpected requests. This is quite different from traditional systems. For instance, in DNS-based systems, administrators need to rewrite DNS records which are used to resolve IP address of web servers before requests come to the web servers. If administrators cannot predict the flood of requests and rewrite DNS records, the systems cannot work well. However, unexpected requests can be possibly handled if a system can work independently from web servers.

Transparency

The system should be able to be used without any troublesome configurations by clients. This means that the effect of the cache system does not depend on client behavior; some clients cannot/do not install special softwares, neither change configurations. This is quite different from traditional systems such as BitTorrent or Coral.

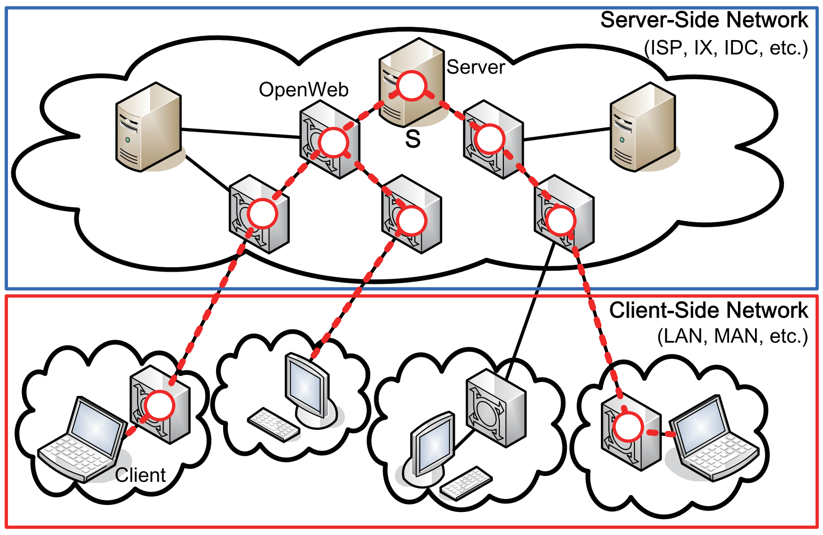


Figure 5.1: Working Image of OpenWeb

Performance

Processing performance should be the same as traditional systems because it is directly linked with user experience.

Working image of OpenWeb is illustrated in **Figure 5.1**. This image just represents the servers, the OpenWeb systems and the clients on the Internet. Requests from the clients go through an array of the OpenWeb systems. OpenWeb is a kind of web proxy systems which works between the clients and the servers at switching layer. This means that if an OpenWeb system has cache of requested content, content is delivered to the user by the OpenWeb system. Additionally, the clients can use OpenWeb systems transparently.

If sufficient amounts of OpenWeb components are arranged in the Internet uniformly, OpenWeb will autonomously organize a CDN of a tree structure which represents paths from clients to a server, which is shown as server S in Figure 5.1, determined by IP routing. In the CDN, a server which has original content is a root, OpenFlow switches are nodes, and clients are leafs. Moreover, the CDN tree structure does not

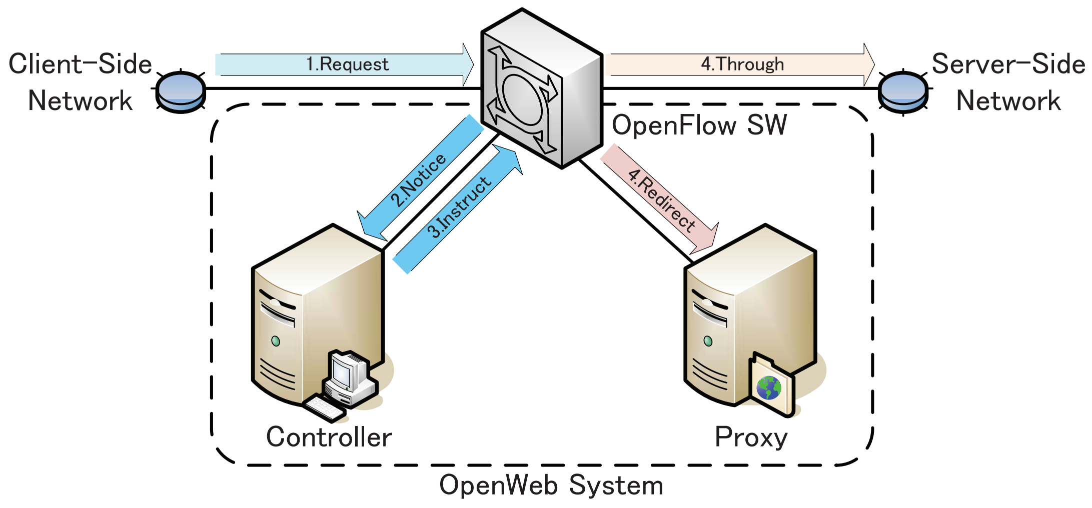


Figure 5.2: A Typical Architecture

stay in application level but in network level. This CDN enables scalable and high-performance delivery of content, saving load on web servers and providing good user experience.

5.4.2 Architecture

A general architecture of OpenWeb is illustrated in **Figure 5.2**. At least one OpenFlow switch, one controller, and one proxy server are required in one OpenWeb system. When the switch detects a request from the client-side network, the switch notifies it to the controller. Then the controller checks the request if it needs to be redirected or not. According to a decision of the controller, the switch puts the request through to the server-side network or redirects the request to the proxy server.

The role of each component is as follows:

- Proxy Server

  General proxy servers are used so far. It processes requests from clients on behalf of web servers. (In the future, other advanced functions such as application containers will be introduced to support real-dynamic content.)

- OpenFlow Switch

  The OpenFlow switch is the core of the OpenFlow platform and OpenWeb. It is a layer-2 switch managed under the controller as described in Section 5.3.

- Controller

  The controller determines whether to redirect a request using the redirection methods described in the next section. This decision can be made according to static configurations by administrators or dynamic configurations based on access trends.

Besides the components listed above, clients and servers which have original content would belong to their own networks. Clients are normal computers used by web users and servers are normal computers managed by server administrators. Note that this server administrators are different from administrators of OpenWeb systems; although they can be an administrator of both normal servers and OpenWeb systems at the same time naturally. Of course, no proxy configuration is required on clients and no server configuration is required on servers to use the system because the system works at layer-2. Moreover, clients/servers and OpenFlow switch can belong to either the same network or different networks.

5.4.3 Redirection Method

The redirection methods determine whether to redirect a request to a proxy or transfer it to a server. Two types of packet redirection methods are formulated for OpenWeb. One is the IP-based redirection, while the other is the URL-based redirection. In both methods, after a two-way redirection between a client and a proxy is established, the communication path of client is redirected to the proxy server until redirection rules become time out. Once the request is decided to be redirected, an OpenFlow switch redirects the request to a proxy server using NAT (Network Address Translation) or

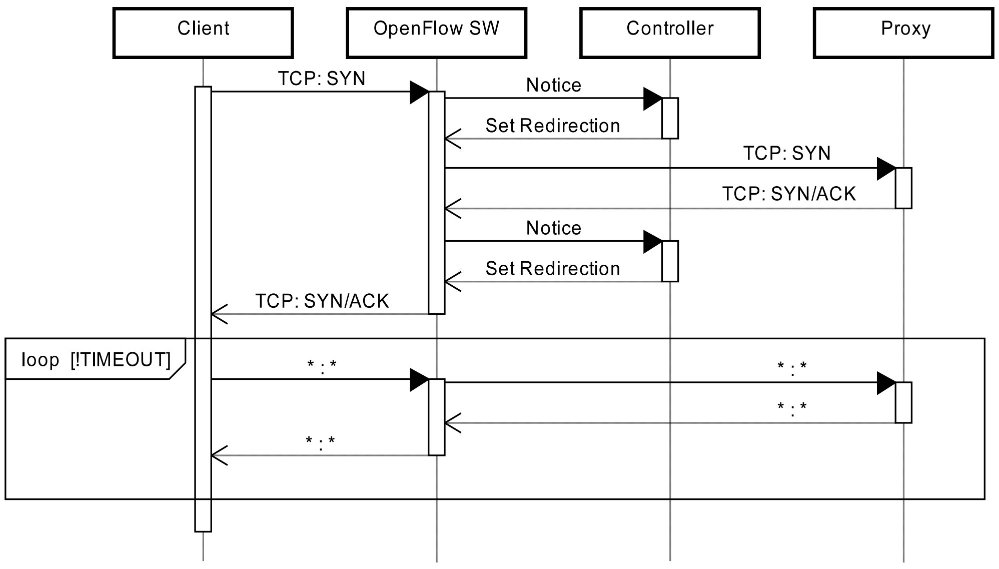


Figure 5.3: IP-based Redirection

NAPT (Network Address Port Translation).

IP-based Redirection

A traditional web server is associated with one IP address per one host name. Therefore, the simplest way of packet redirection is to redirect a packet which contains an IP address of a host toward a proxy server as its destination.

Typical packets flow of the IP-based redirection is illustrated in **Figure 5.3**. First, the client sends a SYN (TCP) packet to a host. The OpenFlow switch notifies the receipt of the packet to the controller. The controller looks into the packet. If the destination IP address of the packet is redirection target, the controller sets a rule to the switch to rewrite the destination IP address of the packet with the IP address of the proxy server. In the IP-based redirection, a list of target hosts are prepared as a list of IP-addresses as a condition of redirections. The proxy server replies with a SYN/ACK (TCP) packet to the client. Likewise, the switch notifies the receipt of the reply packet to the controller, which makes the controller set a rule to the switch to rewrite the source IP address of the packet with the IP address of the host. In this way, a two-way redirection is established.

URL-based Redirection

Nowadays, many web sites employ the DNS round robin for load-balancing and the virtual hosts for server resources sharing. It means that there can be several web servers (or IP addresses) for one host name and several host names for one web server (or IP address). The IP-based redirection is not suitable for these situations because making redirection rules one by one is not realistic and far from robustness. For instance, it is almost impossible to know all IP addresses of hosts operated by a DNS round robin system. To solve this problem, the URL-based redirection is proposed. In the URL-based redirection, the controller uses not the destination IP address of the packet, but the URL of the HTTP request.

Typical packets flow of the URL-based redirection is illustrated in **Figure 5.4**. First, the client sends a SYN (TCP) packet to a host. The OpenFlow switch notifies the receipt of the packet to the controller. The controller checks if the packet is an HTTP packet or not, looking at the port number of the packet. If the packet is not an HTTP packet, it passes the controller through. If the packet is an HTTP packet, in URL redirection, the controller analyzes the body of the packet. This time, because there is no content in the body of the SYN (TCP) packet, it passes the controller through. When the packet comes to the host, the host replies with a SYN/ACK (TCP) packet to the client. Likewise, the switch notifies the receipt of the packet to the controller, which the packet passes through. An ACK (TCP) packet sent by the client is delivered to the host in the same way.

After that, the client sends a GET (HTTP) packet to the host. The switch notifies it. This time, the controller finds the GET command in the body of the packet. The controller looks into it. If the URL of the request is a redirection target, the controller

