New Structured P2P Network with Dynamic Load Balancing Scheme

Atushi TAKEDA*, Takuma OIDE† and Akiko TAKAHASHI‡
* *Department of Information Science, Tohoku Gakuin University*
† *Department of Information Engineering, Sendai National College of Technology*
‡ *Department of Information Systems, Sendai National College of Technology*

***Abstract*—Many proposals have been advanced for structured P2P networks, but it is difficult for existing structured P2P networks to achieve dynamic load balancing sufficient. In this paper, we propose a new structured P2P network called *Waon*, which achieves dynamic load balancing among nodes. Each node in a Waon network controls the number of objects assigned to the node for load balancing of objects. In addition, Waon uses a hop-based routing algorithm for load balancing of messages. Moreover, Waon can support range queries, and Waon can reduce the load on the physical network. In this paper, through simulation results, we report confirmation that Waon's load balancing is better than that of existing algorithms. In addition, a simulation result in this paper shows that Waon can reduce the load on the physical network.**

***Keywords*-p2p network; overlay network; load balancing;**

I. Introduction

Peer-to-peer (P2P) networks are decentralized networks in which all nodes mutually communicate directly. In fact, a P2P network is more scalable and more robust than a client–server network, so many P2P network applications were developed [1], [2]. Particularly, structured P2P networks of many types have been proposed in recent years [3], [4], [5], [6], [7]. A structured P2P network is a scalable network on which nodes can be searched for objects accurately. Existing structured P2P networks, however, can not achieve load balancing sufficiently because the existing schemes assign objects to nodes statically.

As described in this paper, we propose a new structured P2P network named *Waon: Well-distribution Algorithm for Overlay Network*. Waon achieves not only dynamic load balancing of objects but also dynamic load balancing of messages. Waon uses a ring-type identifier space such as Chord's identifier ring [3]. Nodes and objects in a Waon network are put on the identifier ring, and Waon builds an overlay network based on nodes' locations on the identifier ring. In existing schemes, nodes' locations on the identifier ring are calculated statically. In contrast, in a Waon network, nodes' locations on the identifier ring are modified dynamically for dynamic load balancing.

In a Waon network, objects are placed on the identifier ring and assigned to nodes. The object assignment is determined according to objects' locations and nodes' locations on the identifier ring. In a Waon network, an overloaded node modifies its location on the identifier ring to reduce the number of objects assigned to the node. This paper presents how to modify nodes' locations on the identifier ring in a Waon network.

Existing routing algorithms of structured P2P networks assume that nodes are located uniformly on the identifier space. However, in a Waon network, nodes might be located on the identifier ring nonuniformly. Therefore, Waon can not use existing routing algorithms. Waon uses a hop-based routing algorithm. In a Waon network, each node builds a routing table based on the number of hops between nodes. Waon's routing algorithm achieves load balancing of messages even if nodes are placed nonuniformly on the identifier ring. We explain the proposed routing algorithm in this paper.

As described above, Waon's features are following:

1) Each node modifies its location on Waon's identifier ring dynamically to achieve load-balancing of objects.
2) Each node builds a routing table based on the number of hops between nodes to achieve load-balancing of messages.

Features described above give Waon the following advantages over existing structured P2P networks:

1) Waon can support range queries for searching objects. Waon does not assume uniform distribution of objects on the identifier ring. Therefore, Waon enables placement of objects on the identifier ring contiguously; Waon can thereby support range queries such as SkipGraph's range queries [4].
2) Waon can reduce the traffic on the physical network. Waon does not assume uniform distribution of nodes on the identifier ring. Consequently, Waon can reflect the physical network topology into the nodes' location on the identifier ring. Waon can thereby reduce the traffic on the physical network.

Waon's algorithm proposed in this paper is a basic algorithm. For these analyses, we assume a simple environment as follows:

1) The assumed environment is stable. Nodes in a P2P network rarely join the network; also, they rarely leave the network.
2) The assumed environment is homogeneous. All nodes in a P2P network are the same, and popularities of all objects in a P2P network are the same.

