Simple Dynamic Load Balancing Mechanism for Structured P2P Network and its Evaluation

Atsushi Takeda

Department of Information Science,
Tohoku Gakuin University,
2-1-1 Tenjinzawa, Izumi-ku, Sendai 981-3193, Japan
E-mail: takeda@cs.tohoku-gakuin.ac.jp

Takuma Oide

Advanced Course of Information and Electronic System Engineering,
Sendai National College of Technology,
4-16-1 Ayashi-Chuoh, Aoba-ku, Sendai 989-3128, Japan
E-mail: a1102003@sendai-nct.jp

Akiko Takahashi

Department of Information Systems,
Sendai National College of Technology,
4-16-1 Ayashi-Chuoh, Aoba-ku, Sendai 989-3128, Japan
E-mail: akiko@sendai-nct.ac.jp

Abstract: Many proposals have been advanced for structured P2P networks, but it is difficult for existing structured P2P networks to achieve dynamic load balancing sufficiently. In this paper, we propose a new structured P2P network called *Waon*, which achieves dynamic load balancing among nodes. Waon uses a simple algorithm of load balancing and a routing table based on the number of hops between nodes to distribute a communication load uniformly. The proposed scheme enables object deployment based on object semantics. Consequently, Waon can support range queries. Furthermore, the proposed scheme enables node deployment based on node locality. Therefore, Waon can reduce the communication load on the physical network. Using simulation results, this report describes that Waon achieves dynamic load balancing sufficiently. Furthermore, simulation results in this paper demonstrate that Waon can reduce the load on the physical network.

Keywords: structured p2p network; peer-to-peer network; overlay network; dynamic load balancing.

Biographical notes: Atsushi Takeda received both a B.E degree in Electronic Engineering and a M.S degree in Information Science from Tohoku University respectively in 2000 and 2002, and a Ph.D. degree from Tohoku University in 2005. He is currently an Associate Professor at Tohoku Gakuin University. His current research interests are overlay network, p2p network, ubiquitous computing.

Takuma Oide is a student at Sendai National College of Technology. His major is Information Technology, and his current research interests are overlay networks and p2p networks.

Akiko Takahashi received both a B.E degree in Information Engineering and a M.S degree in Information Science from Tohoku University respectively in 2002 and 2004, and a Ph.D. degree from Tohoku University in 2007. She is currently an Assistant Professor at Sendai National College of Technology. Her current research interests are intelligent agent, multiagent system and ubiquitous computing.

1 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. For that reason, many P2P network applications have been developed (Saroiu et al., 2002; Clarke et al., 2002). Particularly, structured

P2P networks of many types have been proposed in recent years (Stoica et al., 2003; Aspnes and Shah, 2007; Ratnasamy et al., 2001; Rowstron and Druschel, 2001; Zhao et al., 2004). A structured P2P network is a scalable network on which nodes can be searched accurately for objects. Existing structured P2P networks, however, can not achieve load balancing sufficiently because existing schemes assign objects to nodes statically.

As described in this paper, we propose a new structured P2P network named the Well-distribution Algorithm for an Overlay Network *Waon*, which achieves dynamic load balancing of messages. Waon uses a ring-type identifier space such as Chord's identifier ring (Stoica et al., 2003). 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, node locations on the identifier ring are calculated statically. In contrast, in a Waon network, node locations on the identifier ring are modified dynamically to achieve dynamic load balancing.

In a Waon network, objects are placed on the identifier ring and are assigned to nodes. The object assignment is determined according to object locations and node locations on the identifier ring. In a Waon network, when too many objects are assigned to a node, the node modifies its locations on the identifier ring to reduce objects assigned to the node. This paper presents a description of how to modify node locations on the identifier ring in a Waon network.

Many existing routing algorithms of structured P2P networks subsume that nodes are located uniformly on the identifier space. However, in a Waon network, nodes might be located non-uniformly on the identifier ring. For that reason, Waon uses a hop-based routing algorithm, which has been presented as a routing algorithm of Chord# (Schütt et al., 2008). A hop-based routing algorithm distributes communication messages uniformly, even if the node distribution on the identifier ring is not uniform. In a Waon network, each node builds a routing table based on the number of hops between nodes. Then the load of all communications for maintaining the network and searching objects is distributed uniformly to each node.

As described above, Waon's features are the following:

1. Each node modifies its location on Waon's identifier ring dynamically to achieve uniform distribution of objects.

2. Each node builds a routing table based on the number of hops between nodes to achieve load balancing of messages.

These features give Waon the following advantages:

1. Waon is a scalable structured P2P network. Waon uses a hop-based routing algorithm, which is known as a scalable routing algorithm (Schütt et al., 2008). In a Waon network, the number of communication messages necessary for maintaining the network and searching objects is $O(\log N)$, where N denotes the number of nodes.

2. Waon achieves dynamic load balancing of communication messages. Waon uses a simple algorithm of dynamic load balancing to assign objects uniformly to each node. The algorithm enables not only uniform assignment of objects but also dynamic load balancing of messages.

3. Waon can support range queries for searching objects. Waon does not assume a 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 (Aspnes and Shah, 2007).

4. Waon can reduce traffic on the physical network. Waon does not assume a 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.

No existing structured P2P network can realize all of these advantages simultaneously, but Waon can.

