A Semi-structured Overlay Network for Large-scale Peer-to-peer Systems

Kousaku Kimura[1], Satoshi Amamiya[2], Tsunenori Mine[2], and Makoto Amamiya[2]

{Faculty[2], Graduate School[1]} of ISEE., Kyushu University
744 Motooka, Nishiku, Fukuoka 819-0395, Japan
{kimura,roger,mine,amamiya}@al.is.kyushu-u.ac.jp

Abstract. Peer-to-peer (P2P) communication and computing frameworks are important for constructing robust large-scale distributed systems. Overlay network systems use distributed hash-table (DHT) to provide scalable and efficient node search capabilities. However, the DHT-based method has a problem for the maintenance cost of dynamically changing large-scale-network, in which nodes are frequently joining and leaving. This paper proposes a novel technique of P2P communication path management. The proposed technique devises a robust semi-structured overlay network called Ordered Tree with Tuft (OTT for short). OTT provides not only efficient node searching, but also low-cost self-maintenance capabilities for the dynamically changing network. In this method, joining and leaving of a node are managed in $O(1)$ with high probability. Furthermore, the proposed OTT-based technique can find and construct a path shorter than that on the normal ordered tree, by setting up bypass links between remote nodes on OTT.

1 Introduction

Peer-to-peer (P2P) message communication and computing techniques are important to develop large-scale distributed systems. Multiagent systems, in which agents are distributed over the large scale network like Internet, particularly need a flexible P2P message communication environment.

Various P2P systems have been proposed so far. For instance, Gnutella [1] and Freenet [2] are in practical use for file exchange and Skype [3] is used as the IP telephone system. In P2P systems, every node is connected only to a few neighbor nodes, and, for sending a message to a distant node, the sender node asks neighbor nodes to relay the message to the destination node. In this framework, it is important for the P2P system in dynamically changing network to guarantee the soundness in that the message arrives at the destination in safe.

Various kinds of DHT-based P2P algorithms [4][5][6][7][8][9][10][11][12] are proposed to solve the problem by configuring the structured overlay network. In these methods, assuming a key is given, each node or content is mapped onto the structure to optimize the performance of contents search. The search cost in these methods is $O(\log N)$ for the N-node network. In DHT, when a node joins

or leaves the network, its neighbor node has to be selected and its routing table has to be modified to maintain the routing paths in the structured network. If the node fails to keep track of the change, the search process will take more time or the search will be in failure. In general, as the network is larger, the maintenance cost becomes higher. For the large-scale network, the maintenance will become the bottleneck when many nodes frequently join and leave. This bottleneck is serious in practical networks. If the maintenance procedure spreads over the network and thus takes more time, the search will frequently fail. If the maintenance can rapidly be done without spreading over the network, the search can be safer and the network will be more robust.

This paper proposes a semi-structured overlay network which uses a new network topology called Ordered Tree with Tuft (OTT for short) to solve the problem of the maintenance cost for the dynamically changing large-scale network. Assuming every node of OTT to be given a unique ID, OTT is configured as an ordered tree according to the ID. Each node of the ordered tree has a set of ring structured nodes, which is called a **tuft**. Furthermore, each node setups a bypassing route by caching its neighbor node on bypassing route that has been used in the previous communication. Each node of OTT maintains its neighbor nodes, and if one of the neighbor nodes is lost, it searches other paths and updates its neighbor nodes on the paths. The OTT method is effective in performance-by-cost of the route maintenance, because each node can maintain in $O(1)$ with high probability as long as tufts remain. and can find paths to a destination node, whose length is much less than $\log N$ for OTT with N nodes in practical use.

This paper describes the structure of OTT, and gives the method of message routing, path searching and routing table maintenance. The evaluation by software simulation is shown and discussed.

2 Structure of OTT

OTT is an overlay network configured on the physical network (e.g. TCP/IP network). OTT is constructed with an ordered tree and ringed tufts. The ringed tuft is attached to each node of the ordered tree. Figure 1 (left) shows an example of OTT. We call the node of the ordered tree **a tree-node** and the node of the ringed tuft **a ring-node**.