The remainder of this paper is organized as follows. Section 2 presents a description of related works and clarifies the issues. Section 3 proposes the Waon algorithm. Section 4 describes evaluation through simulation results. Finally, section 5 presents a summary our conclusions.

II. Related Works

A pure P2P network is classified as an unstructured P2P network or structured P2P network. Nodes in an unstructured P2P network can connect to any other node in the network freely, so we can build an unstructured P2P network based on computing resources or physical locations. However, an unstructured P2P network uses an inefficient search protocol such as *Flooding*. In contrast, a structured P2P network has a topology of an overlay network. A structured P2P network is a scalable P2P network in which nodes can search objects accurately.

The Distributed Hash Table (DHT) is the most famous algorithm used with structured P2P networks. In recent years, many DHT algorithms such as Chord [3], CAN [5], Pastry [6], and Tapestry [7] have been proposed. Nodes and objects in a DHT network are put on an identifier space. A node's location on the identifier space is determined by the node's identifier, which is calculated by consistent hashing of the node's IP address, and the object's location on the identifier is also determined according to the object's identifier, which is calculated by consistent hashing of the object's name. DHT builds an overlay network based on nodes' locations on the identifier space, and DHT assigns objects to nodes based on objects' locations on the identifier space. Nodes in a DHT network have a routing table for efficient object search. The search process of DHT is the same as binary search, so the computing cost required for searching an object is $O(\log n)$, where n is the number of nodes in the DHT network. Therefore, DHT algorithms are scalable. On the other hand, DHT does not support range queries because objects' locations on the identifier space are determined by a one-way hash function, which is a nonlinear function. Additionally, it is not easy to do load balancing in DHT systems because objects are assigned to nodes statically.

SkipGraph is a scalable structured P2P network algorithm [4]. SkipGraph provides efficient object search without hash functions. In a SkipGraph network, each node creates a SkipList [8], and nodes build an overlay network based on their SkipLists. The computational cost of object search in a SkipGraph network is $O(\log n)$, where n is the number of nodes in the network. Additionally, SkipGraph supports range queries for object searching. Aside from SkipGraph, GosSkip has been proposed as a structured P2P network that supports range queries [9]. Actually, GosSkip uses gossip messages to build an overlay network based on SkipList; GosSkip also supports range queries for object search. Both SkipGraph and GosSkip are important proposals because they show that an object search with range queries is possible in structured P2P networks. However, dynamic load balancing in these networks is difficult because these algorithms assign objects to nodes statically.

Dynamic load balancing in structured P2P networks is a challenging study, so many algorithms have been proposed in recent years [10], [11], [12]. These algorithms achieve dynamic load balancing by transferring objects from an overloaded node to an underloaded node. However, these algorithms assume that nodes and objects are located on the identifier space uniformly. Therefore, these algorithms do not support range queries. Mercury is a structured P2P network which supports dynamic load balancing and range queries [13]. Mercury uses a hop-based routing algorithm, so Mercury achieves load balancing of messages even if nodes are located on the identifier ring nonuniformly. Mercury, however, uses the join–leave algorithm for load balancing of objects. Therefore, Mercury can not reduce the traffic on the physical network.

The basic concept of our proposed *Waon* is similar to Mercury. However, Waon differs from existing algorithms. Waon has all of the following features:

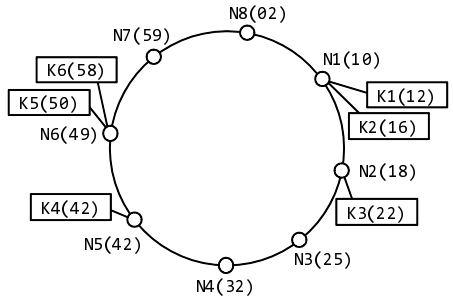
1) Waon is a scalable P2P network.
2) Waon achieves not only dynamic load balancing of objects but also dynamic load balancing of messages.
3) Waon can support range queries for object search.
4) Waon can reduce the traffic on the physical network.

