World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:3, No:9, 2009

A P2P File Sharing Technique by Indexed-Priority Metric

Toshinori Takabatake and Yoshikazu Komano

***Abstract*—Recently, the improvements in processing performance of a computer and in high speed communication of an optical fiber have been achieved, so that the amount of data which are processed by a computer and flowed on a network has been increasing greatly. However, in a client-server system, since the server receives and processes the amount of data from the clients through the network, a load on the server is increasing. Thus, there are needed to introduce a server with high processing ability and to have a line with high bandwidth. In this paper, concerning to P2P networks to resolve the load on a specific server, a criterion called an Indexed-Priority Metric is proposed and its performance is evaluated. The proposed metric is to allocate some files to each node. As a result, the load on a specific server can distribute them to each node equally well. A P2P file sharing system using the proposed metric is implemented. Simulation results show that the proposed metric can make it distribute files on the specific server.

***Keywords*—peer-to-peer, file-sharing system, load-balancing, dependability.

I. Introduction

Recently, the improvements in processing performance of a computer and in high speed communication such as an optical fiber have been achieved. Services of content distribution, e.g., music or movie delivering, make them possible to be provided by service providers. From this, the amount of data which are processed by the computer and flowed on a network has been increasing greatly. In a conventional client-server system, the server provides these services to many clients. However, since the server receives data from the clients and it processes the data through the network, the load on the server is increasing. Thus, for handling the load, it is necessary to introduce a server with high processing ability and to have a line with high bandwidth [1].

To overcome the problem mentioned above, peer-to-peer (P2P) techniques [1]–[3] have been recently focused on the research of the file sharing system. P2P is defined as a way of structuring distributed applications such that the individual nodes have symmetric roles in [3]. P2P networks or P2P overlay networks have been well surveyed in [4],[5]. Many P2P storage techniques have been also surveyed in [6].

There are many P2P networks [7]–[19]. In P2P networks, each node is searching a connected node based on indexing information. The indexing information is consists of pairs of *key* and *value*. Thus, it is very important to how to manage the indexing information. The P2P networks can classify into two classes based on the way of managing the indexing information: *unstructured* and *structured*. The unstructured P2P overlay networks are as follows: Napster [7], Gnutella[8], Freenet[9], BitTorrent[10], KaZaA [11], etc. In the unstructured P2P overlay networks, the network uses the flooding-based techniques for searching a node [4],[5]. However, each node sends a query at once to groups of the adjacent nodes. Thus, the flooding-based techniques cause some traffic on the network and delay to search the files of the node.

On the other hand, the structured P2P overlay networks are as follows: Chord[14], CAN[15], Tapestry[17], Kademlia[16], Pastry[18], Viceroy[19], etc. The structured P2P overlay networks can make it possible to reduce the number of the queries exchanged between the nodes regardless of the system size increasing. However, in the access frequency to data on the structured P2P overlay networks, there is a node which holds some files with a high popularity, so that the node makes it to concentrate a load by accessing from users [4],[5]. Especially, since each node in the P2P system manages some files by sharing them, each node is prone to centralize a load depend on the way of managing the files. Thus, it is important to how to allocate the files to each node by criteria.

In this paper considering the structured P2P overlay networks, to overcome the problem mentioned above, a criterion called an Indexed-Priority Metric is proposed and its performance is evaluated. The proposed metric is to allocate some files to each node so that the load on a server can distribute them to each node equally well. In addition, a P2P file sharing system based on the proposed metric is implemented. Simulation results show that the proposed metric can distribute files to each node well.

The rest of this paper is structured as follows. Section II gives the preliminary in the proposed method. In Section III, we present the proposed indexed-priority metric, and in Section IV, we present the communication in the P2P system. In Section V, we evaluate the performances of the proposed metric as the file-priorities for the large number of files by simulations. Finally, Section VI describes the conclusions and the future work.

II. Preliminary

In this section, an overview of the P2P networks, two types of P2P networks, and the Distributed Hash Table (DHT) used in this paper are described, respectively.