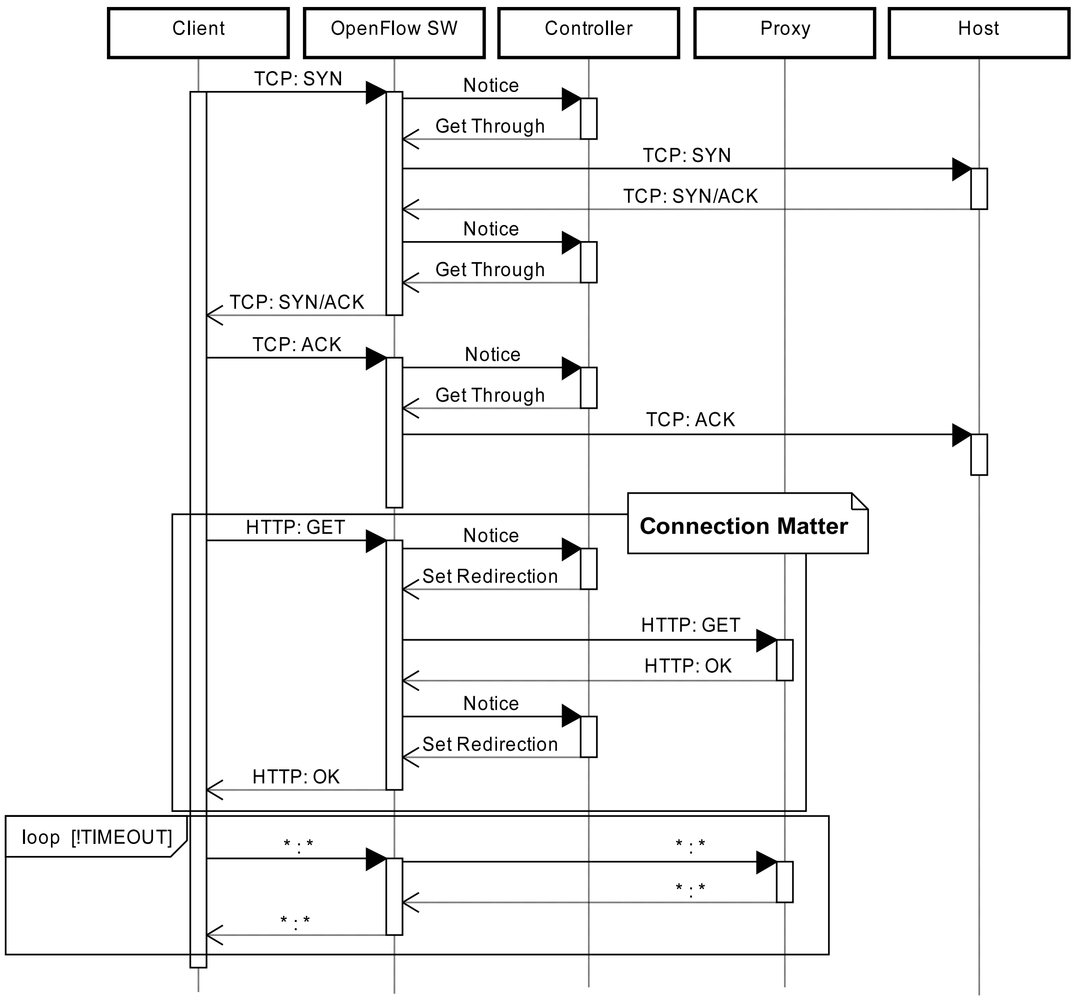


Figure 5.4: URL-based Redirection

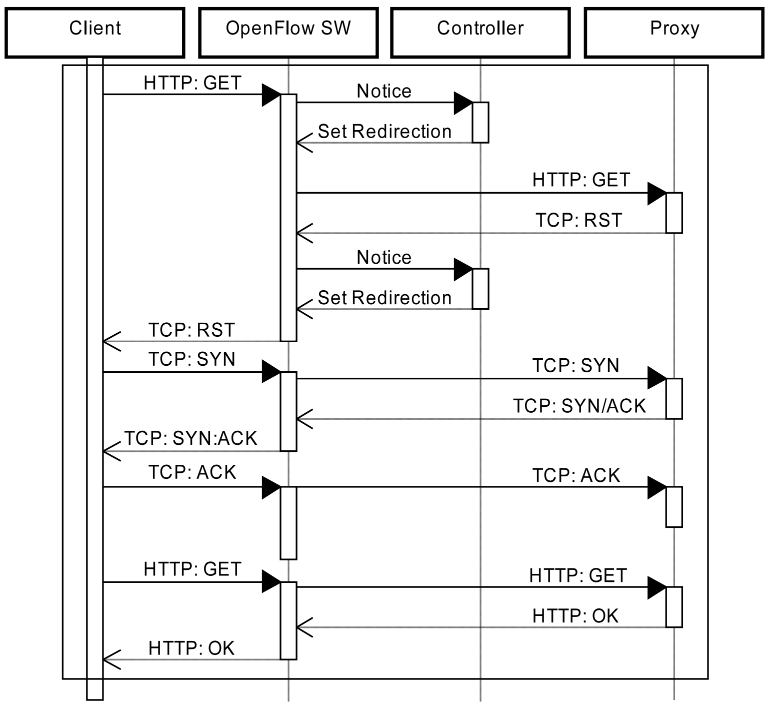


Figure 5.5: Connection between the Client and the Proxy

sets a rule to the switch to rewrite the destination IP address of the packet with the IP address of the proxy server. In the URL-based redirection, a list of target hosts are prepared as a list of URLs as a condition of redirections. The proxy replies with an OK (HTTP) packet to the client. Likewise, the switch notifies the controller of receiving of the reply packet, which makes the controller set a rule to the switch to rewrite the source IP address of the packet with the IP address of the host. In this way, a two-way redirection is established.

To be exact, although the basic idea of the URL-based redirection method is as explained above, there is a connection matter between the client and the proxy (see the noted rectangle in Figure 5.4). That is how to enable the client to communicate with the proxy. Because a connection between the client and the proxy is not established at first, it is impossible to make the proxy process the GET (HTTP) packet just by redirecting the packet. The simplest strategy to solve this situation is to make the client take a reconnect action. Typical packets flow of the reconnect strategy is illustrated in **Figure 5.5**. First, the client sends a GET (HTTP) packet. When the controller detects it, the controller sets a rule to the switch to rewrite the destination IP address of the packet with the IP address of the proxy. When the proxy receives the GET (HTTP) packet, the proxy returns a RST (TCP) packet because the proxy does not have the connection of it. The OpenFlow switch notifies the receipt of the RST (TCP) packet to the controller, which makes the controller set a rule to rewrite the source IP address of the packet with the IP address of the host. After all, a connection between the client and the proxy is established through the 3-way handshake, so that the proxy can process HTTP packets from the client normally. Note that the reconnect strategy depends on TCP implementations and/or client applications such as web browsers. However, most TCP implementations and client applications take a reconnect action, when they receive a RST (TCP) packet for fault resilient. Therefore, OpenWeb employs this strategy so far. More general discussion about connections between clients and proxy servers is held in Section 5.7.2.

5.4.4 Caching Algorithm

For the redirection methods, a list of target hosts must be prepared as a condition of redirections. In addition, if there are several proxy servers in the OpenWeb system, redirections should be set to suitable proxy servers. However, in this chapter, how to create a list of target hosts and proxy server selection method are not discussed, because there a lot of works related these problems. For example, proxy servers can simply cache content which is requested more than a threshold. Proxy servers can be selected by *fastest response times*, and so on. Though it is also not discussed in this chapter, if an administrator manages several OpenWeb systems, he or she can combine controllers of the systems to apply extensions such as pre-caching of content to another

OpenWeb proxy servers. This kind of caching algorithm could be proposed in the future.

5.5 Performance Evaluation

A prototype system of OpenWeb has been built based on NetFPGA, to evaluate the latency required to redirect packets with NAT and NAPT.

5.5.1 Prototyping

To implement a prototype of OpenWeb, NetFPGA is employed as an OpenFlow switch. In addition, NOX [59] is employed to write a controller program of the OpenFlow platform. Using NOX, control programs can be written in Python. In this prototyping, OpenFlow 0.9.0, NOX 0.6.0, and Python 2.6.4 are used.

The NetFPGA-based OpenFlow switch and its controller are working on the same machine, containing Intel Pentium 4 CPU 3.20 GHz with 1024MB memory, running CentOS 5.4 (Linux 2.6.18).

5.5.2 Evaluation Setting

The evaluation environment is the same as Figure 5.2. The controller and the proxy server are connected with the OpenFlow switch, while the client is connected with its network. Two network devices, a normal layer-2 switch and a Linux machine running *iptables*, are prepared to be compared with. The Linux machine is the same machine used by the OpenWeb system. The normal layer-2 switch just transfers packets as usual and the *iptables* do NAT or NAPT. These two devices replace the OpenFlow switch when each device is evaluated. All components are connected by gigabit Ethernet.

The client tries to get content 100 times per each combination of NAT/NAPT and the network devices. As subject contents of the client, 128KB, 256KB, 512KB, 1024KB, and 2048KB files are prepared. The interval time for trial is 1000ms. Every execution

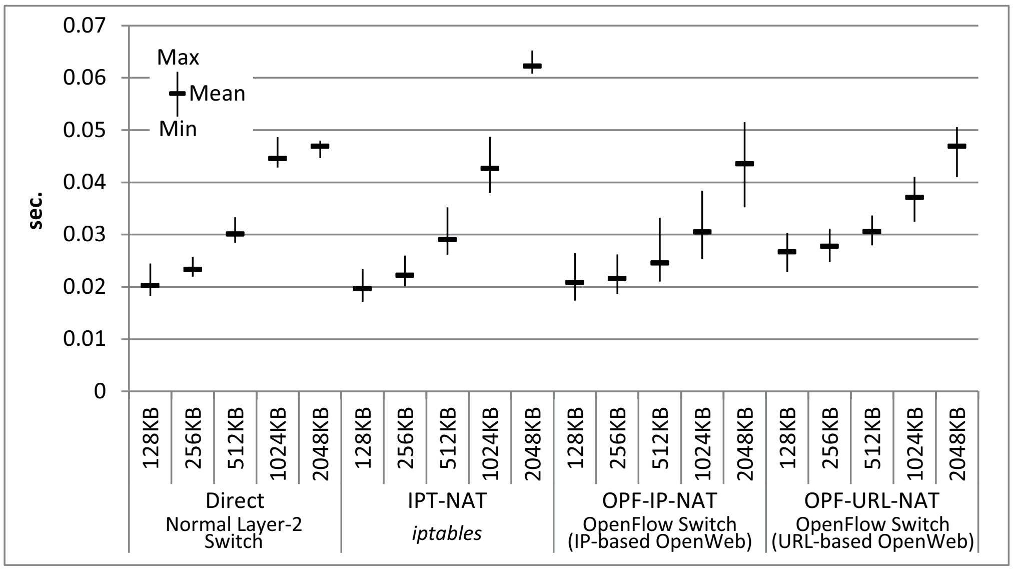


Figure 5.6: Execution Time of NAT

time is recorded, and the minimum, the maximum and the mean are evaluated.

5.5.3 Result

The results comparing OpenWeb with the normal layer-2 switch and the *iptables* are illustrated in **Figure 5.6** and **Figure 5.7**. Figure 5.6 and Figure 5.7 represent the results for NAT and NAPT, respectively. Note that the scales of the vertical axes of the two graphs are different. The "Direct" represents results by the normal layer-2 switch, the "IPT-*-*" represents results by the *iptables*, the "OPF-*-*" represents results by the OpenFlow switch (OpenWeb), the "*-NAT" and the "*-*-NAT" represent results by NAT, the "*-NAPT" and the "*-*-NAPT" represent results by NAPT, the "OPF-IP-*" represents results by IP-based redirection, and the "OPF-URL-*" represents results by URL-based redirection.

Figure 5.6 shows that OpenWeb with NAT (the "OPF-IP-NAT" and the "OPF-URL-NAT") works faster than *iptables* with NAT (the "IPT-NAT") at large. Furthermore, OpenWeb with NAT works faster than the "Direct" in some cases. It would

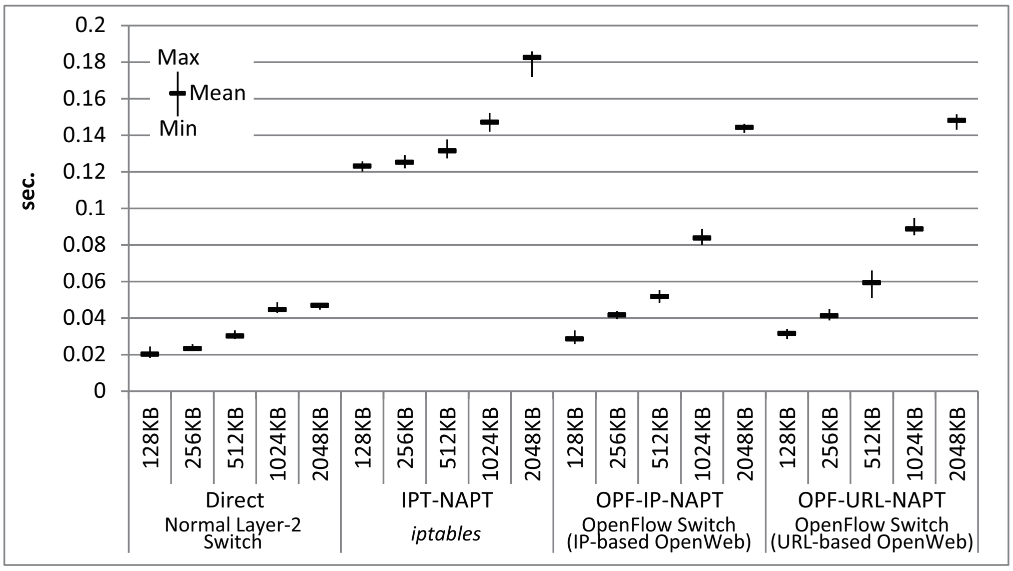


Figure 5.7: Execution Time of NAPT

be due to the high performance of the NetFPGA and/or the poor performance of the layer-2 switch. The "OPF-IP-NAT" is a little faster than the "OPF-URL-NAT."

Figure 5.7 shows that OpenWeb with NAPT (the "OPF-IP-NAPT" and the "OPF-URL-NAPT") works much faster than *iptables* with NAPT (the "IPT-NAPT"). But this time, they work slower than the "Direct", because additional process, rewriting a source and a destination port, is needed. There is a little difference between the "OPF-IP-NAPT"'s performance and the "OPF-URL-NAPT"'s performance in this index.

In these figures, outliers in the results are omitted using the Grubbs' test, which is a statical method for outliers. For instance of an outlier, in the OpenWeb environment, the actual maximum time of OPF-IP-NAT for 2048KB is 0.99218321 seconds, which is about 22 times of its mean. The author could not identify the reason of this, but there is no factor in the OpenWeb environment itself to cause the delay. This problem would be resolved by the improvements of the OpenFlow environment including NetFPGA and Python.

As seen above, although the stabilization is required, OpenWeb will be favorably comparable to a normal layer-2 switch and *iptables*.

5.6 Network Simulation

Network simulations are conducted to investigate how OpenWeb works in a random network. If OpenWeb works effectively even in a random network, OpenWeb will work in the real Internet more effectively because prior knowledge can be used to arrange OpenWeb components.

5.6.1 Simulated Environment: Random Network

It it presupposed that 1024 nodes exist in one random network. A node represents an autonomous system. In this simulation, two types of nodes are prepared: normal nodes and OpenWeb nodes.

- Normal Node

  Normal nodes only forward requests to a server according to routing information.

- OpenWeb Node

  OpenWeb nodes redirect requests to proxy servers as far as its maximum number of incoming connections allows. If the capacity is full, it works as a normal node.

The nodes are connected by 5238 (1/100 of complete graph) edges. An edge represents a connection between nodes. A server and clients are arranged in the network randomly. Note that the server and the clients are not nodes, but they belong to nodes. The number of clients is decided at random for each simulation.

5.6.2 Simulation Scenario

The simulation scenario is as follows.

1. Generate a random network

Table 5.1: Possible Ways of Request Process

Processed by	How processed
Server	*Served* *Waited to be served* *Rejected*
OpenWeb	*Redirected* *Waited to be redirected*

2. Generate a total of 1024 requests from randomly-arranged clients; all the requests try to reach the server which is the sole server in the network
3. The requests head for the server according to routing control which mimics IP routing
4. Requests coming to OpenWeb nodes will be redirected to proxy servers as far as its capacity allows
5. Requests coming to the server will be processed as far as its capacity allows
6. Check how each of the requests is processed

Requests can be processed in five ways as shown in **Table 5.1**. *Served* indicates that requests are served by the server. *Waited to be served* indicates that requests are waited but served by the server. *Rejected* indicates that requests are rejected and not served by the server. *Redirected* indicates that requests are redirected to proxy servers by OpenWeb. *Waited to be redirected* indicates that requests are waited but redirected to proxy servers by OpenWeb.