To achieve the requirements described above, Waon has a simple partition control algorithm and a hop-based routing algorithm. The next section presents details of Waon's algorithms.

III. Proposal: Waon

A. Basic concept of Waon

Waon is a structured P2P network that achieves both dynamic load balancing of objects and dynamic load balancing of messages. Waon uses a ring-type identifier space like Chord's identifier ring [3]. Nodes and objects in a Waon network are put on the identifier ring. Subsequently, nodes build an overlay network based on nodes' locations on the identifier ring. In a Waon network, each node modifies the node's location on the identifier ring to decrease the number of objects assigned to the node, when the number of objects assigned to the node is greater than the number of objects assigned to another node. All nodes perform this process independently, and objects are assigned to each node evenly. In a Waon network, many nodes might be located in a narrow area of the identifier ring because nodes can set their location freely. Existing DHT routing algorithms assume a uniform distribution of nodes on the identifier ring. Therefore, Waon can not use existing DHT routing algorithms because many messages are concentrated on a few nodes if Waon uses an existing DHT routing algorithm. Therefore, Waon uses a



(a) Waon's Identifier ring

```
node:N1                              node:N2
 N1.location = 10                     N2.location = 18
 N1.successor = {N2, N3, N4}          N2.successor = {N3, N4, N5}
 N1.predecessor = {N8, N7, N6}        N2.predecessor = {N1, N8, N7}
 N1.route = {N2, N3, N5}              N2.route = {N3, N4, N6}
 N1.object = {K1, K2}                 N2.object = {K3}
```

(b) Examples of nodes' properties

Figure 1. Waon's identifier ring and examples of nodes' properties.

hop-based routing algorithm that achieves load balancing of messages even if nodes are located on the identifier ring nonuniformly. Waon achieves dynamic load balancing of messages using the hop-based routing algorithm.

Waon does not assume uniform distribution of objects on the identifier ring. Consequently, Waon can put objects on the identifier ring contiguously. Therefore, Waon can support efficient range queries for object searches. Furthermore, Waon does not assume uniform distribution of nodes on the identifier ring. Consequently, Waon can reflect the physical network topology into nodes' locations on the identifier ring. Therefore, Waon can reduce the physical network traffic between nodes.

B. Waon's identifier ring

Fig. 1(a) portrays a Waon identifier ring. Waon's identifier ring is a circle indexed $[0, 2^m)$ linearly (Usually, 2^m is a huge number such as 2^{128}). All nodes in a Waon network are put on this identifier ring. A node's location on the identifier ring is specified in the node's property *location*. Each node in a Waon network can modify its property *location* freely to change its location on the identifier ring.

A node in a Waon network is defined as follows.

$$\begin{aligned} \mathit{node} &:= < \mathit{location}, \mathit{successor}, \\ &\quad \mathit{predecessor}, \mathit{route}, \mathit{object} > \\ \mathit{location} &:= \mathit{INTEGER} \\ \mathit{successor} &:= \{\mathit{node}_0, \mathit{node}_1, \cdots, \mathit{node}_{r-1}\} \\ \mathit{predecessor} &:= \{\mathit{node}_0, \mathit{node}_1, \cdots, \mathit{node}_{r-1}\} \\ \mathit{route} &:= \{\mathit{node}_0, \mathit{node}_1, \cdots, \mathit{node}_{\log(n)-1}\} \\ \mathit{object} &:= \{\mathit{object}_0, \mathit{object}_1, \cdots\} \end{aligned}$$

```
// update location of a node n
n.updateLocation()
    sum = count(n.object);
    for(i = 0; i < r; i = i + 1)
        sum = sum + count(n.predecessor[i].object);
    ave = sum/(r + 1);
    num = count(n.object) − ave;
    if(num > 0)
        object = sortById(n.object);
        n.location = object[num].id;
        for(i = 0; i < num; i = i + 1)
            n.predecessor[0].addObject(object[i]);
            n.removeObject(object[i]);
```

Figure 2. Pseudo-code for modifying node location.