A. Overviews of P2P networks

Peer-to-peer (P2P) is defined as a way of structuring distributed applications such that the individual nodes have symmetric roles in [3]. In P2P applications, since the role as a client is not distinct from it as a server, each node can act as both a client and a server. In a P2P network,

Toshinori Takabatake is with the Faculty of Engineering, Shonan Institute of Technology, Fujisawa, email: toshi(at)info.shonan-it.ac.jp

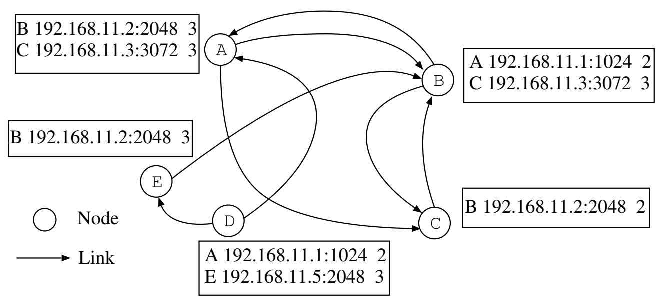


Fig. 1. Example of the proposed method in P2P network.

the processing elements (i.e., computers, storages, etc.) to configure the network are called a node. A P2P network is composed of the nodes of working the P2P applications and the logical links in application layer which are interconnecting between the nodes. P2P systems form their overlay networks based on application layer information [2].

In a client-server system, by comparing with the processes of the server and those of the client, the server mainly executes processes. On the other hand, in a P2P network, all nodes equally execute the processes. The feature of the P2P network is that the performance of the node may not have its high, since the processes in the application are able to distribute to each node. However, since the processes are distributed in each node, it is hard to manage the system as a whole [1]. Although there are many classifications of P2P networks [4],[5], the P2P networks are technically classified into two types by index information: unstructured and structured schemes.

B. Unstructured P2P Overlay Networks

In an unstructured P2P overlay network, each node is connected to any nodes arbitrary. There are also no rules of the location in which data or meta-data is placed. Thus, when each node is searching the data or meta-data, the network uses flooding as the mechanism that each node sends a query at once to groups which are selected among the adjacent nodes. However, the flooding-based techniques cause some traffic on the network and delay to search the data or meta-data. On the other hand, since any topology of the network can compose, it is easy to maintain the system. The unstructured P2P overlay networks are: e.g., Napster [7], Gnutella [8], Freenet [9], BitTorrent [10], KaZaA [11], etc.

C. Structured P2P Overlay Networks

In a structured P2P overlay network, the nodes in which data or meta-data is placed are uniquely determined. Since a topology of the network is also determined according to algorithms in advance, which node being connected to adjacent nodes is defined by the algorithms. Thus, when each node is searching the data or meta-data, the network uses the algorithms as the Distributed Hash Table (DHT)-based systems. The DHT-based systems can find the target data surely. In the systems, the number of forwarding queries which are exchanged between the nodes is small regardless of increasing the number of nodes. However, in the access frequency to data, there is a node which holds data (files) or meta-date with a high popularity, so that the node makes it to concentrate a load by accessing from users. The structured P2P overlay networks are: e.g., OpenNap [12], Chord [14], CAN [15], Kademlia [16], Tapestry [17], Pastry [18], Viceroy [19], etc.

D. Distributed Hash Table

Generally, in a distributed hash table, a hash value is obtained by a hash function [1],[13]. The hash values which are made them possible to obtain by the function are ranged from zero to billions. Theoretically, by using these values in the distributed hash table, each node is almost assigned to a unique hash value, even if there are billions of nodes in the network. Note that, given the hash function denoted by $hash$ and two files denoted by x and y, even if the content of x is almost the same as that of y, $hash(x)$ and $hash(y)$ are quite different values. From this property, since a node can manage some files, a load of the communication on the node can balance to the other nodes.

III. Proposed Method

In this section, an overview of the proposed method, an indexed-priority metric, and a node list are presented.

A. Overview of Proposed Method

In this paper, concerning to the structured P2P overlay networks, a distributed file sharing system is built based on the DHT as well as that in used in Chord [13],[14]. In case of implementing the DHT in a P2P system, it is important how to allocate a file to a node by criteria. Note that a file in this paper is not a file itself but a file containing a meta-data such as information about a directory, file name, date, etc.

Fig.1 shows an overview of the proposed method. In communication between nodes in the system, the proposed method is managing to allocate a file to prevent a specific node from being concentrated a load by accessing from users. In order to decentralize many files for a specific node intensively, the proposed method is made some files distributed to various nodes by introducing a criterion called an indexed-priority metric. In addition, the proposed method is made possible to add or delete some files (meta-data) by using a directory.

The proposed metric has a node- and file-priorities. These priorities are obtained by a hash value from its function detailed in the next subsection. These priorities are recorded in a list in each node. The list is called a node-list.

The overview of the processing for a P2P file sharing system in this paper is described in the following. Note that these processing does not need sequentially because each node is behaved as independently.