5.6.3 Parameter Setting

All parameters for the simulations are shown in **Table 5.2**. First, the number of OpenWeb nodes is a parameter. In one random network, 1024 nodes exist as explained before. Accordingly, the number of OpenWeb nodes ranges from 0 to 1024. Once the

Table 5.2: Parameter Value

Parameter		Value
Number of OpenWeb Nodes		0, 1, 2, 3, ..., 1024
Cache Ratio of OpenWeb Nodes		0%, 25%, 50%, 75%, 100%,
Server	MaxClients	256
	ListenBackLog	511
OpenWeb Node	MaxClients	1, 2, 4, 8, 16, 32, 64, 128, 256, 512

number of OpenWeb nodes is determined, a cache ratio of it—how many OpenWeb nodes already have content cache—needs to be determined. The ratio is represented by the percentage of the number of OpenWeb nodes.

Next, there are two parameters for a server and one parameter for OpenWeb nodes. MaxClients represents the maximum number of connections that can be processed concurrently. At the server, requests are served. At the OpenWeb nodes, requests are redirected to proxy servers. ListenBackLog represents the maximum length of the queue of pending connections which means that requests are waited but they will be surely served later. This time, default values of Apache 2.2 are used for these server parameters. A value of MaxClients for OpenWeb node is fixed to 1, 2, 4, 8, 16, 32, 64, 128, 256, and 512. Finally, 1000 trials are done for each parameter combination.

5.6.4 Result

Because it is not realistic to show all of the simulation results, as an example, **Figure 5.8** shows the results in case when the cache ratio parameter and the MaxClients parameter of OpenWeb node are fixed to 50% and 1, respectively. The horizontal axis represents the number of OpenWeb nodes in the random network. The vertical axis represents the number of requests. Each line represents each result of request processes. As the number of OpenWeb nodes increase, *Served*, *Waited to be Served*, and *Rejected* requests decrease naturally, because the total capacity of request redirection increases. In contrast, *Redirected* and *Waited to be redirected* increase of course.

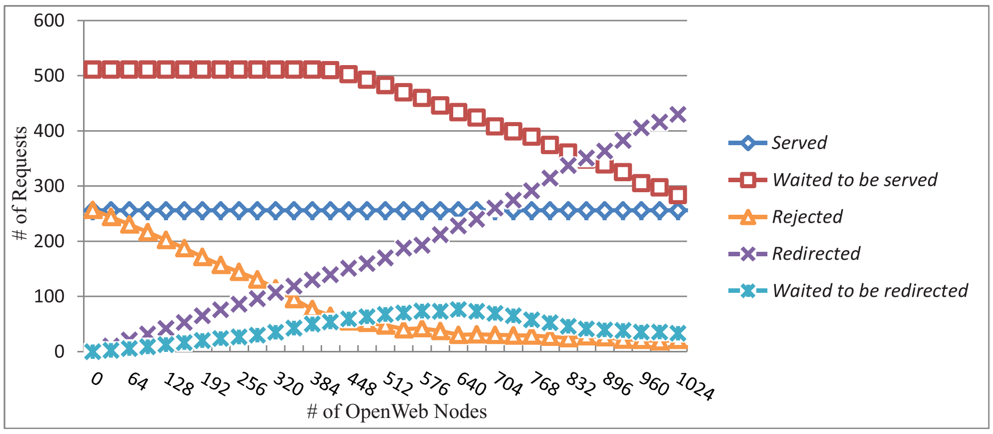


Figure 5.8: All Requests (OpenWeb::Cache Ratio = 50%, OpenWeb::MaxClients = 1)

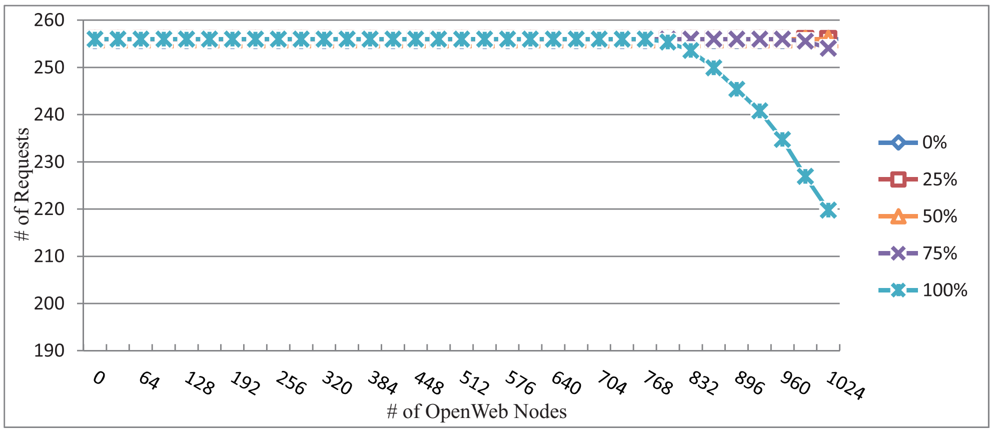


Figure 5.9: *Served* Requests (OpenWeb::MaxClients = 1)

The detailed results in case the MaxClients parameter is fixed to 1 are illustrated in **Figure 5.9** through **5.13**. This time, each line represents each result of cache ratios. Throughout the figures from Figure 5.9 to Figure 5.13, although the difference between cache ratios seems significant, from the viewpoint of content serving to clients, it is not large difference because Figure 5.11 shows that *Rejected* requests are decreasing in the same way in the latter half. This means that if there are sufficient amount of OpenWeb nodes, it is not a major matter whether OpenWeb nodes initially have content cache or not for content serving to clients.

Clearly, the key of load-balancing is to decrease the number of *Rejected* requests

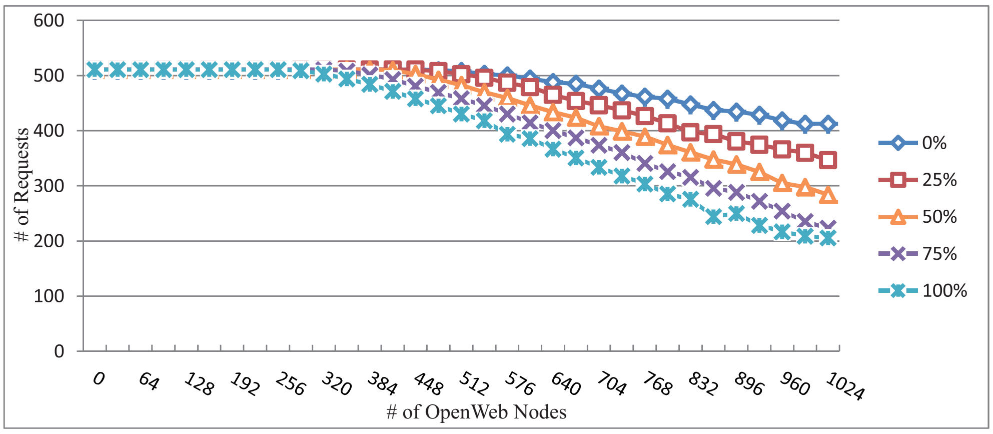


Figure 5.10: *Waited to be served* Requests (OpenWeb::MaxClients = 1)

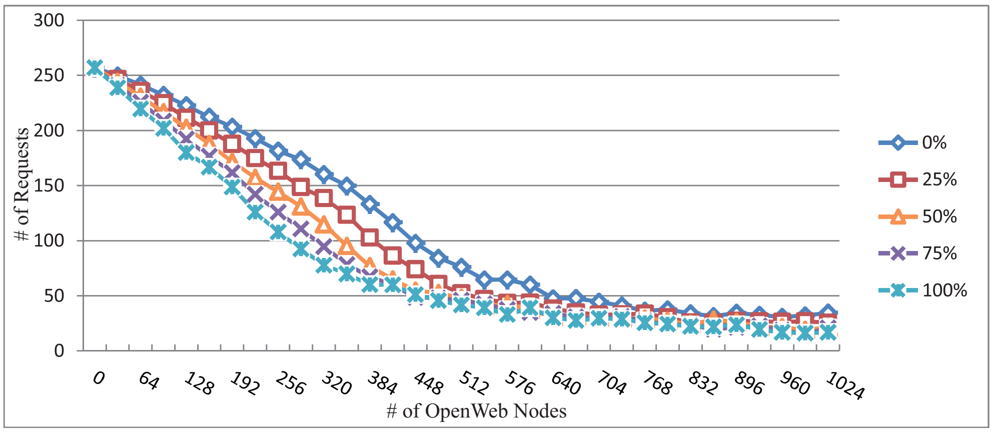


Figure 5.11: *Rejected* Requests (OpenWeb::MaxClients = 1)

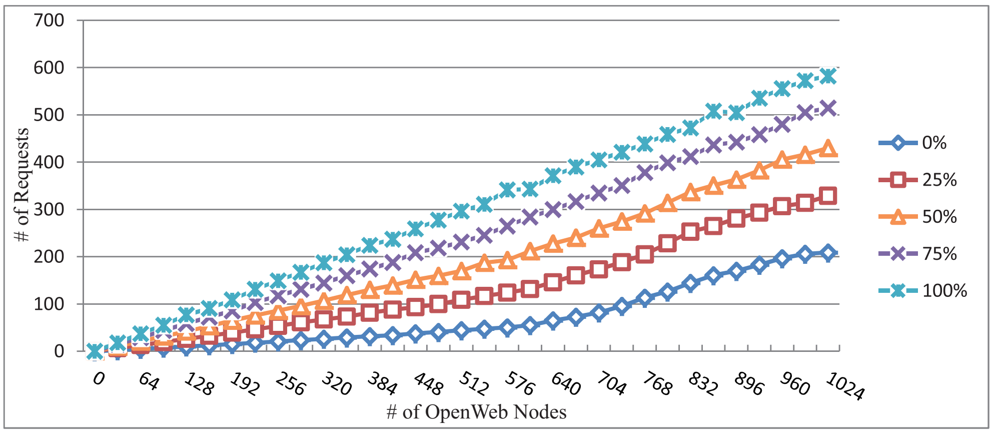


Figure 5.12: *Redirected* Requests (OpenWeb::MaxClients = 1)

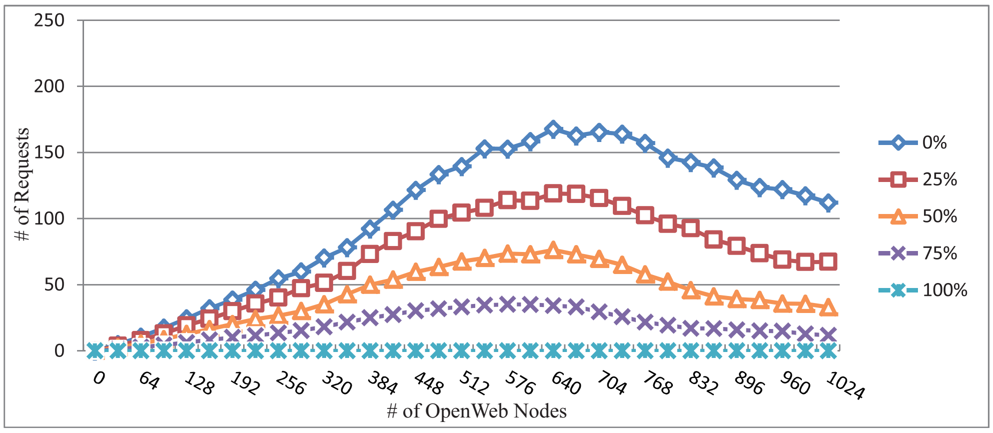


Figure 5.13: *Waited to be redirected* Requests (OpenWeb::MaxClients = 1)

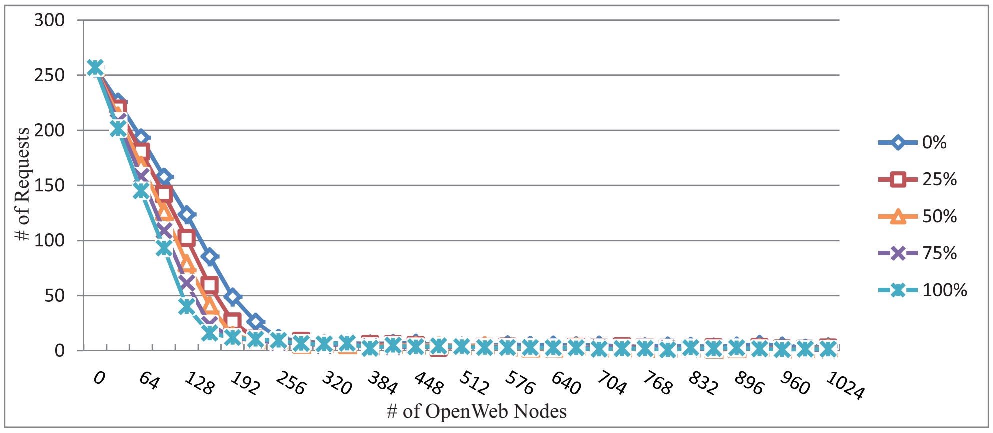


Figure 5.14: *Rejected* Requests (OpenWeb::MaxClients = 8)

because *Rejected* requests are no more than major source of overload. They demand content from a server which is already in full capacity. It is also directly linked with user experience. **Figure 5.14** and **Figure 5.15** indicate *Rejected* results for a MaxClients value of OpenWeb node at 8 and 64, respectively. As shown in these graphs, the number of *Rejected* requests decrease faster than the increase of the number of OpenWeb nodes in any case. The results show that even if OpenWeb systems are arranged in a network randomly, it plays a role as load balancer well.