In this definition, r signifies the number of neighbor nodes managed by a node, and n denotes the number of nodes in a Waon network. Usually, r is 2 or more, but less than $\log n$. Property *location* is the node's location on the identifier ring. Property *successor* is a list of nodes lying in clockwise direction from the node, and property *predecessor* is a list of nodes lying in a counterclockwise direction from the node. Property *route* represents a routing table for searching objects, and property *object* stands for a list of assigned objects. Fig. 1(b) presents examples of nodes' properties.

All objects are put on the identifier ring, and nodes manage objects lying between the node and the node's first successor node. A relation between a node and an assigned object is defined as

$$\begin{aligned} &\mathit{node}.\mathit{object}[i].\mathit{location} \\ &\quad \in [\mathit{node}.\mathit{location}, \mathit{node}.\mathit{successor}[0].\mathit{location}) \end{aligned}$$

For example, in the circumstances depicted in Fig. 1, node *N1* has object *C1* and *C2* because these objects are located between node *N1* and *N1*'s first successor node *N2*.

C. Load balancing of objects

Nodes in a Waon network change their location on the identifier ring due to load balancing of objects. Fig. 2 shows a pseudo-code for changing node's location on the identifier ring. All nodes repeatedly execute a function **updateLocation** at regular time intervals.

In function **updateLocation**, node *n* performs the following process:

1) Node *n* sends queries to predecessor nodes of node *n* to obtain the number of objects assigned to the predecessor nodes.
2) Node *n* calculates the average of the number of objects assigned to the predecessor nodes. Node *n* is presumed to be overloaded if the number of objects assigned to

```
// update routing-table
n.updateRoute()
    node = n.successor[0];
    distance = 0;
    for(i = 0; distance < distance(n, node); i = i + 1)
        n.route[i] = node;
        distance = distance(n, node);
        node = node.route[i];
```

Figure 3. Pseudo-code for updating a routing table.

node *n* is more than the average of the number of objects assigned to the predecessor nodes.

3) Node *n* modifies the location of node *n* to narrow the area between node *n* and its first successor node on the identifier ring if node *n* is overloaded. As a consequence of this process, the number of objects assigned to node *n* is reduced. After that, node *n* transfers some objects to its first predecessor node.

All nodes in a Waon network repeatedly execute a function **updateLocation** independently. Thereby, Waon achieves load balancing of objects among nodes.

D. Routing algorithm

Nodes in a Waon network change their location on the identifier ring. For that reason, many nodes might be located in a narrow area of the identifier ring. In existing DHT routing algorithms, many messages are concentrated on a few nodes when distribution of nodes' locations on the identifier ring is nonuniform. Therefore, Waon uses a hop-based routing algorithm to facilitate load balancing of messages.

A node in a Waon network has a routing table, which is node's property *route*. Each node creates a routing table under the following rule:

$$node.route[i+1] = node.route[i].route[i]$$

This rule means that node $route[i]$ is the 2^i*th* successor node. Fig. 3 portrays a pseudo-code for updating a routing table. All nodes in a Waon network repeatedly execute function **updateRoute** at regular time intervals. Waon's routing table is not dependent on distribution of nodes' locations on the identifier ring. Therefore, Waon achieves efficient object searching even if nodes' locations on the identifier ring are distributed nonuniformly.