(Updating a node-list (1)) : Each node is periodically updating a node-list by connecting the node which has the highest node-priority. At this point, in the node-list of the connecting node, the lowest node-priority is deleted, and the highest node-priority of the connected node is added.

(Updating a node-list (2)) : Each node is conforming for the existence of nodes in the node-list by periodically sending

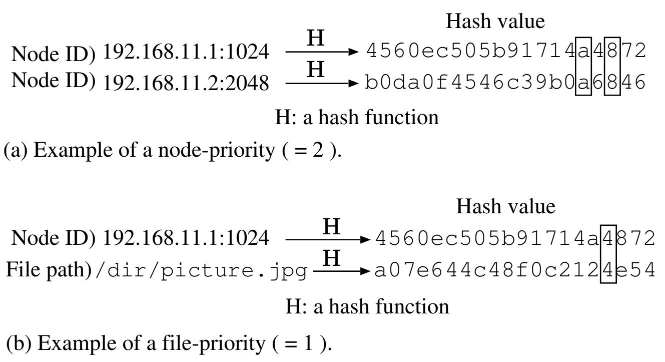

Hash value
Node ID) 192.168.11.1:1024 → H → 4560ec505b91714a4872
Node ID) 192.168.11.2:2048 → H → b0da0f4546c39b0a6846
H: a hash function
(a) Example of a node-priority (= 2).
Hash value
Node ID) 192.168.11.1:1024 → H → 4560ec505b91714a4872
File path) /dir/picture.jpg → H → a07e644c48f0c2124e54
H: a hash function
(b) Example of a file-priority (= 1).

Fig. 2. Examples of the node- and file-priorities.

a signal (e.g., Ping) to the nodes. If the nodes have been not conformed for, the nodes are deleted in the node-list.

(Adding or updating of file) : When each node is adding or updating a file, the each node selects a node from the node-list in which has the highest file-priority, so that the each node connect the selected node. Furthermore, if there is a node which has a higher file-priority in the node-list of the connecting node, the each node will connect the node. At this point, if there is not a file in the connected node, the each node is sending the file to the connected node; on the other hand, if there is not a file-priority in the node-list, the each node is sending the file-priority.

(Referring or deleting of file) : When each node is referring or deleting a file, the each node traces a node of the node-list back by connecting a node in the same way in the adding or updating of a file. So that the target node which has a file is possible to be found.

(Synchronization of file) : Each node which has a file is periodically synchronizing a node which has the highest file-priority by the same way in the updating a node-list. That is, a copy of the file is sent to the node. This is because to prevent from disappearing the file on the network, where the node with the highest file-priority has the copy of the file.

B. Indexed-Priority Metric

An indexed-priority metric is criteria to decide to which each node connects one in communication between nodes and in transferring files, which are needed for the maintenance in the system. Each node has a priority value as an index, which is used for the selection of the connecting node.

Now, an index and a node-ID which are needed to find the priority is defined in the following:

Definition 1: **(Index)** : An index of a node in this paper is a label such as an IP address, a node name, or a MAC address, etc.

Fig. 1 shows an example of the indexes of the nodes such as 'A', 'B', 'C', 'D', and 'E'.

Definition 2: **(Node-ID)** : A node-ID is a string which is combined an index of a node, e.g., an IP address and a port number of it with a symbol, e.g., ':'.

For example, when an IP address of a node is `192.168.11.1` and a port number of it is `1024`, a node-ID

Index	Node ID (IP address + Port No.)	Node, File priority value
A	192.168.11.1:1024	2
B	192.168.11.2:2048	2
C	192.168.11.3:3072	3, 2
D	192.168.11.4:3072	3
E	192.168.11.5:4096	4, 3
F	192.168.11.6:2048	5, 4
G	192.168.11.7:1024	3
H	192.168.11.8:1024	2

Fig. 3. Example of a node-list.

is thus `192.168.11.1:1024`. Next, a node-priority and a file-priority based on the node-ID are defined as follows:

Definition 3: **(Node-priority)** : A node-priority is a sum of the same digit in each hash value which is obtained by a hash function from two node-IDs, where each hash value is scanned from the Most Significant Digit (MSD) to the Least Significant Digit (LSD) of the hash value.

Fig. 2(a) shows an example of the procedure in finding the node-priority. In this example, a node-priority is finding from two node-IDs: an IP address is `192.168.11.1` and a port number is `1024`; an IP address is `192.168.11.2` and a port number is `2048`. First, each hash value is obtained by a hash function (e.g., SHA-1 [1]) from each node-ID. Next, each digit of the obtained hash value is scanning from the MSD, then the same digit is counted. In this example, since 'a' in 16th and '8' in 18th are matched, a node-priority is thus two as shown in Fig.2(a).