According to these results, even if a MaxClients value of OpenWeb node is increased, *Rejected* requests cannot be zero if the number of OpenWeb nodes is less than approx-

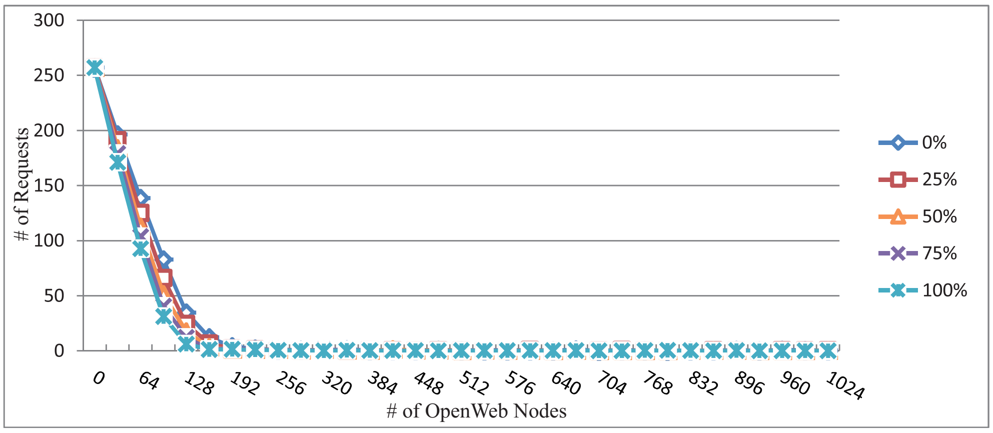


Figure 5.15: *Rejected* Requests (OpenWeb::MaxClients = 64)

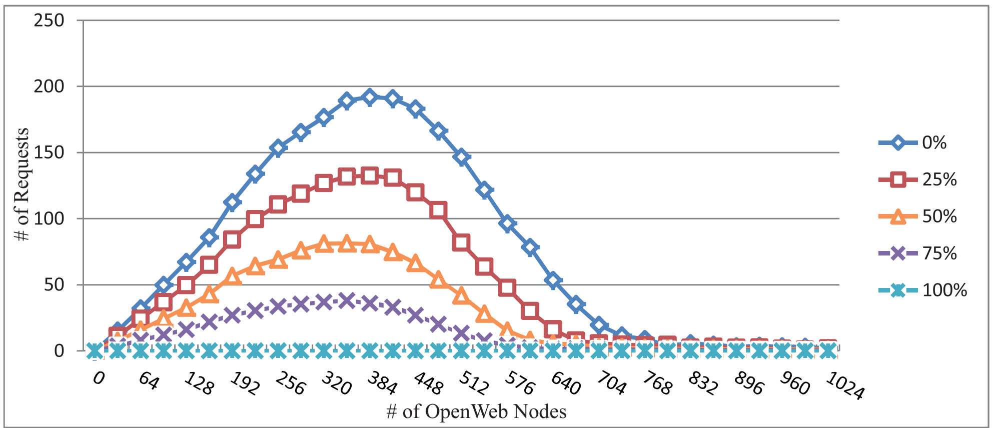


Figure 5.16: *Waited to be redirected* Requests (OpenWeb::MaxClients = 8)

imately 128. This convergence means that about 128 OpenWeb nodes are enough to satisfy all the requests in this simulation settings.

Considering user experience, it is an important factor whether a user request is waited or not waited (or even rejected). According to the results, as a MaxClients value of OpenWeb node increases, a bell curve of *Waited to be redirected* requests gets smaller but it also seems to converge. **Figure 5.16** and **Figure 5.17** indicate *Waited to be redirected* results for a MaxClients value of OpenWeb node at 8 and 64, respectively. This convergence means that at least about 110 clients need to be waited if all OpenWeb nodes do not have content cache.

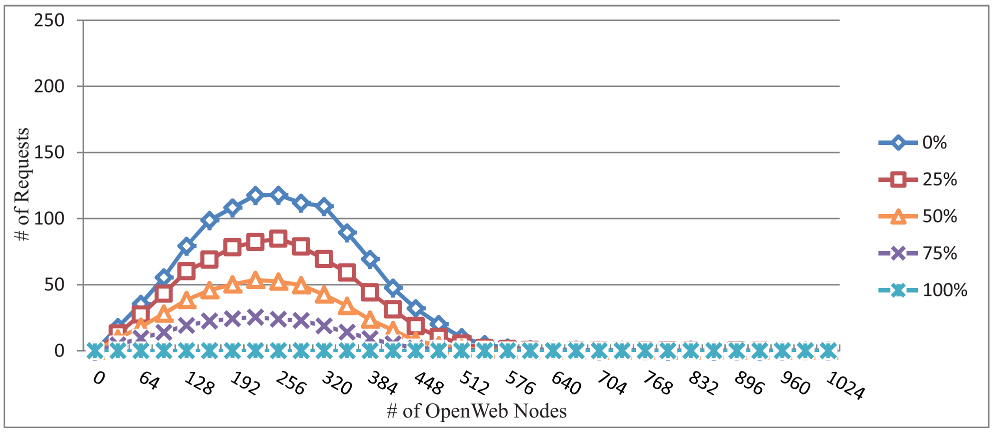


Figure 5.17: *Waited to be redirected* Requests (OpenWeb::MaxClients = 64)

Table 5.3: User Experience Metric

	Served	*Waited to be served*	*Rejected*	*Redirected*	*Waited to be redirected*
S1	1	1	0	1	1
S2	1	0.5	0	1	0.5
S3	1	0.5	0	0.5	0.25

5.7 Discussion

5.7.1 User Experience Metric

To know how much OpenWeb nodes are required for real situation, user experience metrics should be formulated.

Table 5.3 shows three possible metrics. A satisfaction level of clients is indicated by the number, from 0 to 1. Zero indicates that clients are completely dissatisfied and 1 indicates that clients are completely satisfied. S1 is the most loose metric; no matter how long clients are waited and who serves clients with the content, the clients are satisfied if they can get the content. S2 is focused on waiting time; no matter who serves clients with the content, the clients are satisfied if they are not waited. S3 is one of strict metrics; clients mind waiting time and who serves them with the content.

For instance, from the results of Section 5.6, **Figure 5.18** shows the results in case when the cache ratio parameter and the MaxClients parameter of OpenWeb node are fixed to 50% and 1, respectively. **Figure 5.19** shows the results in case when the cache ratio parameter and the MaxClients parameter of OpenWeb node are fixed to 50% and 8, respectively. **Figure 5.20** shows the results in case when the cache ratio parameter and the MaxClients parameter of OpenWeb node are fixed to 50% and 64, respectively. The horizontal axis represents the number of OpenWeb nodes in the random network. The vertical axis represents the total score of user experience.

Throughout the figures from Figure 5.18 to Figure 5.20, it would be fair to say that S1 and S2 can be used as a metric to determine the number of OpenWeb node in a network, because they seem to converge in any case. On the other hand, S3 would be hard to use as a metric, because its curve seems to transform depending on the MaxClients parameter of OpenWeb node.

For example, in case the cache ratio parameter and the MaxClients parameter of OpenWeb node are fixed to 50% and 64, using S1 metric, it can be said that about 128 OpenWeb nodes are enough to satisfy all clients demands.

5.7.2 Connections Between Clients and Proxy Servers

Of course, clients get content from proxy servers through TCP connections. Therefore, it is a key perspective how connections are established between clients and proxy servers in redirection-based cache systems like OpenWeb.

If a cache system works at layer-2 to layer-4, TCP connections are established between clients and proxy servers directly. In this case, the number of TCP connections used in a session is one as clients connect to original hosts.

In contrast, if a cache system works at layer-5 to layer-7, TCP connections between clients and the cache system are established first because information of layer-5 to layer-7 is not available before TCP connections are established. After the information is obtained, TCP connections between the cache system and proxy servers are established.

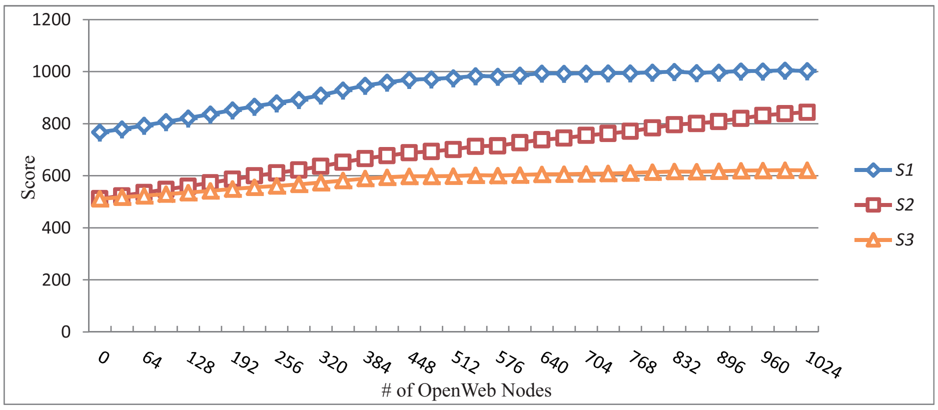


Figure 5.18: User Experience (OpenWeb::Cache Ratio = 50%, OpenWeb::MaxClients = 1)

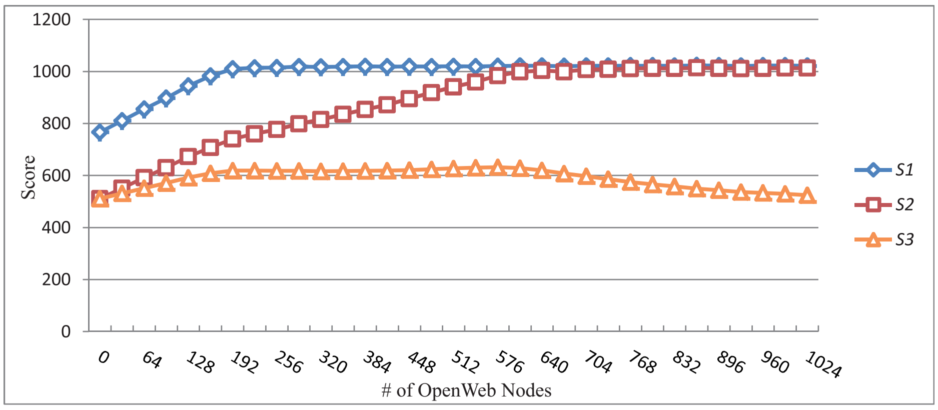


Figure 5.19: User Experience (OpenWeb::Cache Ratio = 50%, OpenWeb::MaxClients = 8)

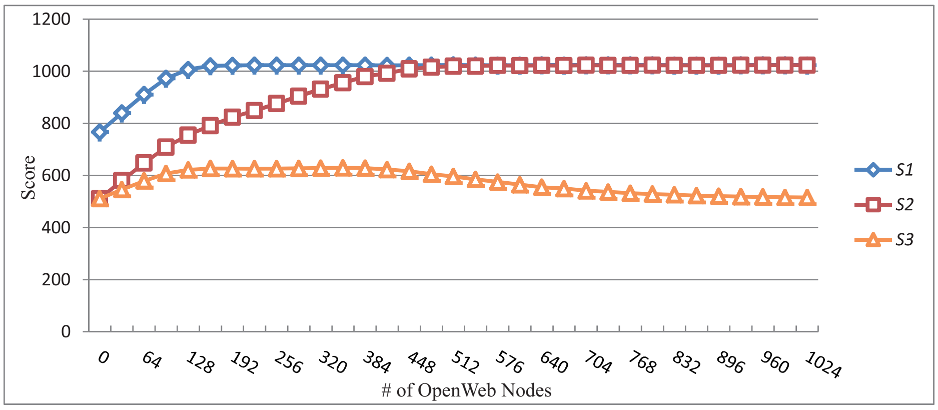


Figure 5.20: User Experience (OpenWeb::Cache Ratio = 50%, OpenWeb::MaxClients = 64)

In this case, the number of TCP connections used in a session is at least two.

Obviously, the former can achieve better performance than the latter, because the number of TCP connections is small. TCP connections supply sequence control, congestion control, and so on, which take some processing time. Alternatively the latter can control the system behavior more flexibly than the former, because information of layer-5 to layer-7 can be used.

Though OpenWeb can use information of all layers (layer-2 to layer-7), OpenWeb works at layer-2. This means that the number of TCP connections is basically one. However, to get information of layer-5 to layer-7, TCP connections should be established. OpenWeb employs the reconnect strategy to get the information as explained in Section 5.4.3.

Although this strategy will work for the time being, there is a little problem of transparency. Technically, clients can find some traces of redirections but they cannot believe that their requests are redirected, because the traces are the same as normal error recovery protocols of TCP connections. In order to hide the redirection from clients completely, at least two TCP connections are required like the latter case. Of course, it is easy to establish two TCP connections. However, performance would be more important than transparency in this case. Alternatively, though proxy servers get to be difficult to be maintained, it is also possible to modify a TCP implementation and a proxy application to enable the proxy servers to process HTTP packets even if there are no connections before the HTTP packets arrive at the proxy servers.

5.7.3 Verification of Research Goal

Before summary, the aim of this research is revisited. The aim of the research is to enable users to use the cache system without any configurations and administrators to manage the cache system, easily and robustly. Achieving this aim, three things have been kept in mind.

The first was independency. The system should be independent from web servers and handle unexpected requests. The network simulations which have been done in random network have confirmed these requirements.

The second was transparency. Clients can use the system without any troublesome configurations. The effect does not depend on client behavior. Of course, these are satisfied because OpenWeb redirects requests at layer-2 using the OpenFlow platform.

The last was performance. The basic performance evaluation shows that performance of OpenWeb achieves some positive results.

As seen above, it can be said that the goal has been almost accomplished. In addition, using prior knowledge to arrange OpenWeb components, and applying advanced caching algorithm, OpenWeb can improve its functionality and performance toward the future.

5.8 Summary

This chapter presented the design of OpenWeb, a transparent overlay connection proxy resident on an OpenFlow layer-2 switch. OpenWeb requires no configurations of clients to use the system. While only the context of an HTTP connection proxy is described in this chapter, the functionality is fully generalizable to any IP-based overlay proxy service, including the services accessed by predecessor services such as Oasis and Ocala. Further, the programmability and fine-grained nature of the OpenFlow programming and switching environment offers possibilities which are not presented for earlier writes in this field; in particular, different proxy and overlay services can be accessed on a per-application, per-content, or per-user basis, among others. As a future work, it is should be considered to explore those possibilities, and other rich application areas.