OTT takes advantage of features of the ordered tree structure and the ring structure. The complexity of node insertion, deletion and search are $O(\log N)$ in the ordered tree with N nodes. In contrast, for the ring of size M, the complexity is $O(1)$ for both node insertion and deletion and $O(M)$ for search.

2.1 ID and Adjacent Table

Every OTT node has a unique ID. The ID is given as a tuple of hashed values (u, v). The left child of each tree-node has a u value smaller than that of the parent tree-node, and the right child has a u value larger than that of the parent

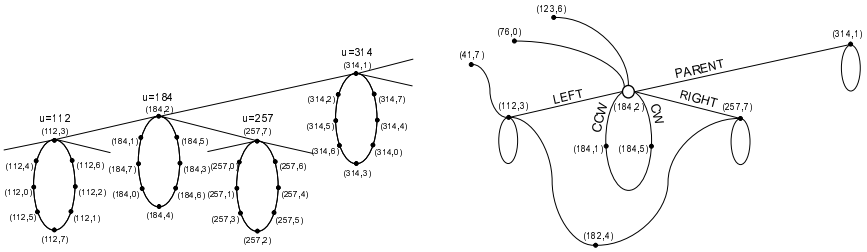

Fig. 1. Example of OTT (left) and connected nodes of (184,2) (right).

tree-node. Each tree-node has a ringed tuft. All nodes in the tuft have the same u value as its tree-node, and each ring-node is identified by its v value. Each node carries an Adjacent Table (AT for short) and a Link Table (LT for short) in order to maintain P2P communication paths on OTT. AT holds IDs of adjacent nodes in OTT, i.e. parent node (PARENT), left child node (LEFT), right child node (RIGHT), clockwise adjacent node in the ring (CW) and counter clockwise adjacent node in the ring (CCW). Figure 1 (right) and table 1 (left) show an example of OTT and AT for the node $(184, 2)$, respectively.

2.2 Link Table (LT)

LT is used for message routing and path searching. All connections of each node are held in LT. Successor nodes held in LT are on the path to its destination. Each LT entry holds several candidates of successor node to the same destination. Thus, every node holds multiple routes to the same destination, and if the current route does not work in some reason, another successor node is selected from the stored candidates. Note that all nodes in AT are also held in LT.

LT also holds successor nodes other than adjacent nodes. We call such a node **a bypass node**. The number of bypass nodes is set to low ratio of the number of the whole LT entries. If the number of bypass nodes is set to the larger, the shorter path will be found, but path search messages will increase and flood over OTT. Therefore the number of bypass nodes should be decided at the practical usage of OTT[1]. Table 1 (right) shows the LT of the node $(184, 2)$.

2.3 Path Search and Min-Max Value

Naive Path Search Method When a sender node holds no path information to a destination node, the sender node searches and sets up the path information to the destination. The search process begins at the sender node. Suppose the ID of the destination node is (u_d, v_d). The search message is delivered to its successor nodes, i.e. adjacent nodes and bypass nodes. When a successor node n_0 with ID (u_0, v_0) receives the message, it checks whether $u_d = u_0$. If $u_d = u_0$ then search

[1] In our experiment, the number of bypass nodes is set from 16 to 32 for the whole numbers of LT entries between 1024 and 8192.

Table 1. Adjacent Table (left) and Link Table (right) of (184,2).