Definition 4: **(File-priority)** : A file-priority is sum of the number in the digit being scanned from the MSD of each hash value which is obtained by a hash function from a node-ID and a file pass.

Fig. 2(b) shows an example of the procedure in finding the file-priority. In this example, a node-ID is based on which an IP address is `192.168.11.1` and a port number is `1024`; a file pass is `/dir/picture.jpg`. First, in the same way of finding the node-priority, each hash value is obtained from a hash function from the node-ID and the file pass. Next, each digit of the obtained hash value is scanning from the MSD in order, then the same digit is counted. In this example, '4' in 17th are matched and others are not matched, then the file-priority is thus one as shown in 2(b). Based on the file-priority, a file is allocated in a node.

Note that the node-priority is used for making a node-list which is described detail in the next subsection. Also that the file-priority is used for adding, referring, and deleting a file.

C. Node-List

All communication starts after each node established a connection with a node which has been selected from a node-list defined in the following:

Definition 5: **(Node-list)** : A node-list is a list in which node-IDs of higher node- and file-priorities are saved, respectively.

Fig. 3 shows an example of a node-list. Note that, when the system is implemented, the maximum number of indexes

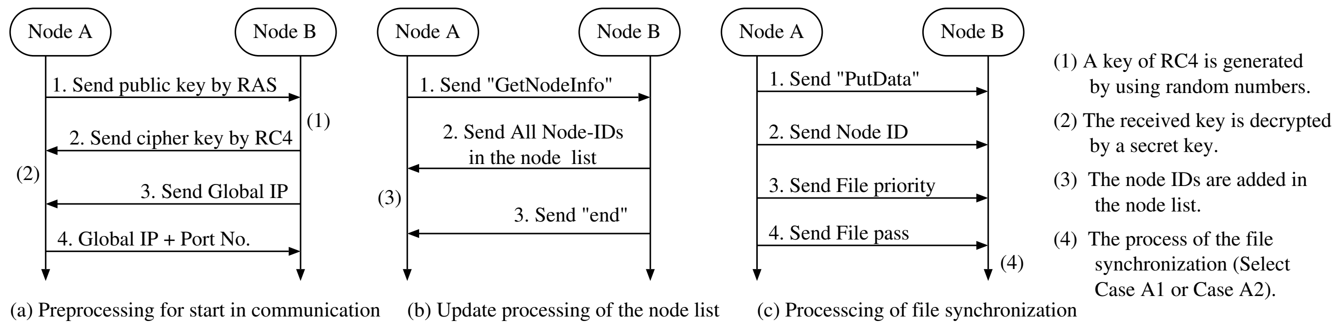

Node A
Node B
1. Send public key by RAS
2. Send cipher key by RC4
3. Send Global IP
4. Global IP + Port No.
(1)
(2)
1. Send "GetNodeInfo"
2. Send All Node-IDs in the node list
3. Send "end"
(3)
1. Send "PutData"
2. Send Node ID
3. Send File priority
4. Send File pass
(4)
(1) A key of RC4 is generated by using random numbers.
(2) The received key is decrypted by a secret key.
(3) The node IDs are added in the node list.
(4) The process of the file synchronization (Select Case A1 or Case A2).
(a) Preprocessing for start in communication
(b) Update processing of the node list
(c) Processcing of file synchronization

Fig. 4. Setup procedures for communication between nodes.

International Science Index, Computer and Information Engineering Vol:3, No:9, 2009 waset.org/Publication/4498

of a node-list can set arbitrarily. However, it is preferable that the size of the node-list is not so large, because of limitations of resources such as memory or HDD in a computer. Also note that the size of the node-list is set as eight in the next section. Now, the procedure of adding a node-ID in a node-list by each node is described in the following. The preceding node-ID which is adding in a node-list is called a candidate node-ID in the following.

Step.1 It is checked whether there is a candidate node-ID which is the same that in the node-list or not. If there is the same one, then this procedure is terminated without adding the candidate node-ID; otherwise, go to Step 2.

Step 2. It is checked whether there is any space to add in the node-list or not. If there is a space, then the candidate node-ID is added and this procedure is terminated; otherwise, go to step 3.

Step 3. A node-ID of the lowest node-priority in the node-list is obtained. Then, the node-priority of the obtained node-ID is compared with that of the candidate node-ID.