Fig. 4 portrays a pseudo-code for searching for an object. Waon's search process is fundamentally identical to Chord's search process. Therefore, the cost of object search is $O(\log n)$, where n is the number of nodes in the Waon network. Therefore, Waon is a scalable structured P2P network.

```
// search a object k
n.searchObject(k)
    if(distance(n, k) < distance(n, n.successor[0]))
        foreach(o : n.object)
            if(k = o.name)
                return(o);
        return(null);
    else
        next = n.successor[0];
        foreach(i = 0; i < length(n.route); i = i + 1)
            if(distance(n, k) < distance(n, n.route[i]))
                break;
            else
                next = n.route[i];
        return(next.searchObject(k));
```

Figure 4. Pseudo-code for searching an object.

```
var/lib/pgsql/base/17229/16693        => 192.168.1.16
usr/share/man/man3/Net.Ping.3pm       => 192.168.1.5
usr/include/ncurses/term.h            => 192.168.1.9
usr/lib/python2.5/test/test_capi.py   => 192.168.1.12
usr/share/man/man3/clearok.3x.gz      => 192.168.1.17
bin/c++filt.exe                       => 192.168.1.3
lib/ruby/1.8/soap/streamHandler.rb    => 192.168.1.6
lib/ruby/1.8/tk/listbox.rb            => 192.168.1.6
lib/python2.5/hmac.pyo                => 192.168.1.4
```

Figure 5. Objects used for the simulation.

IV. Evaluation

A. Simulation environment

To confirm Waon's advantages over an existing scheme, we evaluated Waon's performance through simulation, which is implemented with Java. In this simulation, all nodes in a Waon network perform the following process at every step.

1) All nodes send *ping* messages to all predecessor nodes and to all successor nodes to obtain information of the nodes.
2) As described in III-C, all nodes modify their location on the identifier ring for load balancing of objects.
3) As described in III-D, all nodes update their routing tables.

In addition, each node searches all objects at every 10 steps. Parameter m, which is the size of Waon's identifier ring, is 64. Parameter r, which is the size of neighbor node list, is 7.

Fig. 5 portrays some objects used in the simulation. This simulation assumes that shared objects in the P2P network are files in a UNIX operating system. Consequently, in this simulation, nodes share a part of files of *FreeBSD 8.0*, which is a UNIX operating system. In this simulation, all objects are small, and a node requires only a message when the

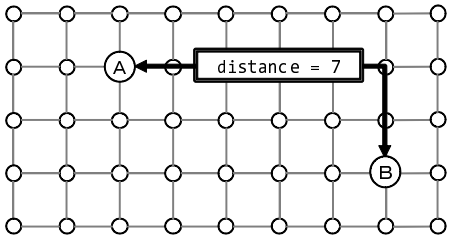


Figure 6. Physical network assumed in the simulation.

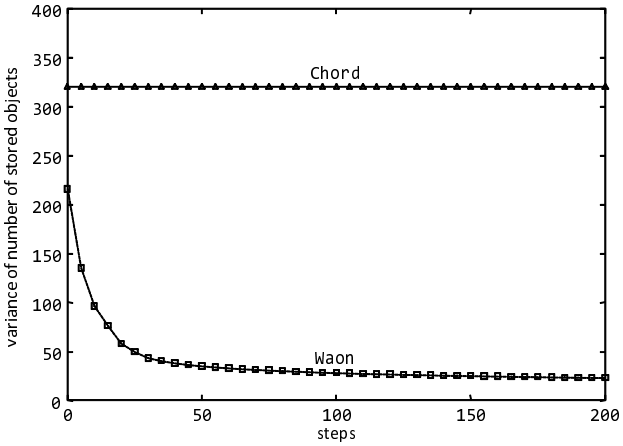


Figure 7. Variance of the number of objects assigned to a node.

node transfers an object to another node. This simulation assumes that shared objects are pointers such as URL. The shared objects are put on the Waon's identifier ring based on the files' paths. Therefore, file objects contained in the same directory are located closely on the identifier ring in this simulation.

Fig. 6 shows the physical network assumed in this simulation. The physical network uses lattice topology. This simulation assumes that failures such as packet loss and packet delay are not caused on the physical network. For example, when 100 nodes run, the physical network assumed in this simulation is a stable 10×10 lattice network.