Chapter 6

Conclusion

In this thesis, the three approaches have been proposed that can be employed in content distribution environments. In the first method, places where content is exchanged are organized into hierarchies so that the extent of content distribution can be controlled. The second method makes content distribution and reputation aggregation work efficiently in a P2P manner to reduce the load of central servers. As the third method, a layer-2 redirect method is implemented on top of the OpenFlow programmable switch platform to deliver content to users requested seamlessly.

Chapter 3 has proposed a method to create user clusters which enables the control of the geographical extent of content distribution. The proposed clustering method organizes places into hierarchies. The extent of content distribution is controlled by its parameters to swap user records among nodes organizing clusters. Not all information should be distributed to people all over the world. Each content would have the proper extent for its distribution. The proposed method can provide a means to control it on a user cluster basis.

Chapter 4 has proposed a method to distribute content and aggregate reputation without central servers. The proposed method for content distribution and reputation aggregation works in a P2P manner. The simulation results has provided insights about trade-offs between consumed network resource and required time for content distribution and reputation aggregation. Enabling content distribution and reputation aggregation in a P2P manner is not only meaningful because of cost problems, but also for protecting the right to freedom of expression.

Chapter 5 has proposed a method to redirect content requests at layer-2. The proposed layer-2 redirect method is implemented on top of the OpenFlow programmable switch platform. It shows a better performance compared with a conventional redirect system working at layer-3. Simulation results made in random networks also show that the proposed system can operate collaboratively on user demands, which reduces the load of central servers and improves the user experience. Software-defined networking (SDN) will be more common in the future Internet. The proposed method can offer a new direction in the cache system for SDN.

In the future, information exchange among people will be more universal and seamless. The proposed three methods have resolved respective problems at a certain level. Using the three methods, a new user-behavior driven infrastructure can be constructed for content distribution environments. It enables the control of the geographical extent of content distribution, simultaneous content distribution and reputation aggregation, and seamless content delivery reducing the utilization of central servers or clouds. In other words, the distribution of content can be controlled geographically with the assurance of the quality of information under an efficient network environment. The author believes that the three major contributions described in this thesis can play a significant role in future content distribution environments to enrich the quality of human life.

Acknowledgment

At the end of my thesis, I would first like to express my deep appreciation and sincere gratitude to Prof. Hideyuki Takada at Ritsumeikan University for supervising and guiding my thesis researches over the years. It would not have been possible to write this thesis without the kind support and help of him.

Next, I must express my special thanks to the cooperation of Dr. Rick McGeer at Hewlett-Packard Laboratories Palo Alto. He provided a comfortable developing environment and insightful discussions about a research when I was in the HP Labs Palo Alto during the summer of 2009.

I would also like to thank the professors, especially Prof. Hiromitsu Shimakawa and Prof. Nobuhiko Nishio, and the past/current members of my laboratory at Ritsumeikan University who have been instrumental in helping me throughout my researches.

Finally, I wish to express my warmest thanks to my wonderful family and friends for always being by my side, supporting and encouraging me to be on track.

References

[1] Cisco Systems. Entering the zettabyte era. White paper, Cisco Systems, June 2011.

[2] Takeshi Sakaki, Fujio Toriumi, and Yutaka Matsuo. Tweet trend analysis in an emergency situation. In *Proceedings of the Special Workshop on Internet and Disasters*, SWID '11, pp. 3:1–3:8, New York, NY, USA, 2011. ACM.

[3] M. Parameswaran, A. Susarla, and A.B. Whinston. P2P networking: An information sharing alternative. *Computer*, Vol. 34, No. 7, pp. 31 –38, Jul 2001.

[4] Alberto Montresor and Luca Abeni. Cloudy weather for P2P, with a chance of gossip. In *Peer-to-Peer Computing (P2P), 2011 IEEE International Conference on*, pp. 250–259. IEEE, 2011.

[5] B. Swanson. The Coming Exaflood. *Wall Street Journal*, Vol. 20, p. A11, 2007.

[6] T. Berners-Lee, R. Cailliau, J.F. Groff, and B. Pollermann. World-wide web: The information universe. *Internet Research*, Vol. 20, No. 4, pp. 461–471, 2010.

[7] New IT Infrastructure for the Information-explosion Era, MEXT Grant-in-Aid for Scientific Research on Priority Areas (online). available from <http://www.infoplosion.nii.ac.jp/info-plosion/ctr.php/m/IndexEng/a/Index/> (accessed 2011-05-19).

[8] R. Nock and F. Nielsen. On weighting clustering. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, Vol. 28, No. 8, pp. 1223–1235, 2006.

[9] M.K. Bergman. The deep web: surfacing hidden value. *Journal of Electronic Publishing*, Vol. 7, No. 1, 2001.

[10] Ritu Khare, Yuan An, and Il-Yeol Song. Understanding deep web search interfaces: a survey. *ACM SIGMOD Record*, Vol. 39, No. 1, pp. 33–40, 2010.

[11] Paul Bristow. The digital divide: Is it an age old question? In *ITI 7th International Conference on Communications and Information Technology (ICICT2009)*, pp. 61–75, Cairo, December 2009. IEEE.

[12] Yasuto Nakanishi, Kazunari Takahashi, Takayuki Tsuji, and Katsuya Hakozaki. iCAMS: A mobile communication tool using location and schedule information. *IEEE Pervasive Computing*, Vol. 3, No. 1, pp. 82–88, 2004.

[13] James Nord, Kåre Synnes, and Peter Parnes. An architecture for location aware applications. In *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*, pp. 3805–3810. IEEE, 2002.

[14] Juha Kolari, Timo Laakko, Tapio Hiltunen, Veikko Ikonen, Minna Kulju, Raisa Suihkonen, Santtu Toivonen, and Tytti Virtanen. Context-aware services for mobile users. *VTT PUBLICATIONS*, 2004.

[15] goo lab (online). available from <http://lifelog.machi.goo.ne.jp/> (accessed 2009-03-19) (in Japanese).

[16] PiTaPa goopas (online). available from <http://http://www.surutto.com/about/release/p040727.pdf> (accessed 2009-03-19) (in Japanese).

[17] Center for spatial information science at the University of Tokyo (online). available from <http://newspat.csis.u-tokyo.ac.jp/geocode/> (accessed 2009-03-19) (in Japanese).

[18] Ace research institute ltd., editor. *A comprehensive list of passengers.* Ace Research Institute Ltd., 1996 (in Japanese).

[19] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '01, pp. 149–160, New York, NY, USA, 2001. ACM.

[20] Shaomei Wu, Jake M. Hofman, Winter A. Mason, and Duncan J. Watts. Who says what to whom on Twitter. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pp. 705–714, New York, NY, USA, 2011. ACM, ACM.

[21] Paolo Costa and Davide Frey. Publish-subscribe tree maintenance over a DHT. In *Distributed Computing Systems Workshops, 2005. 25th IEEE International Conference on*, pp. 414–420. IEEE, 2005.

[22] Yingwu Zhu and Yiming Hu. Ferry: A P2P-based architecture for content-based publish/subscribe services. *Parallel and Distributed Systems, IEEE Transactions on*, Vol. 18, No. 5, pp. 672–685, 2007.

[23] Weixiong Rao, R. Vitenberg, and S. Tarkoma. Towards optimal keyword-based content dissemination in DHT-based P2P networks. In *2011 IEEE International Conference on Peer-to-Peer Computing*, pp. 102–111. IEEE, September 2011.

[24] G. Mega, A. Montresor, and G.P. Picco. Efficient dissemination in decentralized social networks. In *2011 IEEE International Conference on Peer-to-Peer Computing*, pp. 338–347. IEEE, September 2011.

[25] A-M Kermarrec, Alessio Pace, Vivien Quéma, and Valerio Schiavoni. Nat-resilient gossip peer sampling. In *Distributed Computing Systems, 2009. ICDCS'09. 29th IEEE International Conference on*, pp. 360–367. IEEE, 2009.

[26] Spyros Voulgaris, Daniela Gavidia, and Maarten Van Steen. Cyclon: Inexpensive membership management for unstructured P2P overlays. *Journal of Network and Systems Management*, Vol. 13, No. 2, pp. 197–217, 2005.

[27] Nazanin Magharei, Reza Rejaie, and Yang Guo. Mesh or multiple-tree: A comparative study of live P2P streaming approaches. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pp. 1424–1432. IEEE, 2007.

[28] F. Wang, J. Liu, and Y. Xiong. Stable peers: Existence, importance, and application in peer-to-peer live video streaming. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pp. 1364–1372. IEEE, 2008.

[29] Runfang Zhou, Kai Hwang, and Min Cai. Gossiptrust for fast reputation aggregation in peer-to-peer networks. *Knowledge and Data Engineering, IEEE Transactions on*, Vol. 20, No. 9, pp. 1282–1295, 2008.

[30] Takashi Yajima, Asaki Matsumoto, and Hiroshi Shigeno. Ptrust: Provisional value based trust for reputation aggregation in peer-to-peer networks. In *Access Spaces (ISAS), 2011 1st International Symposium on*, pp. 180–185. IEEE, 2011.

[31] Kota Abe, Toshiyuki Abe, Tatsuya Ueda, Hayato Ishibashi, and Toshio Matsuura. Aggregation skip graph: A skip graph extension for efficient aggregation query

over p2p networks. *International Journal On Advances in Internet Technology*, Vol. 4, No. 3 and 4, pp. 103–110, 2012.

[32] James Aspnes and Gauri Shah. Skip graphs. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '03, pp. 384–393, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.

[33] Agoop Corp. Data analysis for smartphone packet communication (online). available from <http://www.agoop.co.jp/solutions/reports/report_3/report.html> (accessed 2013-03-24) (in Japanese).

[34] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is Twitter, a social network or a news media? In *WWW '10: Proceedings of the 19th international conference on World wide web*, pp. 591–600, New York, NY, USA, 2010. ACM.

[35] N. Radio, Ying Zhang, M. Tatipamula, and V.K. Madisetti. Next-generation applications on cellular networks: Trends, challenges, and solutions. *Proceedings of the IEEE*, Vol. 100, No. 4, pp. 841–854, 2012.

[36] Browserscope. A community-driven project for profiling web browsers (online). available from <http://www.browserscope.org/> (accessed 2013-07-10).

[37] E. Nygren, R.K. Sitaraman, and J. Sun. The akamai network: A platform for high-performance Internet applications. *ACM SIGOPS Operating Systems Review*, Vol. 44, No. 3, pp. 2–19, 2010.

[38] M. El Dick, E. Pacitti, and B. Kemme. A highly robust P2P-CDN under large-scale and dynamic participation. In *Advances in P2P Systems, 2009. AP2PS'09. First International Conference on*, pp. 180–185. IEEE, 2009.

[39] Antony I. T. Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, Middleware '01, pp. 329–350, London, UK, UK, 2001. Springer-Verlag.

[40] Petar Maymounkov and David Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In *Peer-to-Peer Systems*, pp. 53–65. Springer, 2002.

[41] Thorsten Schütt, Florian Schintke, and Alexander Reinefeld. Range queries on structured overlay networks. *Comput. Commun.*, Vol. 31, No. 2, pp. 280–291, February 2008.

[42] L. Garces-Erice, E. W. Biersack, P. A. Felber, K. W. Ross, and G. Urvoy-Keller. Hierarchical peer-to-peer systems. *Euro-Par 2003 Parallel Processing*, pp. 1230–1239, 2003.

[43] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: enabling innovation in campus networks. *ACM Computer Communication Review*, Vol. 38, No. 2, pp. 69–74, 2008.

[44] Vivek S. Pai, Limin Wang, KyoungSoo Park, Ruoming Pang, and Larry Peterson. The Dark Side of the Web: An Open Proxy's View. In *HotNets-II*, 2002.

[45] Limin Wang, KyoungSoo Park, Ruoming Pang, Vivek S. Pai, and Larry Peterson. Reliability and Security in the CoDeeN Content Distribution Network. In *Proceedings of the USENIX 2004 Annual Technical Conference*, 2004.

[46] KyoungSoo Park and Vivek S. Pai. Scale and Performance in the CoBlitz Large-File Distribution Service. In *Proceedings of the 3rd Symposium on Networked Systems Design and Implementation (NSDI '06)*, 2006.

[47] Michael J. Freedman, Eric Freudenthal, and David Mazieres. Democratizing Content Publication with Coral. In *Proc. 1st USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI '04)*, 2004.

[48] Arnaud Legout, Nikitas Liogkas, Eddie Kohler, and Lixia Zhang. Clustering and sharing incentives in bittorrent systems. In *Proceedings of the 2007 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, SIGMETRICS '07, pp. 301–312, New York, NY, USA, 2007. ACM.

[49] Ryan S. Peterson and Emin Gun Sirer. Antfarm: efficient content distribution with managed swarms. In *Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, 2009.

[50] S. Iyer, A. Rowstron, and P. Druschel. Squirrel: A decentralized peer-to-peer web cache. In *Proceedings of the twenty-first annual symposium on Principles of distributed computing*, pp. 213–222. ACM, 2002.

[51] Mayur R. Palankar, Adriana Iamnitchi, Matei Ripeanu, and Simson Garfinkel. Amazon s3 for science grids: a viable solution? In *Proceedings of the 2008 international workshop on Data-aware distributed computing*, DADC '08, pp. 55–64, New York, NY, USA, 2008. ACM.

[52] Sean Rhea, Patrick Eaton, Dennis Geels, Hakim Weatherspoon, Ben Zhao, and John Kubiatowicz. Pond: the OceanStore Prototype. In *Proceedings of the 2nd USENIX Conference on File and Storage Technologies (FAST '03)*, 2003.

[53] M. Beck, Y. Ding, T. Moore, and J. Plank. Transnet Architecture and LogisticalNetworking for Distributed Storage. In *Workshop on Scalable File System and StorageTechnologies*, 2004.

[54] Dilip Joseph, Jayanthkumar Kannan, Ayumu Kubota, Karthik Lakshminarayanan, Ion Stoica, and Klaus Wehrle. OCALA: An Architecture for Supporting Legacy Applications over Overlays. In *3rd USENIX/ACM Symposium on Networked Systems Design and Implementation*, May 2006.

[55] Harsha V. Madhyastha, Arun Venkataramani, Arvind Krishnamurthy, and Thomas Anderson. Oasis: An Overlay Aware Network Stack. *SIGOPS Operating Systems Review*, Vol. 40, No. 1, pp. 41–48, 2006.