Step 4. As a result of Step 3., in the case that the node-priority of the candidate node-ID is equal to or higher than that of the obtained node-ID, the candidate node-ID is not added in the node-list and this procedure is terminated; otherwise, go to Step 5.

Step 5. The node-ID of the lowest node-priority is deleted in the list and the candidate node-ID is added in the node-list, then this procedure is terminated.

Note that the file-priority is added in the node-list by the same procedure of the node-priority.

IV. Communication in the P2P System

In this paper, since a P2P system is implemented on a Window platform, communications in the system are followed by Winsock. Preprocessing for start in communications and processing for the system maintenance are described in the following.

A. Preprocessing for Start in Communication

Fig. 4(a) shows the procedure for start in communications. To protect communication from wiretapping or falsification, a public key cryptosystem by RSA [1] is used in all communications. A key which is encrypted by RC4 [1] is exchanged in communication. All IP addresses that are able to obtain by Winsock are one which is allocated in a network card of the sender. Thus, there is need to request for sending the global IP address of the sender to the node of the receiver after exchanging the key.

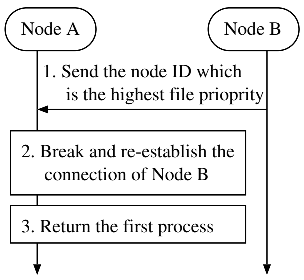
From the above-procedure for communications, the encrypted key is able to send to the node of the receiver safely. Since the procedure is executed in all communications when nodes communicate, the different keys are thus used in every communication.

B. Processing for System Maintenance

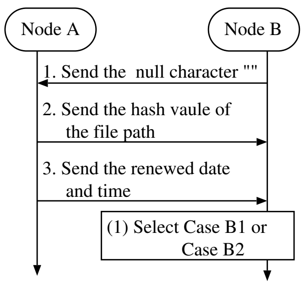
Generally, P2P networks make a node to allow joining and leaving in the network freely. Thus, since the proposed method is subject to P2P networks, the existence of the nodes which they can communicate with each other is always changing. So that in case that a node with a higher node- or file-priority joins the P2P network, the node must be added in the node-list. It is necessary for the system to detect the node which was left from the network and to delete it from the node-list. Furthermore, even if a node which has managed files could leave from the network, the files must be protected to vanish on the network. In this way to maintain a function as a file system, some files are needed to copy to several nodes to some extent.

In this paper, to execute communication to be needed for the system maintenance in the mentioned above, while a node becomes a waiting status temporally, the node connects to the other nodes periodically. Note that, in the implementation described in the next section, the interval of the connection is set to 30 seconds. The operations in the waiting status are three processes in the following:

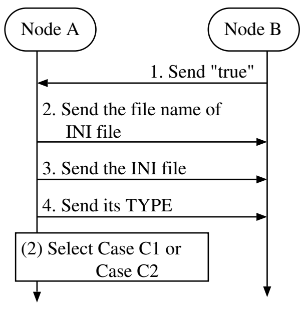
- **Updating a node-list** : The target node of updating a node-list is selecting a node of the highest node-priority from the node-list and it connects to the selected node. To do so, first, after processing for start in communication as shown in Fig. 4(a), the node-list is updated. Fig. 4(b) shows the procedure of updating the node-list.
- **Conformation of a node alive** : Each node is tried to connect all nodes which are recorded in the node-list. If the connection has succeeded, the existence of the nodes is confirmed; on the other hand, if it has not succeeded, the nodes to which could be not connected are deleted in the node-list.


Node A
Node B
1. Send the node ID which is the highest file prioprity
2. Break and re-establish the connection of Node B
3. Return the first process

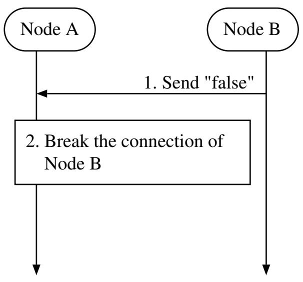
(a) File synchronization (Case A1: the received file priority of Node A is higher than that of Node B).


Node A
Node B
1. Send the null character ""
2. Send the hash vaule of the file path
3. Send the renewed date and time
(1) Select Case B1 or Case B2

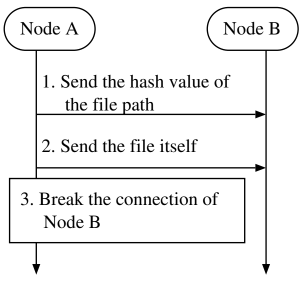
(b) File synchronization (Case A2: the received file priority of Node A is lower than that of Node B).