Direction	ID
LEFT	(112,3)
RIGHT	(257,7)
PARENT	(314,1)
CW	(184,5)
CCW	(184,1)

Destination	min-max value	Candidate successor nodes 1st	2nd	3rd	4th
self	$< 82, 272 >$	N/A	N/A	N/A	N/A
(112,3)	$< 82, 135 >$	(112,3)	-	-	-
(257,7)	$< 240, 272 >$	(257,7)	-	-	-
(314,1)	$< 82, 423 >$	(314,1)	-	-	-
(184,5)	$< 82, 272 >$	(184,5)	-	-	-
(184,1)	$< 82, 272 >$	(184,1)	-	-	-
(76,0)	$< 50, 78 >$	(76,0)	-	-	-
(123,6)	$< 121, 125 >$	(123,6)	-	-	-
(41,7)	-	(112,3)	-	-	-
(182,4)	-	(257,7)	(112,3)	-	-

message is delivered into ringed tuft and each ring-node checks whether $v_d = v_0$. Otherwise, the search message is delivered to the left child node if $u_d < u_0$, or to the right child node if $u_d > u_0$.

At the same time, the search message will be delivered to its parent node. In this search process, each node will receive the same search message multiple times. In order to avoid redundant search message propagation, each node, when receiving the same search message, discards the redundant message.

Efficient Path Search with Min-Max Value If the search message is delivered to all the adjacent and bypass nodes, the search messages will flood over OTT. In order to avoid the flooding, we make each tree-node hold a pair of minimum and maximum of u values of its subtree. We call it the min-max value. Suppose a node n_0 has its own u value u_0 and the min-max value $[u_{0-min}, u_{0-max}]$. When the node n_0 receives the search message to the destination node n_d whose ID is (u_d, v_d), if $u_d < u_{0-min} \vee u_{0-max} < u_d$ the node n_0 never delivers the search message into its subtree, because the the target node n_d is not in the subtree. Thus, the flooding of useless search messages is suppressed.

The min-value (max-value) is set to the min-value (max-value) of its left (right) child. If the node has no left (right) child it is set to the u value of the node itself. When the left or right child is replaced with another node or changes its min-max value, the min-max value is recalculated, and the new value is transmitted to all the connected nodes.

3 Protocols

The topology of OTT keeps down the reconstructing cost for a dynamically changing network in which nodes frequently join and leave. In this reconstruction process, mini-max value calculation and its updating messages are refrained from spreading out beyond the neighbor nodes. In the following description, we assume that every node of OTT is assigned to a distributed platform, which is connected to a physical network (e.g. Internet), and every procedure on the node runs on its platform[2].

[2] In our case, multiagent Kodama[13] is used as the platform.

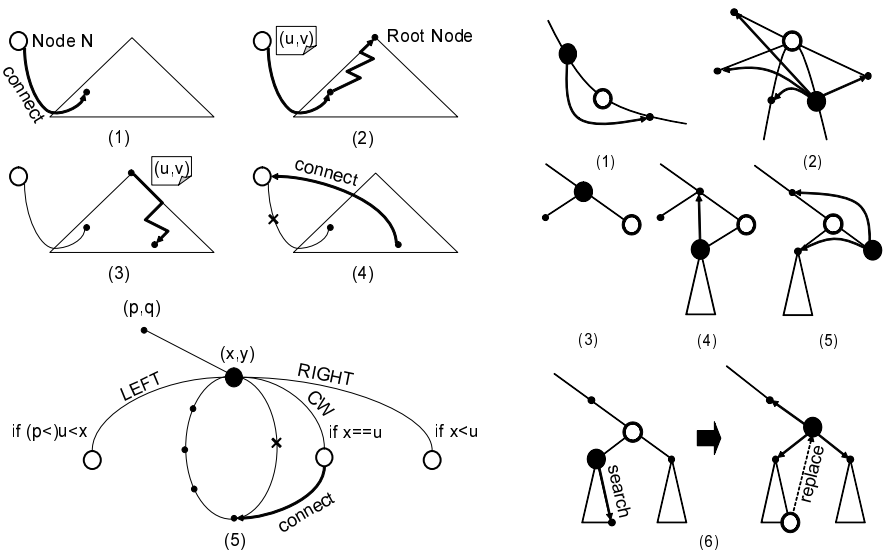

Fig. 2. Join (left) and Leave (right) procedure.