We performed a simulation described above. This section shows Waon's advantages over Chord [3], a famous structured P2P network, using simulation results.

B. Load balancing of objects

Fig. 7 shows the variance of the number of objects assigned to each node. The variance of Chord is constant because Chord assigns objects to nodes statically. However, the variance of Waon decreases with steps because nodes in a Waon network change the nodes' location on the identifier ring to control the nodes' load as described in III-C. Consequently, Waon achieves load balancing of objects.

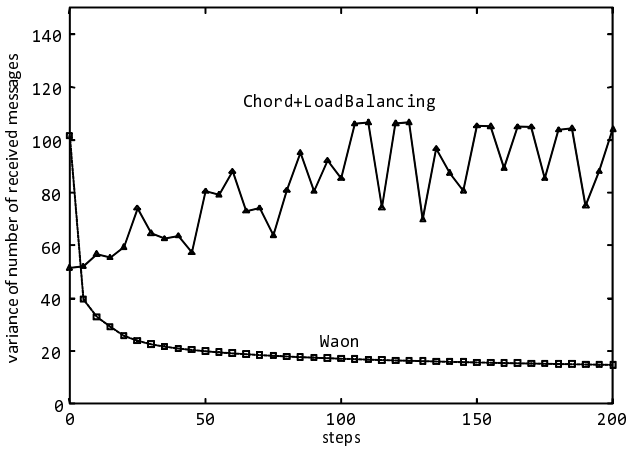


Figure 8. Variance of number of message received using a node.

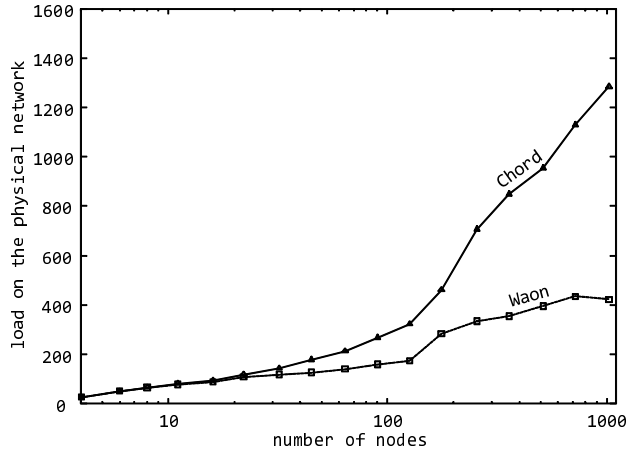


Figure 9. Load on the physical network for P2P network maintenance.

C. Load balancing of messages

Fig. 8 shows the variance of the number of messages received by each node for searching an object. In Fig. 8, *Chord+LoadBalancing* is Chord with a load balancing mechanism. Chord+LoadBalancing achieves load balancing of objects. But Chord+LoadBalancing can not achieve load balancing of messages because Chord's routing algorithm concentrates many messages on a few nodes when many nodes are located in a narrow area on the identifier ring. On the other hand, Waon uses a hop-based routing algorithm described in III-D for load balancing of messages. Therefore, the variance of Waon is less than the variance of Chord+LoadBalancing. Waon achieves not only load balancing of objects but also load balancing of messages.

D. Traffic load on the physical network

Fig. 9 shows the load of the physical network caused by maintenance packets of the overlay network. This simulation assumes that Waon puts a node on the identifier ring as a

predecessor node of another node which is the nearest on the physical network. On the other hand, Chord decides the nodes' locations on the identifier ring using a one-way hash function. In addition, this simulation assumes that $d \times m$ traffic load is caused on the physical network when a node sends m messages to a node that is d away. For example, in Fig. 6, when node *A* sends four messages to node *B*, traffic load caused on the physical network is 28.