Node A
Node B
1. Send "true"
2. Send the file name of INI file
3. Send the INI file
4. Send its TYPE
(2) Select Case C1 or Case C2

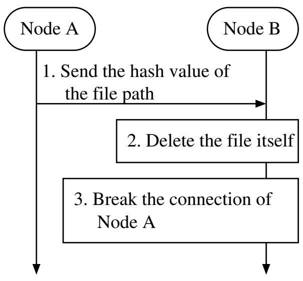
(c) File synchronization (Case B1: the renewed date and time of Node A are newer than those of Node B).



(d) File synchronization (Case B2: the renewed data and time of Node A are not newer than that of Node B).


Node A
Node B
1. Send the hash value of the file path
2. Send the file itself
3. Break the connection of Node B

(e) File synchronization (Case C1: TYPE is FILE).


Node A
Node B
1. Send the hash value of the file path
2. Delete the file itself
3. Break the connection of Node A

(f) File synchronization (Case C2: TYPE is DELETED).

Fig. 5. Procedures for file synchronization.

- **File Synchronization** : The file which is allocated in each node is saved in "cache" folder. Each node is periodically scanning the folder and they are tried to synchronize a node of the highest file-priority in the node-list.

Fig. 4(c) and Fig. 5 show the procedures of file synchronization. As shown in Fig. 4(c)(4), the node B obtains the highest file-priority of the node-ID from the node-list. The obtained file-priority is compared with the received file-priority. From this result, the procedure is executed as follows:

```
if (the received file-priority of Node A
    is higher than the file-priority of
    Node B) {
    Case A1;
} else {
    Case A2;
}
```

In Case A2 of the process in Fig. 5(b), Node B is compared the renewed date and time in the received file of Node A with these in that of Node B. From this result, the procedure is executed as follows:

```
if (the renewed date and time of Node A
    are newer than the date and time of
    Node B) {
    Case B1;
} else {
    Case B2;
}
```

In addition, in Case B1 of the process in Fig. 5(c), Node A executes as follows:

```
if (the received TYPE is FILE) {
    Case C1;
} else if (the received TYPE is DELETED) {
    Case C2;
}
```

Figs. 5(e) and (f) show the processes of Case C1 and Case C2. Case C1 is to transfer a file or Case C2 is to delete a file.

From the above-mentioned procedures of file synchronization, a function as a file system can maintain in the P2P network even if the existence of the nodes is always changing.

V. Simulations

In this section, simulation environment, simulation method, and simulation results are described.

A. Simulation Environment

Since limitations existed in the physical environment to execute simulations, it was difficult to verify the proposed method by using many computers actually. Thus, to achieve the proposed method, simulations were performed on several nodes virtually by using different port numbers on a computer. A program was executed with the port numbers changing only, so that communication between nodes was achieved in the following environment. To also achieve a number of communications in the simulation, many programs were executed. The simulation environment was as follows: CPU is Athlon64 3500+, memory is 1GB, OS is Windows XP SP2, compiler is Visual Studio .NET 2003, and programming language is C++.

B. Simulation Method

First, the simulation was executed to verify whether files were distributed on each node equally or not, where the number of files was 40000 and 160000 as the number of nodes was 4, 16, and 64, respectively. To execute this simulation, a program was made for finding a file-priority. In the program, each node counted the file-priority for each file, respectively, and the sum of the file-priorities was calculated. Next, the output of the program for each node was as follows: "`IP address, hash value, sum of file-priority`". For example, the output was "`192.168.1.1:1024, 6c0458f1b7, 24740`".

If the sum of the file-priorities is not so far away from the average of them, it is preferable because the files are distributed equally.

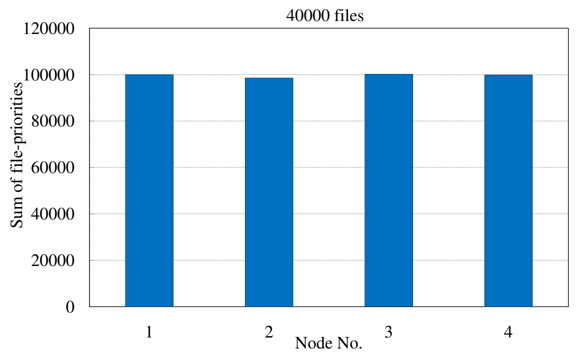


Fig. 6. Sum of file-priorities for 40000 files as the number of nodes with 4.