3.1 Joining of Node

When joining, a new node searches for its position in OTT using its ID. In the course of the search, the min-max value of each node is updated if necessary. Figure 2 (left) depicts the procedure when a new node joins OTT. The detail of the procedure is as follows:

(1) When joining OTT, the joining node is connected to another node on a list given by a bootstrap node like [7] or a Web server.
(2) The joining node sends the search message, which consists of its address and ID, toward the OTT root node.
(3) The search message is transmitted from the root through tree-nodes of OTT until the appropriate position to insert the node is found. When a tree-node n_0 receives the message, it compares the u value of the joining node (say u_j) with that of itself (say u_0). If $u_j = u_0$ then the appropriate position to insert the node is found. Otherwise, if $u_j > u_0$ (or $u_j < u_0$), the search message is sent to the right (or left) child node.
(4) When the position is found, the joining node is re-connected from that position and disconnects the node which was connected at the step (1).
(5) Furthermore, the joining node is placed between the tree-node and the right next ring-node. At the same time, the min-(or max-)value of the node n_0 is replaced with u_j if $u_j <$ min-value (or $u_j >$ max-value).

If the tree-node of $u_j = u_0$ is not found, the joining node is a virgin. In this case, the virgin node is connected to the leaf tree-node as its right (or left) child

if $u_j > u_0$ (or if $u_j < u_0$). Both the min- and max-values of the joining node are set to u_j.

When a node joins, the address of all co-existent nodes in its tuft are memorized (in the platform of the node), and the position search process is cut out unless the tuft has disappeared when the node once left out joins again.

3.2 Leaving of Node

We assume OTT uses TCP as its physical network protocol. OTT has the two-way connection between connected nodes, and every node can figure out the status of its connected node. Therefore, the leaving node is quickly detected by its connected nodes, and the OTT reconstruction immediately begins.[3]

Figure 2 (right) depicts six cases of leaving. When a node detects its adjacent node is leaving, it performs one of the following actions according to the position of the leaving node. In any cases, if a node changes its min-max value, the modified value is notified to all its adjacent and bypass nodes.

(1) If the leaving node is a ring-node, its right next ring-node (CW node) is connected to its left next ring-node (CCW node).
(2) If the leaving node is a non-leaf tree-node and has its tuft, the right next ring-node comes to the position of the tree-node and the replaced node is connected to its left next ring-node (CCW), left child (LEFT), right child (RIGHT) and parent (PARENT) tree-nodes.
(3) If the leaving node is a leaf and has no tuft, nothing is done.
(4) If the leaving node is a tree-node with only one child and has no tuft, its child node comes to the position of the tree-node, and the replaced node is connected to its parent tree-node (PARENT).
(5) If the leaving node is a tree-node with both children and has no tuft, then, if its right (or left) child has no child, the right (or left) child is connected to the parent (PARENT) and left (or right) child (LEFT (or RIGHT)) tree-node of the leaving node.
(6) Otherwise, the node with the largest u value is searched in the left subtree and comes to the position of the leaving node, and the connection of the replaced node is modified.

In the above procedure, each node notifies the information of its AT to the nodes registered in its AT and LT so that the nodes keep the connection information. By this, when a node leaves, only its adjacent nodes reconstruct their connections, except for the case (6), as shown in Figure 2 (right). The case (6) requires searching for a node which has the greatest u value in the subtree. However the case (3), (4), (5) and (6) will seldom occur as long as OTT never declines so much as tufts disappear.

Even if a node happens to be in the case that all its successor nodes disappear, the node will be able to take the join procedure as a virgin node.

[3] If the UDP connection is used, periodical confirmation of the node status is required. In this case, detection of node leaving from the network might be delayed.

3.3 Routing

When sending a message to a destination node, the sender node issues the message to its highest-ranked successor node. If the successor node returns no acknowledge, the next highest-ranked successor is chosen and tried again.

The successor node, when having received the message, checks whether the node itself is the targeted destination. If true, the the routing process terminates. Otherwise the successor node relays the message to its highest-ranked successor node. This message relaying continues until the message arrives at the destination node. If any paths to the destination are broken off, the failure message is returned to the sender node, and the sender node begins the path search.