[56] IETF: Web Cache Communication Protocol V2.0 (online). available from <http://tools.ietf.org/id/draft-wilson-wrec-wccp-v2-01.txt> (accessed 2011-05-19).

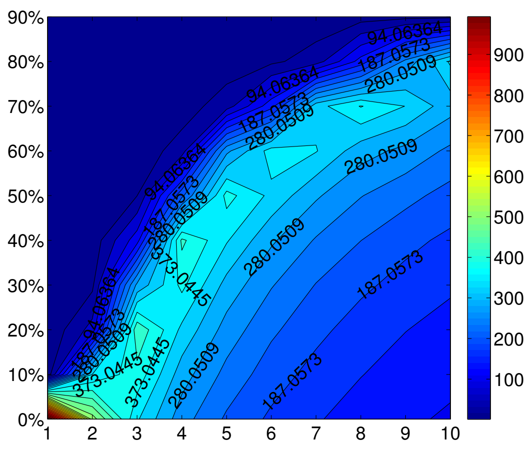
[57] J.W. Lockwood, N. McKeown, G. Watson, G. Gibb, P. Hartke, J. Naous, R. Raghuraman, and J. Luo. NetFPGA–An Open Platform for Gigabit-Rate Network Switching and Routing. In *Microelectronic Systems Education, 2007. MSE'07. IEEE International Conference on*, pp. 160–161. IEEE, 2007.

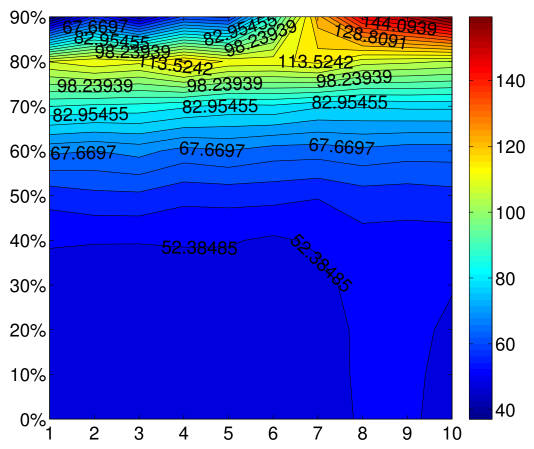
[58] Martin Casado, Michael J. Freedman, Justin Pettit, Jianying Luo, Nick McKeown, and Scott Shenker. Ethane: Taking Control of the Enterprise. In *Proceedings of SIGCOMM*, 2007.

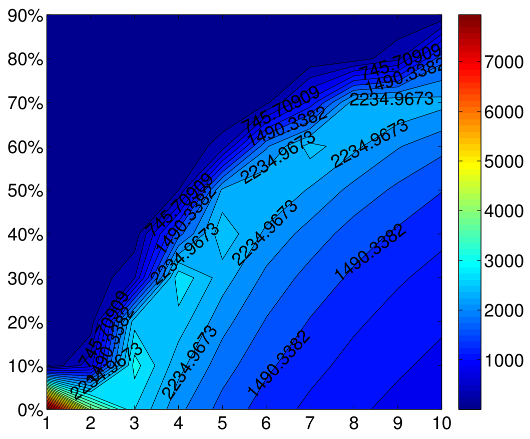
[59] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker. NOX: towards an operating system for networks. *ACM SIGCOMM Computer Communication Review*, Vol. 38, No. 3, pp. 105–110, 2008.

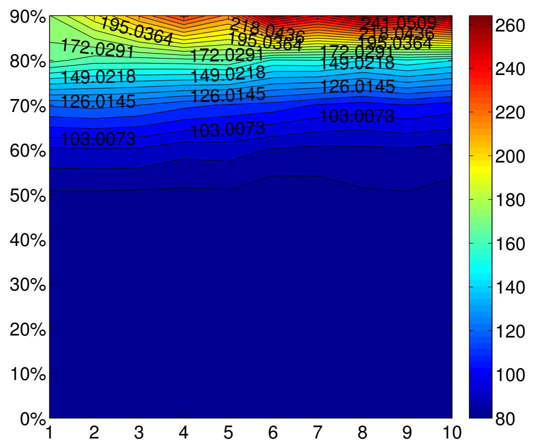
Appendix

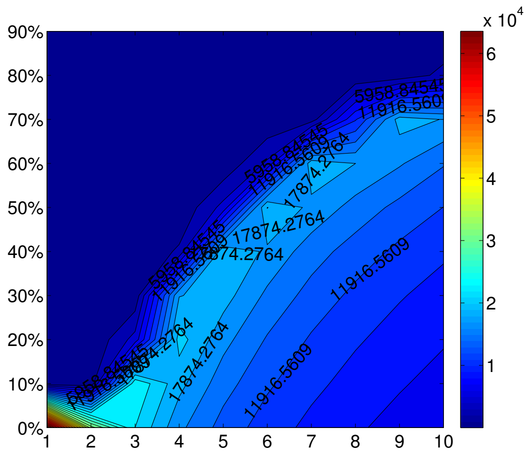
All of the results of the simulation in Section 4.4 are shown in **Figure A.1** through **Figure A.6**. The total efficiency for each result is also shown in **Figure A.7**.


(a) 2^{10} Peers (succ only)


(b) 2^{10} Peers (succ w/ fing)


(c) 2^{13} Peers (succ only)


(d) 2^{13} Peers (succ w/ fing)


(e) 2^{16} Peers (succ only)

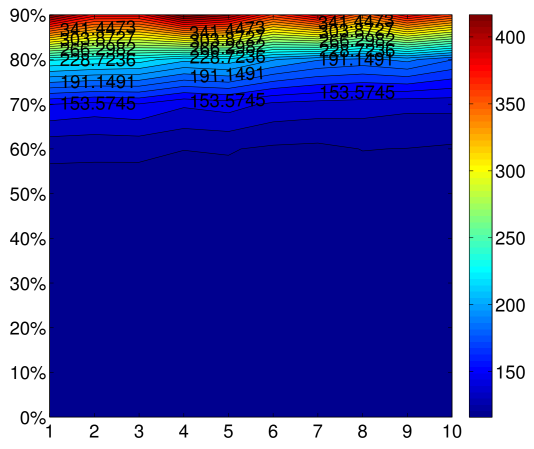
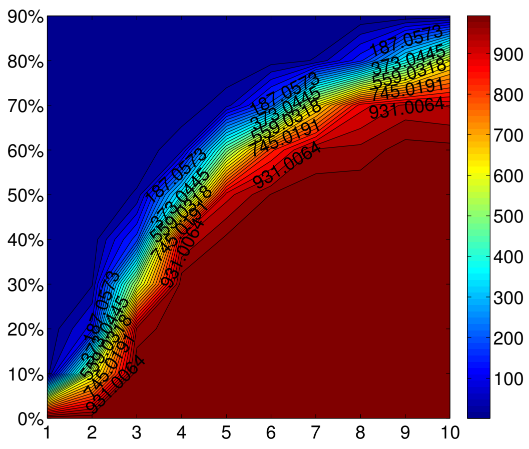
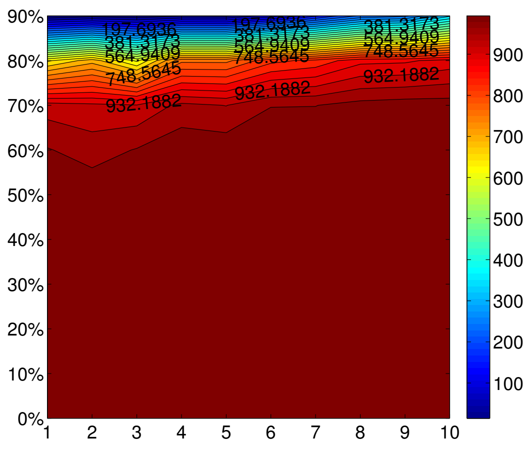
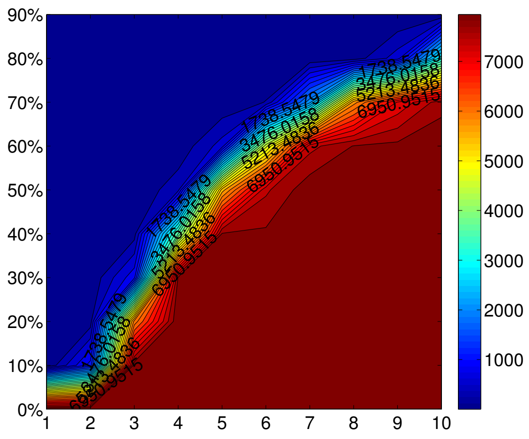
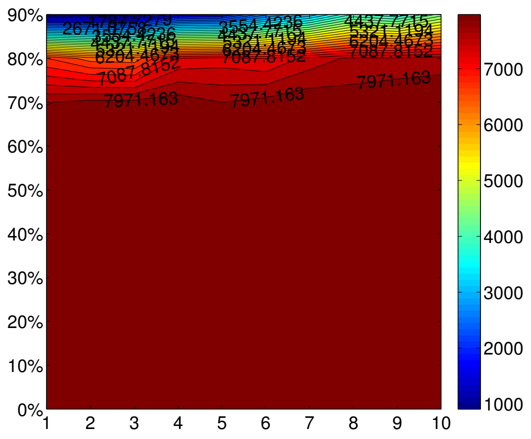

(f) 2^{16} Peers (succ w/ fing)

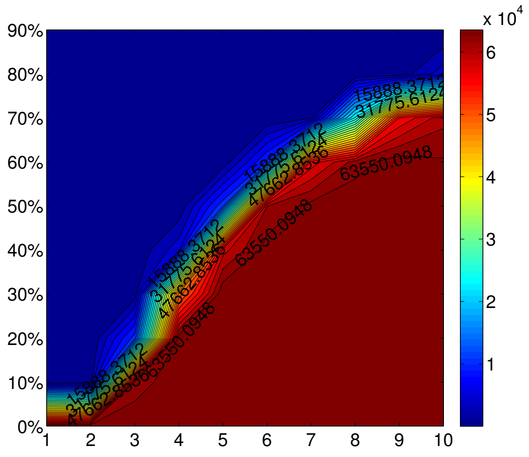
Figure A.1: Execution Time


(a) 2^{10} Peers (succ only)


(b) 2^{10} Peers (succ w/ fing)


(c) 2^{13} Peers (succ only)


(d) 2^{13} Peers (succ w/ fing)


(e) 2^{16} Peers (succ only)

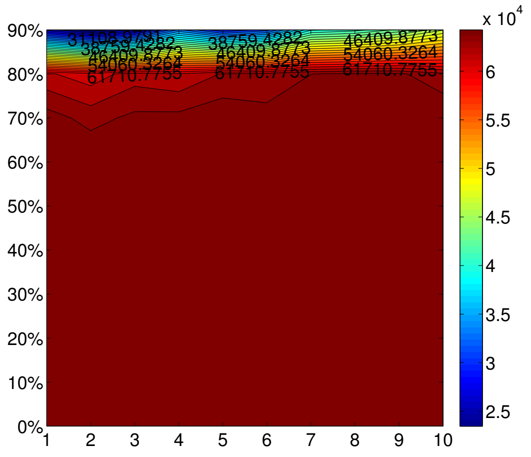
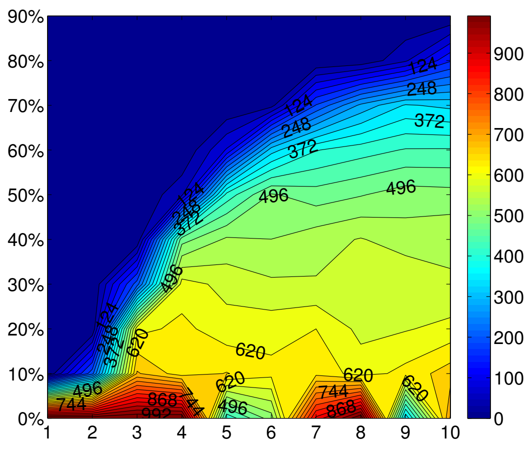
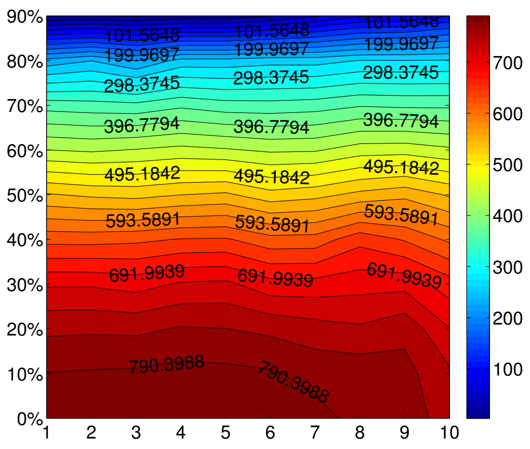

(f) 2^{16} Peers (succ w/ fing)

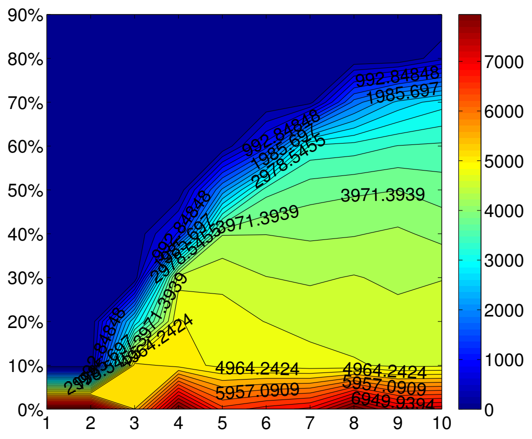
Figure A.2: Informed Peer



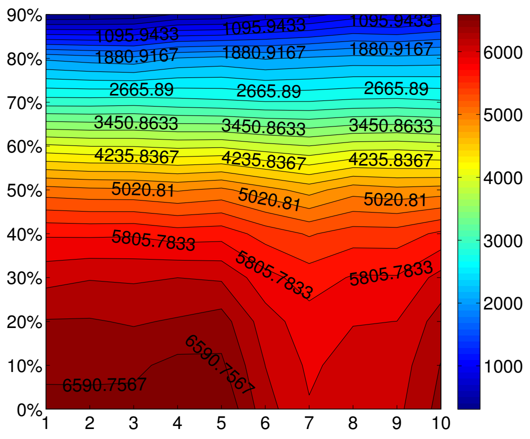
(a) 2^{10} Peers (succ only)