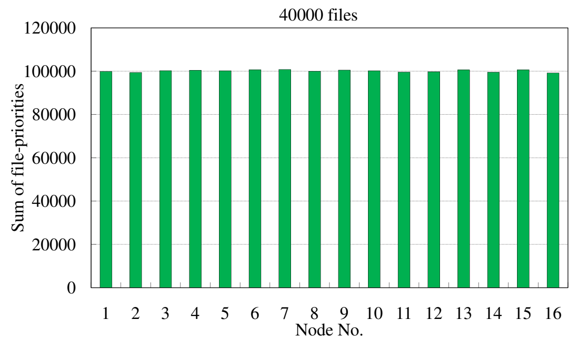


Fig. 7. Sum of file-priorities for 40000 files as the number of nodes with 16.

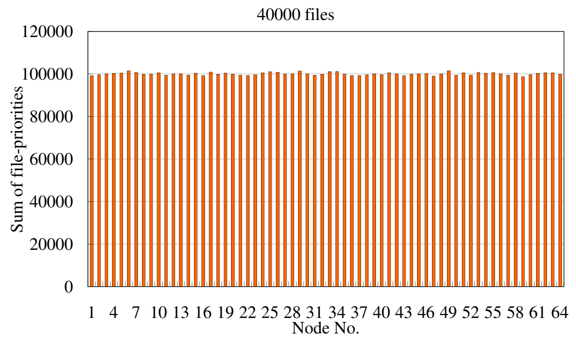


Fig. 8. Sum of file-priorities for 40000 files as the number of nodes with 64.

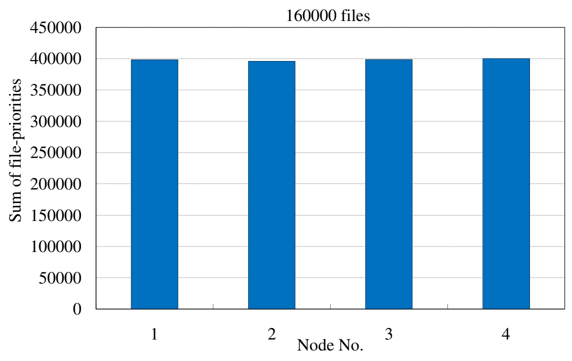


Fig. 9. Sum of file-priorities for 160000 files as the number of nodes with 4.

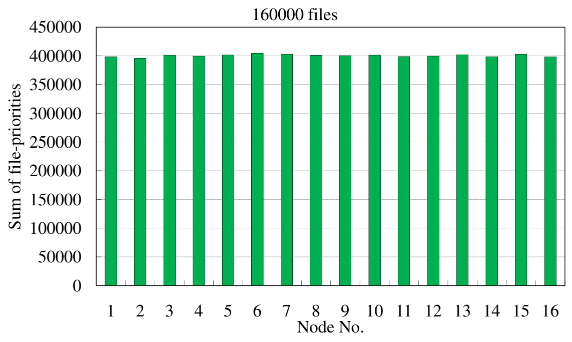


Fig. 10. Sum of file-priorities for 160000 files as the number of nodes with 16.

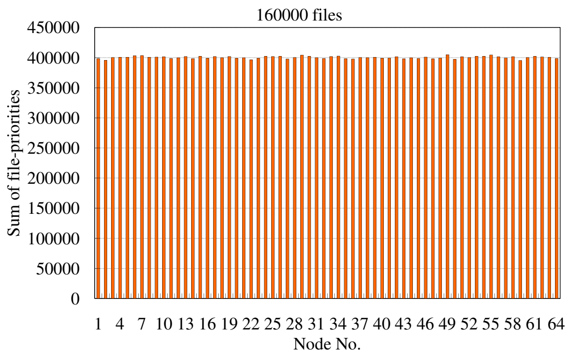


Fig. 11. Sum of file-priorities for 160000 files as the number of nodes with 64.

C. Results

Figs. 6–8 for which the number of files was 40000 show the sum of file-priorities which were allocated to each node as the number of nodes was 4, 16, and 64, respectively. Fig. 9–11 for which the number of files was 160000 show the sum of file-priorities which were allocated to each node as the number of nodes was 4, 16, and 64, respectively. Table I shows the evaluation summary of the file-priorities.

As shown in Figs. 6–8, each node in the sum of the file-priorities was about 100 thousand equally regardless of the number of nodes. In the same as shown in Figs. 9-11, each node in the sum of the file-priorities was about 400 thousand equally regardless of the number of nodes.