3.4 Path Search

The path search is conducted by two methods: a normal-ordered-tree search and flooding-and-subtree search. The normal-ordered-tree search is carried out on OTT without using bypass links. The flooding-and-subtree search repeatedly issues a search message, using bypass links on OTT, from node to node until the value of TTL (Time To Live) becomes 0.

Assume here the ID of the targeted destination node n_d is (u_d, v_d). When a node n_0 receives the search message, n_0 performs the following procedure:

(1) If the node n_0 is the destination, i.e. $(u_0, v_0) = (u_d, v_d)$, the node returns an answer of the success to the sender node.
(2) Otherwise, if the node n_0 is directly connected to the destination node n_d, it sends the message to n_d.
(3) Otherwise, if $u_0 = u_d \wedge v_0 \neq v_d$, the node n_0 passes the message to the right next ring-node in its tuft to search for the node n_d with v_d.
(4) Otherwise, if the node n_0 finds the successor node n_s from the candidate successor nodes (say n_i's) in LT, n_0 sends the message to node n_s. Here, the node n_s has the min-max value $[u_{s-min}, u_{s-max}]$ such that $(u_{s-min} \leq u_d \leq u_{s-max}) \wedge (u_{s-max} - u_{s-min} = \min\{u_{i-max} - u_{i-min}\})$.
(5) Otherwise,
    (a) When performing the normal-ordered-tree search, the node n_0 sends the message to its adjacent nodes held in AT: If n_0 is a tree-node, it sends to the PARENT node, or if n_0 is a ring-node, sends to the CW node. TTL value is not counted in this search.
    (b) When performing the flooding-and-subtree search, if $TTL > 0$, n_0 sends the message to all the bypass nodes after decrementing TTL.

The difference between the normal-ordered-tree search and the flooding-and-subtree search is only in step (5), which, however, makes considerable difference in performance. If we use the two methods at the same time, the search is sound and can find a shorter path, even though it might make the search cost higher. Furthermore, by combining the two methods, multiple paths can be found, and each node on the paths can select several successor nodes on the shortest path and registers them in LT as the highest-ranked successor candidates.

4 Evaluation

We implemented a software simulator of the OTT method and evaluated for a large-scale network of 4194304($= 2^{22}$) nodes.

In the simulation, the node ID is given by the following ID allocation strategy: The maximum ring size is set to m. When a new node is created, if the ring size is less than m then its ID is set to the same u value as its tree-node and a unique v value in the tuft ring. Otherwise, another u value is selected. In the evaluation, we set $m = 8$, and the ratio of tree-node size to the total ring-node size is 1 to 7. For instance, in the OTT of 128 nodes, the number of tree-nodes is 16, and the total number of ring-nodes is 112 ($= 7 * 16$). In the following discussion, we use two parameters N and B, where N is the number of nodes in OTT, and B the number of bypass links of each node, respectively. Each node makes B bypass links to nodes chosen randomly.

4.1 Cost Performance

First, we evaluate the maintenance cost and path search cost. Here, the cost is defined as the total number of message transfers occurred between nodes during link connection, link disconnection and communication.

Here, N is the number of tree-nodes, B is the number of the bypass links held in a node, and k is the average number of nodes needed to modify their min-max value.

Joining of Node The cost of each step in the joining process described in Section 3.1 is estimated on average as: 1 for the step (1), $\log N$ for the step (2), $2 \log N + k \times B$ for the step (3), 1 for the step (4) and 1 for the step (5) when the joined node is a tree-node or 2 when the joined node is a ring-node.

Figure 3 (left) plots the maintenance cost for the first time joining nodes (virgin nodes), where N varies from 128($= 2^7$) to 4194304($= 2^{22}$) and B=16, 24, 32. The figure shows that the cost is almost proportional to logarithm of the number of nodes, e.g. it is about 110 when $N = 2^{22}$ and $B = 32$. This figure also shows that k is about 2 since the cost of node joining is $2 \log N + kB$, where $2 \log N$ is the cost of normal-ordered-tree search and kB is the cost of min-max value transmission with bypass links, e.g. it is about 2×32 when $B = 32$.