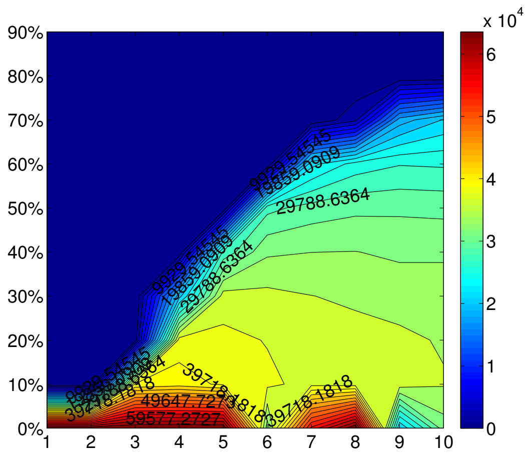
(b) 2^{10} Peers (succ w/ fing)



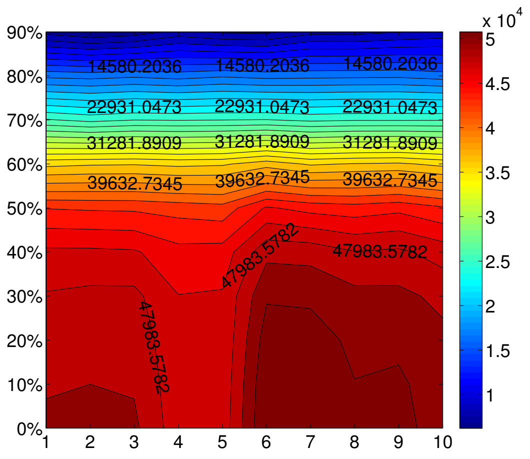
(c) 2^{13} Peers (succ only)



(d) 2^{13} Peers (succ w/ fing)

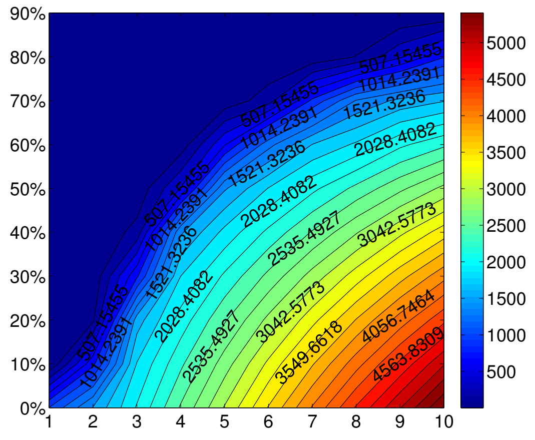


(e) 2^{16} Peers (succ only)

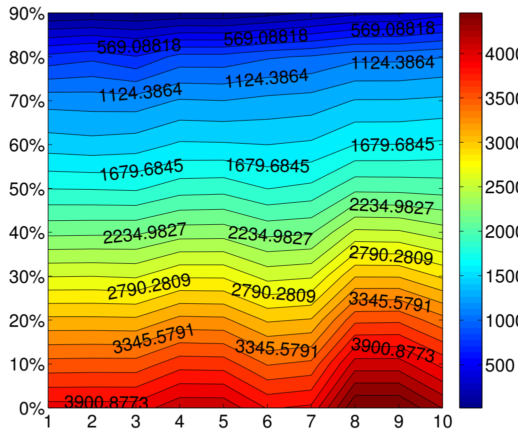


(f) 2^{16} Peers (succ w/ fing)

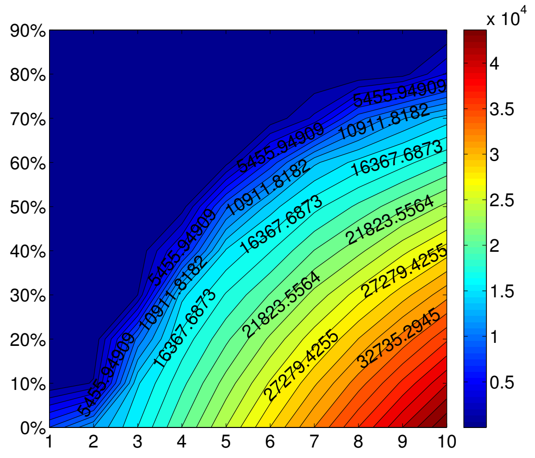
Figure A.3: Tallied Reputation



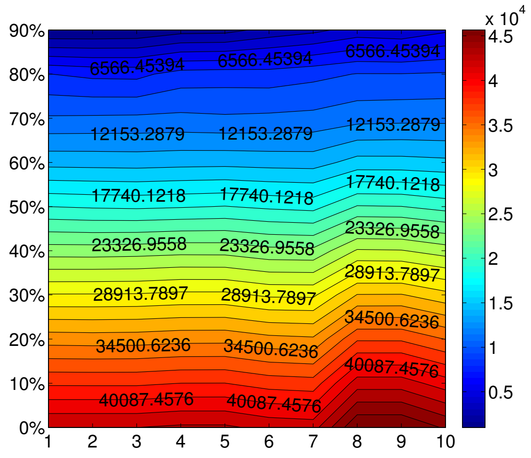
(a) 2^{10} Peers (succ only)



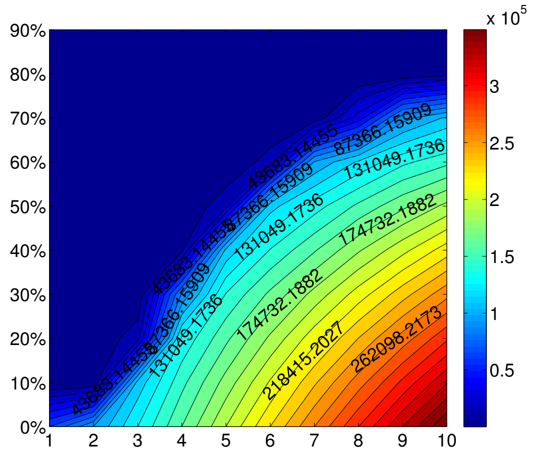
(b) 2^{10} Peers (succ w/ fing)



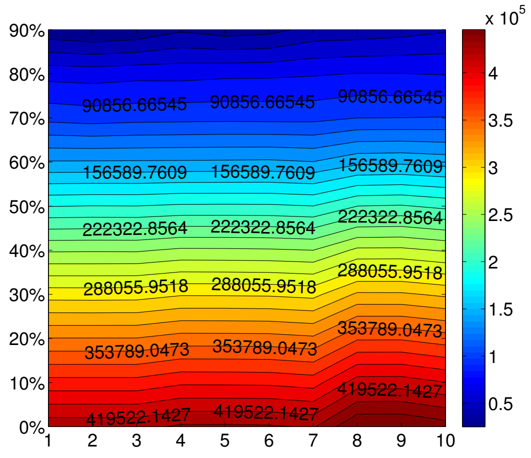
(c) 2^{13} Peers (succ only)



(d) 2^{13} Peers (succ w/ fing)

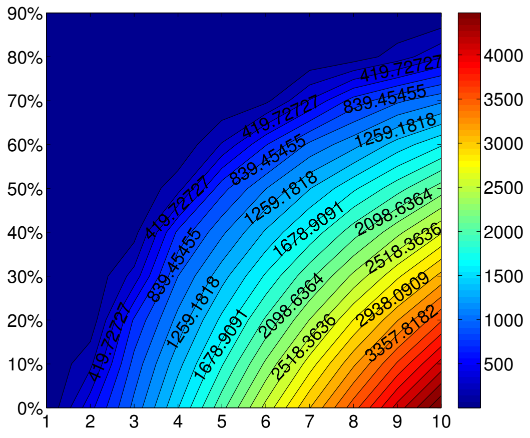


(e) 2^{16} Peers (succ only)

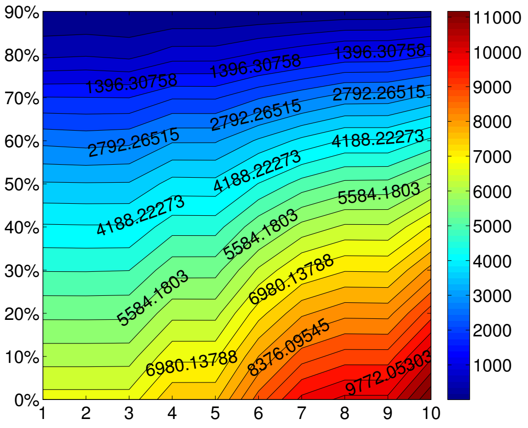


(f) 2^{16} Peers (succ w/ fing)

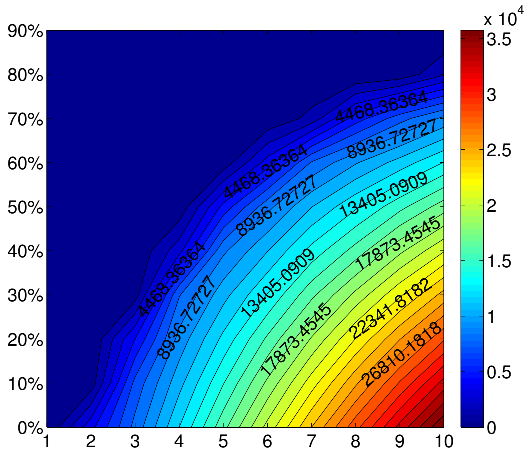
Figure A.4: Effective Transmission



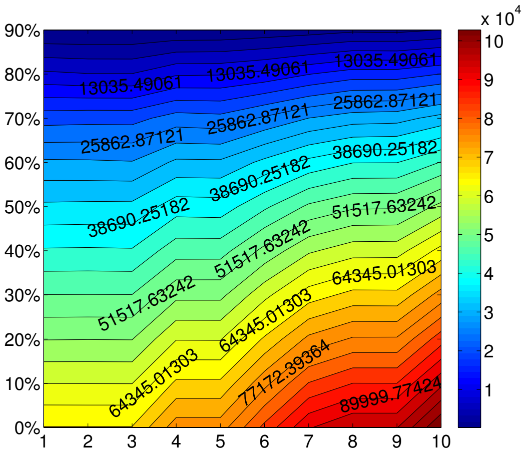
(a) 2^{10} Peers (succ only)



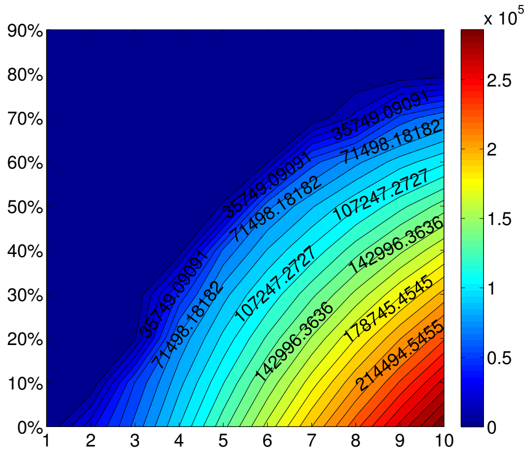
(b) 2^{10} Peers (succ w/ fing)



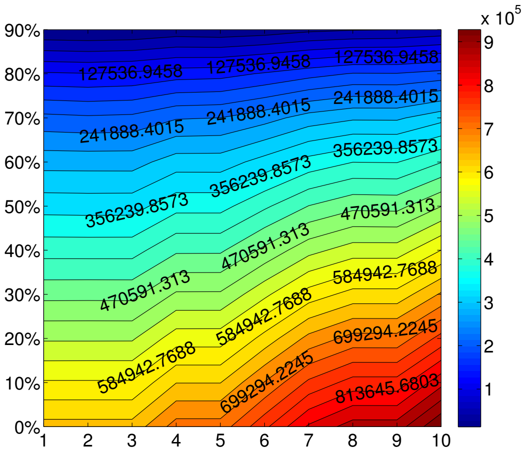
(c) 2^{13} Peers (succ only)



(d) 2^{13} Peers (succ w/ fing)

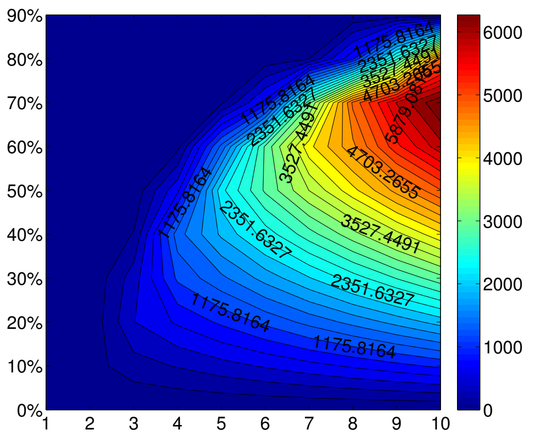


(e) 2^{16} Peers (succ only)

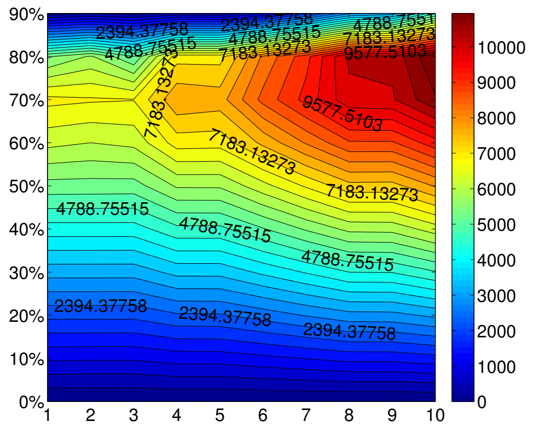


(f) 2^{16} Peers (succ w/ fing)