Fig. 9 shows that the Waon's traffic load on the physical network is less than Chord's traffic load on the physical network. In both a Waon network and a Chord network, each node frequently sends maintenance packets to neighbor nodes, which are predecessor nodes and successor nodes. Therefore, the load traffic caused by Waon's maintenance packets is less than the load traffic caused by Chord's maintenance packets because neighbor nodes in a Waon network are close nodes on the physical network.

V. Conclusion

A structured P2P network is a scalable network that enables accurate object search. Because of that feature, structured P2P networks of many types have been proposed. However, it is difficult for existing structured P2P networks to achieve dynamic load balancing because existing structured P2P networks assign objects to nodes statically. In this paper, we proposed a new structured P2P network designated as the *Well-distribution Algorithm for Overlay Network: Waon*. Waon achieves not only dynamic load balancing of objects but also dynamic load balancing of messages. Each node in a Waon network modifies its location on the identifier ring for load balancing of objects. In addition, each node in a Waon network uses a hop-based routing algorithm for load balancing of messages. In this paper, we confirmed Waon's advantages over an existing structured P2P network through simulation results. Simulation results show that Waon achieves not only load balancing of objects but also load balancing of messages. Additionally, the simulation results show that Waon can reduce the load on the physical network.

Future studies will address dynamic load balancing of Waon in a heterogeneous environment. In this paper, we assume a homogeneous environment. However, a real P2P network contains both low-spec nodes and high-spec nodes; these nodes share both popular objects and unpopular objects. We will develop a mechanism of dynamic load balancing for Waon in a heterogeneous environment in the future.

Acknowledgment

This research was partly funded by a Ministry of Education, Culture, Sports, Science, and Technology Grant-in-Aid for Young Scientists, 20700069, 2008.

References

[1] S. Saroiu, K. P. Gummadi, and S. D. Gribble, "A measurement study of peer-to-peer file sharing systems," *Proc. Multimedia Computing and Networking (MMCN) 2002*, 2002.

[2] I. Clarke, S. G. Miller, T. W. Hong, O. Sandberg, and B. Wiley, "Protecting free expression online with freenet," *IEEE Internet Computing*, vol. 6, no. 1, pp. 40–49, 2002.

[3] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 17–32, 2003.

[4] J. Aspnes and G. Shah, "Skip graphs," *ACM Transactions on Algorithms*, vol. 3, no. 4, 2007.

[5] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," *Proc. ACM SIGCOMM*, pp. 161–172, 2001.

[6] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," *Proc. IFIP/ACM International Conference on Distributed Systems Platforms*, pp. 329–350, 2001.

[7] 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 Journal on Selected Areas in Communications*, vol. 22, no. 1, pp. 41–53, 2004.

[8] W. Pugh, "Skip lists: A probabilistic alternative to balanced trees," *Communications of the ACM*, vol. 33, no. 6, pp. 668–676, 1990.

[9] R. Guerraoui, S. B. Handurukande, K. Huguenin, A.-M. Kermarrec, F. L. Fessant, and E. Riviere, "Gosskip, an efficient, fault-tolerant and self organizing overlay using gossip-based construction and skip-lists principles," *Proc. Sixth IEEE International Conference on Peer-to-Peer Computing*, pp. 12–22, 2006.

[10] A. Rao, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica, "Load balancing in structured p2p systems," *Lecture Notes in Computer Science*, vol. 2735, pp. 68–79, 2003.

[11] B. G. Karthik, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica, "Load balancing in dynamic structured p2p systems," *Proc. IEEE INFOCOM 2004*, vol. 4, pp. 2253–2262, 2004.

[12] J. Ledlie and M. Seltzer, "Distributed, secure load balancing with skew, heterogeneity, and churn," *Proc. IEEE INFOCOM 2005*, vol. 2, pp. 1419–1430, 2005.

[13] A. R. Bharambe, M. Agrawal, and S. Seshan, "Mercury: Supporting scalable multi-attribute range queries," *Proc. ACM SIGCOMM*, vol. 34, pp. 353–366, 2004.