The maintenance cost for re-joining node is $O(1)$, since the position search is cut out unless the tuft has disappeared (see Section 3.1). Figure 4 (left) plots the ratio of survived tufts in the OTT of $N = 2^{22}$, when a leaving node is successively selected at random. This figure shows that only 2% of tufts have disappeared even when a half of 2^{22} nodes have left. Therefore, the maintenance cost for rejoining node is very low, almost $O(1)$ when more than a half of nodes are alive.

Leaving of Node The cost of each step in the node leaving process described in Section 3.2 is estimated as: 1 for the step (1), 4 for the step (2), 0 for the

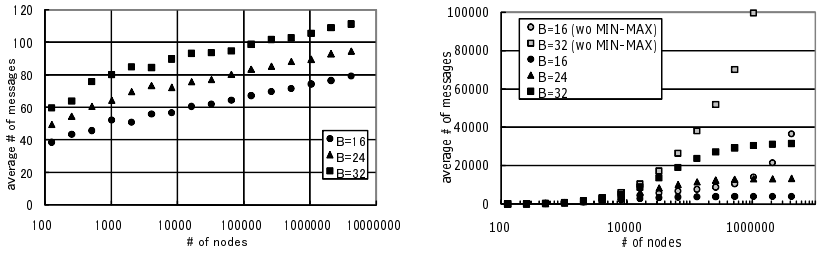

Fig. 3. Average maintenance cost for the first-time joining node (left) and Average cost of path search (right)

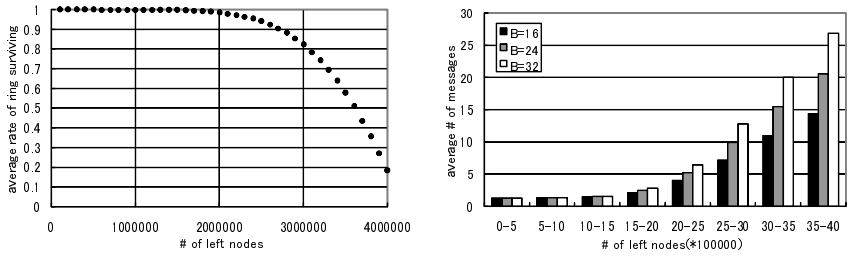

Fig. 4. The ratio of survived tufts in OTT of size 2^{22} nodes (left) and average maintenance cost for the leaving node selected at random and withdrawn from the OTT of 2^{22} nodes (right)

step (3), 1 for the step (4), 2 for the step (5) and $\log M_{sub} + \alpha$ for the step (6). Here, M_{sub} is the number of nodes in the subtree of the leaving node, and α is the cost to the leaving node that has the largest u value and at most 4.

Figure 4 (right) plots the average cost per 500,000 times when the leaving node is successively selected at random from the OTT with $N = 2^{22}$ and $B = 16, 24, 32$. Note that the cost of min-max value transmission is included in this simulation data. This figure shows that the cost is kept very low until a half of 2^{22} nodes has left out. Most of nodes need not reconfigure the ordered tree because their tufts exist. When more than half of nodes withdrew, the number of tree-nodes without tuft increases and the cost becomes higher.

Path Search In the path search evaluation, the pair of sender and receiver nodes is selected at random, and path searching is tried 10000 times for each pair on the OTT of size N ranging from $2^7 (= 128)$ to $2^{22} (= 4194304)$. Figure 3 (right) depicts the average search cost with and without min-max value. The two methods of the normal-ordered-tree search and the flooding-and-subtree search are simultaneously performed. The cost of the normal-ordered-tree search is $O(2 \log N + 8)$ since the cost of climbing up and down on the ordered tree is $O(2 \log N)$ and the number of ring-nodes of each tree-node is 8. In contrast, the cost of flooding-and-subtree search is $O(B^{TTL})$ when the number of bypass links

assigned to each node is B. The simulation is performed for the case of TTL=3 and B=16, 24, 32. In the search process, even if TTL becomes 0 at some node, the destination search continues further downward to its subtree if the node has bypass links or child nodes whose mini-max values cover the destination address. We call this "subtree search." Figure (right) plots the average number of search messages for the case of TTL=3. The figure shows that the search cost strongly depends on the flooding. However, the search cost of the method using the min-max value is largely suppressed and slowly increases to the number of nodes, compared with the method without using the min-max value.