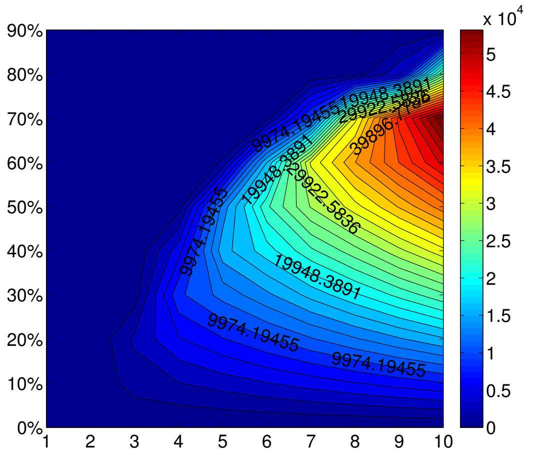
Figure A.5: Ineffective Transmission



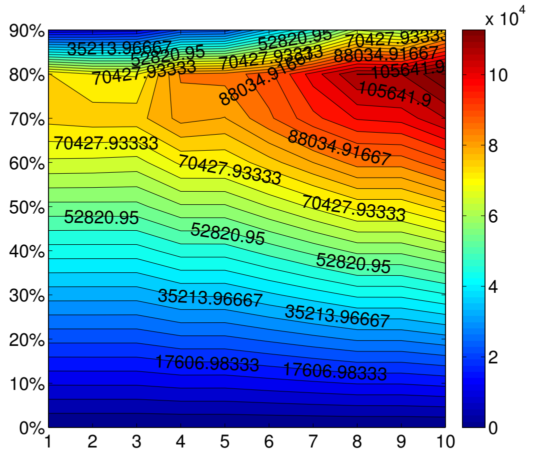
(a) 2^{10} Peers (succ only)



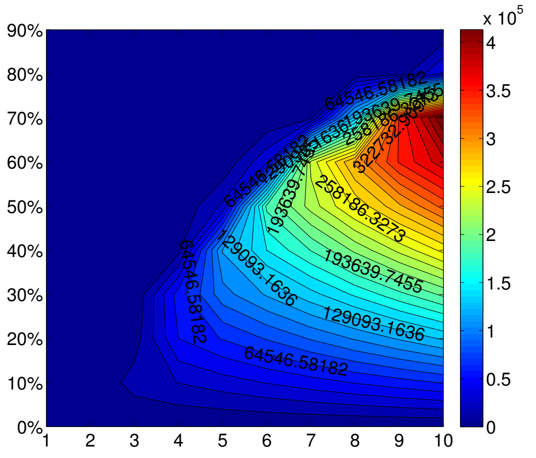
(b) 2^{10} Peers (succ w/ fing)



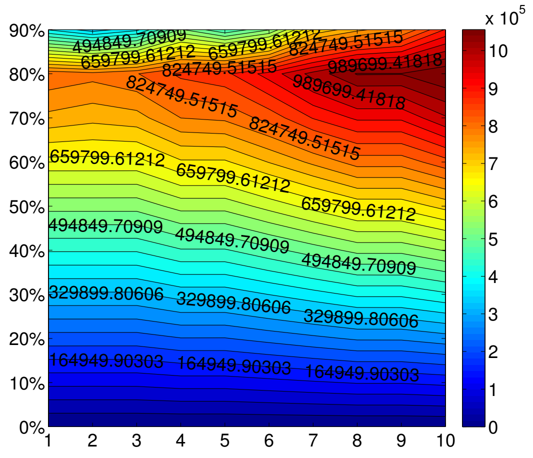
(c) 2^{13} Peers (succ only)



(d) 2^{13} Peers (succ w/ fing)

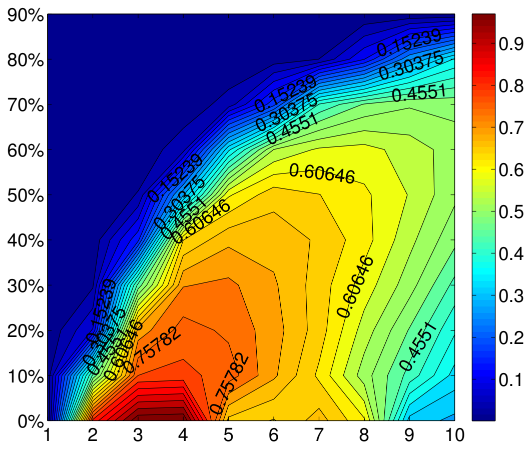


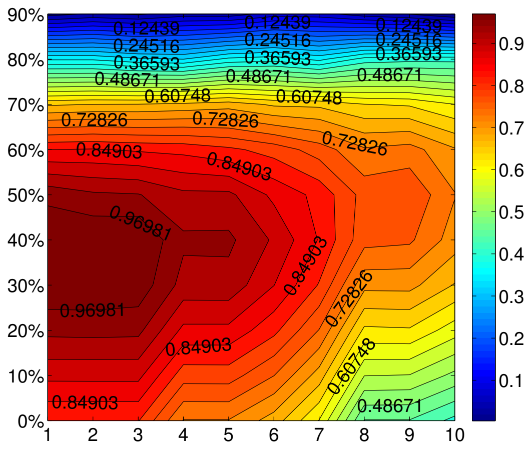
(e) 2^{16} Peers (succ only)

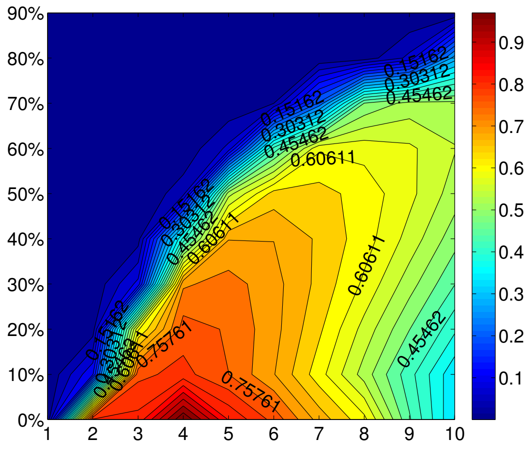


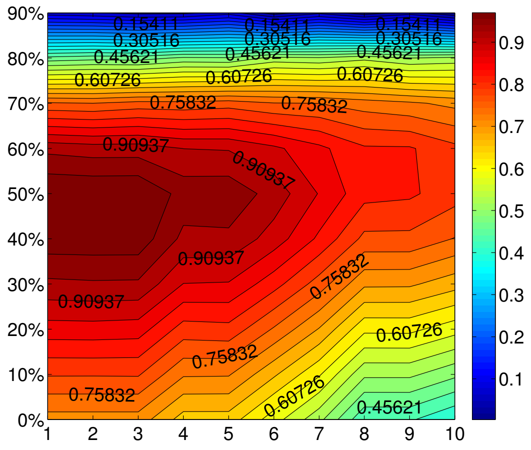
(f) 2^{16} Peers (succ w/ fing)

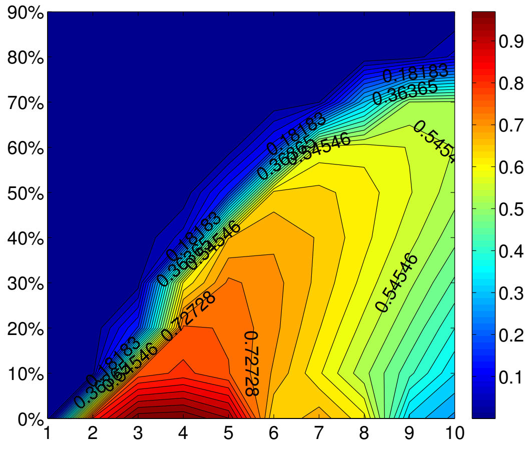
Figure A.6: Connection Failure


(a) 2^{10} Peers (succ only)


(b) 2^{10} Peers (succ w/ fing)


(c) 2^{13} Peers (succ only)


(d) 2^{13} Peers (succ w/ fing)


(e) 2^{16} Peers (succ only)

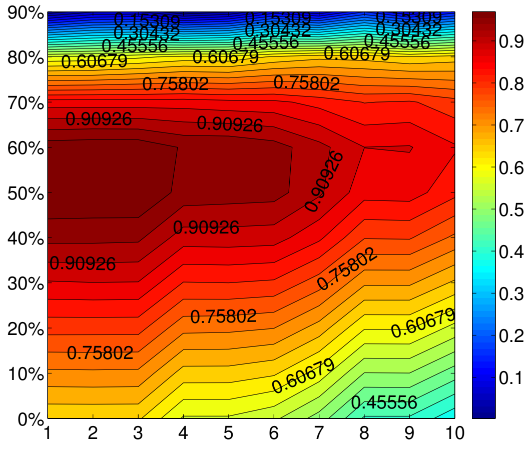

(f) 2^{16} Peers (succ w/ fing)

Figure A.7: Total Efficiency

List of Publications

Journal Papers

[1] Yoshio Sakurauchi and Hideyuki Takada. Efficient Information Dissemination and Reputation Aggregation for User Centric Media Utilizing the Circular Board Method. *Journal of Information Processing*, Vol. 22, No. 1, January, 2014 (in press).

[2] Yoshio Sakurauchi, Rick McGeer, and Hideyuki Takada. OpenWeb: Seamless Proxy Interconnection at the Switching Layer. *International Journal of Networking and Computing*, Vol. 1, No. 2, pp. 157–177, July, 2011.

[3] Yoshio Sakurauchi, Masahiro Ohnishi, and Hideyuki Takada. An Information Exchange Platform by Utilizing the Overlay Location Network Based on User Similarity Considering on Place and Time. *IPSJ Journal*, Vol. 50, No. 12, pp. 3261–3271, December, 2009 (in Japanese).

Refereed Conference Papers

[4] Yoshio Sakurauchi, Rick McGeer, and Hideyuki Takada. OpenWeb: Seamless Proxy Interconnection at the Switching Layer. *The First International Conference on Networking and Computing 2010 (ICNC2010)*, pp. 285–289 (Short Paper), Higashi Hiroshima, November, 2010.

[5] Yoshio Sakurauchi，Masahiro Ohnishi，and Hideyuki Takada. An Information Exchange Platform by Utilizing the Overlay Location Network Considering on Place Similarity. *Groupware and Network services Workshop 2008 (GNWS2008)*, pp. 91–96, November, 2008 (in Japanese).

[6] Syotaro Torikoshi, Tatsuma Kakiuchi, Wataru Ueda, Yoshio Sakurauchi, Kiyotaka Goto, Hideyuki Takada, Oki Fujiwara, Yuya Moriguchi, and Yuki Yamamoto. The Development and Evaluation of SnowBoy: A Programming Environment for Children with Collaborative 3D Graphics Creation. *Interaction 2010*, PA11, March, 2010 (in Japanese).

Technical Reports

[7] Yoshio Sakurauchi and Hideyuki Takada. S-Ring: A P2P-based Information Diffusion Environment Utilizing the Ring Topology. *Groupware and Network services Workshop 2012 (GNWS2012)*, November, 2012.

[8] Yoshio Sakurauchi and Hideyuki Takada. S-Ring: A P2P-based Information Diffusion Environment Utilizing the Ring Topology. *The Fourth Asian Joint Workshop on Information Technology (AJWIT2012)*, pp. 27–31, October, 2012.

[9] Yoshio Sakurauchi and Hideyuki Takada. An Information Exchange and Diffusion Environment Based on the Information Stream through User Terminals. *83rd Groupware and Network services*, 2012-GN-83, No. 4, March, 2012.

[10] Tatsuma Kakiuchi, Shotaro Torikoshi, Yoshio Sakurauchi, Tadayuki Otowa, Shogo Noguchi, and Hideyuki Takada. SnowBoy: A Collaborative Learning Support System for Collaborative Creation by Sharing Programming Projects in Classroom. *72nd Groupware and Network services*, GN72-2, May, 2009 (in Japanese).

Convention Records

[11] Yoshio Sakurauchi and Hideyuki Takada. Proposal of an Information Exchange and Diffusion Environment Based on the Information Stream through User Terminals *The 10th Forum on Information Technology (FIT2011)*, M-040, September, 2011 (in Japanese).

[12] Yoshio Sakurauchi, Rick McGeer, and Hideyuki Takada. Implementation and Evaluation of Server Load-balancing with Network Caching Using the Independent and Distributed Layer-2 Transparent Proxy. *The 9th Forum on Information Technology (FIT2010)*, M-036, September, 2010 (in Japanese).

[13] Yoshio Sakurauchi, Rick McGeer, and Hideyuki Takada. OpenProxy: Dynamic Controllable Layer-2 Transparent Proxy System Implemented on an OpenFlow Programmable Switch. *The 72nd Convention of IPSJ*, 5ZB-4, March, 2010 (in Japanese).

[14] Yoshio Sakurauchi and Hideyuki Takada. An Information Exchange Platform by Utilizing the Overlay Location Network Considering on Place Similarity. *The 7th Forum on Information Technology (FIT2008)*, M-020, September, 2008 (in Japanese).

[15] Jyunya Yamamoto, Yoshio Sakurauchi, Fumihiro Nakagawa, Kiyotaka Goto, Naoya Kadota, Shuhei Nishiguchi, and Hideyuki Takada. Sharing Microblog using Pass-by Connection. *Interaction 2011*, 1CR3-1, March, 2011 (in Japanese).

[16] Jyunya Yamamoto, Yoshio Sakurauchi, and Hideyuki Takada. An Information Categorization Method for Mobile Information Exchange Considering Time Constraints of Passing People. *The 9th Forum on Information Technology (FIT2010)*, M-009, September, 2010 (in Japanese).

[17] Satoru Suzumoto, Yoshio Sakurauchi, and Hideyuki Takada. Evaluation of Management Methods for P2P-based Replicated Objects on Heterogeneous Devices. *The 72nd Convention of IPSJ*, 2ZC-6, March, 2010 (in Japanese).

[18] Satoru Suzumoto, Yoshio Sakurauchi, and Hideyuki Takada. P2P-based Object Replication Environment for Collaborative Work Support Systems Utilizing Mobile Devices. *The 8th Forum on Information Technology (FIT2009)*, M-046, September, 2009 (in Japanese).