Thus, from the figures and the table mentioned above, the sum of file-priorities is able to allocate in each node equally if the number of files are enough for the number of nodes. Since many files make possible to distribute in some nodes by the indexed-priority metric, the load in the nodes can balance. The proposed method can prevent from the load on a specific node managed many files.

In a P2P network, simulation results show that the node- and file priorities make possible to distributed files. In addition, we confirmed to the behavior of the network by a program which communicates between the nodes as tens of computers. The network was built as small-sized one due to the limitations physical resources. Thus, the proposed method is able to used for a P2P network with small-sized build such as LAN or WAN.

VI. Conclusions

In this paper, we proposed the indexed-priority metric that is to distribute some files almost equally to each of nodes in a P2P network. Since the proposed method is able to allocate

TABLE I
EVALUATION SUMMARY IN SIMULATIONS.

No. nodes	40000 files		160000 files	
	Avg.	Std.	Avg.	Std.
4	99677.00	736.37	398305.25	1784.86
16	100098.56	501.60	400390.94	2115.34
64	100094.50	625.24	400331.72	1986.61

these files to each node, a load on a specific node can make balance. To verify this effect of the proposed method, a file system is implemented in a P2P network. Simulation results also show that the proposed method can distribute some files to each of nodes equally.

Future research on the proposed method remains to be explored: there are to verify the work of the effect by building a large-sized network such as a hundreds or thousands of computers, and to develop techniques with fault tolerance.

REFERENCES

[1] Ian J. Taylor, *From P2P to web services and grids: Peers in a client/server world*, Springer, 2005.
[2] G. Camarilla, Ed. "Peer-to-peer (P2P) architectures," draft-iab-p2p-archs-01.txt, April 18, 2009.
[3] IRTF Research Groups, Peer-to-peer research group, http://www.irtf.org/charter?getype=rg&group=p2prg, May 2009.
[4] J. Risson and T. Moors, "Survey of research towards robust peer-to-peer networks: search methods," *Computer Networks*, vol.50, Issue 17, pp.3485–3521, Dec. 2006.
[5] E.K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," *IEEE Communications Surveys and Tutorials*, vol.7, Issue 2, pp.72–93, 2005.
[6] R. Hasan, Z. Anwar, W. Yurcik, L. Brumbaugh, and R. Campbell, "A survey of peer-to-peer storage techniques for distributed file systems," *Proc. Int'l. Conf. Information Technology: Coding and Computing (ITCC 2005)*, vol.2, pp.205–213, 2005.
[7] Napster Inc., http://www.napster.com/, May 2009.
[8] Gnutella Protocol Development, http://rfc-gnutella.sourceforge.net/index.html/, May 2009.
[9] The Free Network Project, http://freenetproject.org/index.html/, May 2009.
[10] Bittorrent Inc., http://www.bittorrent.com/, May 2009.
[11] Kazaa, http://www.kazaa.com/, May 2009.
[12] OpenNap, http://opennap.sourceforge.net/, May 2009.
[13] H. Balakrishnam, M.F. Kaashoek, D. Karger, R. Morris, I. Stoica, "Looking up data in P2P systems," *CACM*, vol.46, Issue 2, pp.43–48, Feb. 2003.
[14] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnam, "Chord: A scalable peer-to-peer lookup service for Internet applications," *ACM SIGCOMM*, pp.149–160, Aug. 2001.
[15] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A scalable content-addressable network," *ACM SIGCOMM*, pp.161–172, Aug. 2001.
[16] P. Maymounkova and D. Mazieres, "Kademlia: A peer-to-peer information system based on the XOR metric," *Proc. 1st Int'l. Workshop Peer-to-Peer Systems*, pp.53–65, 2002.
[17] B.Y. Zhao, L. Huang, J. Stribling, S.C. Rhea, A.D. Joseph, and J.D. Kubiatowicz, "Tapestry: A resilient global-scale overlay for service deployment," *IEEE J. Selected Areas in Communications*, vol.22, no.1, pp.41–53, Jan. 2004.
[18] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," *Proc. IFIP/ACM Int'l. Conf. Distributed Systems Platforms (Middleware)*, pp.329–350, 2001.
[19] D. Malkhi, M. Naor, and D. Ratajczak, "Viceroy: A scalable and dynamic emulation of the butterfly," *Proc. 21st Annual Symp. Principles of Distributed Computing (PODC'02)*, pp.183–192, 2002.