4.2 Path Length

In order to measure the average length of searched paths, we selected two end nodes, search-message-sender node and target node, at random from the set of OTT nodes of size N ranging from $128(= 2^7)$ to $4194304(= 2^{22})$. We measured it 10000 times for each N, for $B = 16, 24, 32$, and $TTL = 3$. Figure 5 (left) depicts the average length of searched paths. This figure shows that the average length

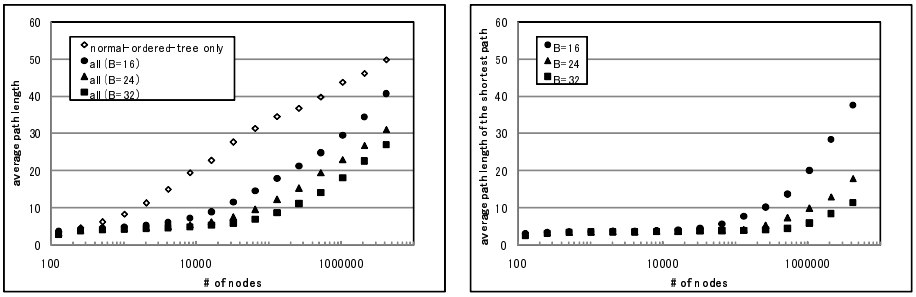

Fig. 5. Average path length(left) and average length of the shortest path(right).

of the paths searched by the normal-ordered-tree search is about $2 \log N$ and is longer than that of paths searched by both the normal-ordered-tree and the flooding-and-subtree searches. Figures 5(right) shows the average length of the shortest path.[4] These results show the effect of the flooding-and-subtree search.

5 Related Work

Overlay networks for the P2P systems are classified to structured and unstructured ones. Gnutella [1] and Freenet [2] are the unstructured networks. Unstructured overlay networks mainly employ a flooding method for node searching. The flooding method has advantages in the point that each node does not have

[4] In the simulation, the path of shortest turnaround time is selected.

to maintain link connections and the searcher node can receive multiple answers. However, the search is not complete because the search message can not be be assured to reach the destination if the TTL is set to a limit to prevent the flooding of search messages.

DHT is one of the most popular methods for structured overlay networks, and various kinds of network configurations are proposed: circle [4], hypercube [7], n-ary tree [5] , B-tree [9][10], butterfly network [11], de Bruijn Graph [12], and so on. Node searching in DHT is complete and its cost is $O(\log N)$. In addition, the processing load is well balanced among nodes because the nodes and contents are evenly scattered. However, DHT is harder to make the search flexible compared to the unstructured network. Although the path length is $O(\log N)$ in DHT, the number of hops does not necessarily mean the network proximity. Therefore, we have to devise the choice mechanism of routes, i.e. the choice of neighbor nodes and ID [14] , by considering the network proximity. In addition, each node have to maintain its bypass link table of size $O(\log N)$ whenever a new node joins or an existing node leaves. As the network becomes larger, the maintenance cost will become higher, and at worst, the search will fall into failure because of the time consuming maintenance [15].

Our method devised on the OTT-based semi-structured overlay network has advantages both the unstructured and structured networks. The flexible and efficient path finding is obtained by applying both the normal-ordered-tree search and the flooding-and-subtree search to the OTT-structure. The network is robust because each node can maintain in $O(1)$ with high probability if ring tufts remain. This will occur until half of OTT nodes leave the network. In addition, choosing the first found route in path searching, we can take into account the network proximity.

6 Conclusion

This paper proposed a novel technique of P2P communication path management. The proposed technique devises a robust semi-structured overlay network called OTT (Ordered Tree with Tuft). The OTT method is effective in performance-by-cost of the route maintenance for dynamically changing network.

The effect of the OTT method is evaluated by software simulation. The simulation result shows that the maintenance cost is $O(1)$ with high probability if a half of nodes in OTT is alive (except for the first joining), and can find one or more paths to a destination node, whose length is much less than $\log N$ for N nodes in practical use, while the path search cost goes up to B^{TTL} in the worst case. Therefore, we can say that OTT keeps robustness by performing the quick maintenance of OTT when a half of nodes in the network is alive. In addition, the flooding-and-subtree search successfully finds short-length paths on OTT by using bypass links.

This method is implemented as an overlay network in Kodama [13] system. As an application of Kodama we are developing a business matching and collab-

oration support system. In this system, each node delivers and receives business plans and company information. Therefore it needs robust overlay network.

Further research issue is to develop the distributed node ID allocation method. The node ID allocation was performed as a centralized process in the simulation. But the ID allocation process should be performed in the distributed environment. Another one is the more precise simulation to reflect practical application. The simulation of this paper separately performed the node join, leave and search processes. However, these three processes will occur concurrently in the practical network services. Therefore the simulation is needed for more realistic dynamic network environments where the node join, leave and search processes occur in a highly concurrent way.

References

1. Gnutella: The gnutella protocol specification v0.4, url: http://www.gnutella.com/ (2000)
2. Clarke, I., Sandberg, O., Wiley, B., Hong, T.W.: Freenet: A Distributed Anonymous Information Storage and Retrieval System. In: the Workshop on Design Issues in Anonymity and Unobservablity. (2000) 46–66
3. web site, S.: (http://www.skype.com/)
4. Stoica, I., Morris, R., Karger, D., Kaashoek, F., Balakrishnan, H.: Chord: A scalable Peer-To-Peer Lookup Service for Internet Applications. In: the 2001 ACM SIGCOMM Conference. (2001) 149–160
5. Rowstron, A., Druschel, P.: Pastry: Scalable, Decentralized Object Location, and Routing for Large-scale Peer-to-Peer Systems. In: IFIP/ACM International Conference on Distributed Systems Platforms (Middleware). (2001) 329–350
6. Zhao, B.Y., Kubiatowicz, J.D., Joseph, A.D.: Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, UC Berkeley (2001)
7. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A scalable content addressable network. Technical Report TR-00-010, Berkeley, CA (2000)
8. Maymounkov, P., Maziéres, D.: Kademlia: A peer-to-peer information system based on the XOR metric. In: ITPTS'02. (2002) 53–65
9. Baquero, C., Lopes, N.: B+tree on p2p: Providing content indexing over dht overlays. Technical report, Universidade do Minho (2004)
10. Prakash, A.C.: P-Tree: A P2P Index for Resource Discovery Applications. In: 13th International World Wide Web Conference. (2004) 390–391
11. Malkhi, D., Naor, M., Ratajzcak, D.: Viceroy: Scalable emulation of butterfly networks for distributed hash tables (2003)
12. Kaashoek, M.F., Karger, D.R.: Koorde: A simple degree-optimal distributed hash table. In: IPTPS'03. (2003) 323–336
13. Zhong, G., Amamiya, S., Takahashi, K., Mine, T., Amamiya, M.: The Design and Implementation of KODAMA System. IEICE Transactions INF **E85-D** (2002) 637–646
14. Song, J., Park, S., Yang, J.: An Adaptive Proximity Route Selection Scheme in DHT-Based Peer to Peer Systems. Parallel and Distributed Computing: Applications and Technologies **3320** (2004) 778–781
15. Rhea, S., Geels, D., Roscoe, T., Kubiatowicz, J.: Handling Churn in a DHT. In: the 2004 USENIX Technical Conference. (2004)