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 of our conclusions.

2 Related Work

A pure P2P network is classified as an unstructured P2P network or structured P2P network. Nodes in an unstructured P2P network can connect freely to any other network node. For that reason, 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 node objects can be sought 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 (Stoica et al., 2003), CAN (Ratnasamy et al., 2001), Pastry (Rowstron and Druschel, 2001), and Tapestry (Zhao et al., 2004) 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 using consistent hashing of the node's IP address. The object location on the identifier is also determined according to the object identifier, which is calculated using consistent hashing of the object name. DHT builds an overlay network based on node locations on the identifier space, and DHT assigns objects to nodes on the basis of object 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. Therefore, the computing cost of searching an object is $O(\log n)$, where n is the number of nodes in the DHT network. Therefore, DHT algorithms are scalable. However, DHT does not support range queries because object locations on the identifier space are determined using 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 (Aspnes and Shah, 2007). SkipGraph provides efficient object search without hash functions. In a SkipGraph network, each node creates a SkipList (Pugh, 1990), 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 (Guerraoui et al., 2006). Actually, GosSkip uses gossip messages to build an overlay network based on SkipList; GosSkip also supports range queries for object searching. 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.

Because dynamic load balancing in structured P2P networks is a challenging study, many algorithms have been proposed in recent years. These dynamic load balancing are categorized into two basic approaches: *virtual server* schemes and *partition split* schemes. The use of a virtual server scheme for structured P2P networks has been proposed as a dynamic load balancing scheme (Rao et al., 2003; Karthik et al., 2004). In a structured P2P network with virtual server scheme, a node is implemented as a virtual server, and a virtual server runs on a physical node. When there is an overloaded node, the node transfers virtual servers to another physical node for load balancing. An important advantage of this scheme is that it is possible to achieve dynamic load balancing in heterogeneous environments. Nevertheless, it is difficult for a virtual server scheme to reflect the physical network on the overlay network. Therefore, this scheme can not reduce the traffic on the physical network. In addition, a virtual server scheme requires extra communication for dynamic load balancing.

The partition split scheme has also been proposed as a means of dynamic load balancing (Ledlie and Seltzer, 2005; Bienkowski et al., 2005; Kenthapadi and Manku, 2005). The scheme achieves dynamic load balancing by transferring objects from an overloaded node to an under-loaded node. Usually, the partition split scheme assumes uniform distribution of objects on the identifier space. Therefore, this scheme demands an object distribution mechanism such as consistent-hashing, which makes supporting range queries impossible. Mercury is a structured P2P network that not only achieves dynamic load balancing with the partition split scheme but supports also range queries (Bharambe et al., 2004). Mercury uses a density-based routing algorithm. Therefore, Mercury achieves load balancing of messages even if nodes are located non-uniformly on the identifier ring. Mercury, however, uses the leave–rejoin mechanism for load balancing of objects. Therefore, Mercury can not reduce the traffic on the physical network. In addition, Mercury requires extra communication to estimate the density of the nodes on the identifier ring.

Consistent hashing is the reason why DHT networks can not support range queries. Therefore, a structured P2P network without consistent hashing is proposed: Chord# (Schütt et al., 2008). Chord# uses a hop-based routing algorithm instead of ID-based routing algorithm to support range queries. Chord# can achieve dynamic load balancing by using the leave–rejoin mechanism. For that reason, it is difficult for Chord# to reduce traffic on the physical network.

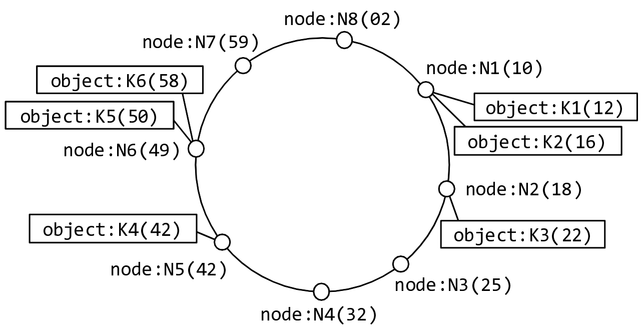
To reduce traffic of the physical network, some structured P2P networks have been developed (Zhu and Hu, 2005; Shen and Xu, 2008; Shen, 2009). These structured P2P networks build an overlay network based on the physical network, and reduce the traffic on the physical network. These studies also demonstrated that it is possible to reduce the traffic on the physical network using a landmark technique (Ratnasamy et al., 2002). However, it is not easy for these structured P2P networks to achieve dynamic load balancing.

The fundamental concept of our proposed *Waon* is the combination of a partition split scheme proposed by Mercury and a hop-based routing algorithm proposed by Chord#. However, Waon differs from existing algorithms. Waon achieves high scalability, dynamic load balancing, supporting range queries and traffic reduction on the physical network. Existing proposals have presented means to achieve some of the requirements described above, but no structured network that achieves all of these requirements has been reported.

3 Proposal: Waon

3.1 Basic concept of Waon

Waon is a scalable structured P2P network that achieves dynamic load balancing, supports range queries, and reduces the physical network traffic. Waon uses a ring-type identifier space like Chord's identifier ring (Stoica et al., 2003). Nodes and objects in a Waon network



(a) Waon's Identifier ring
(The numbers in parenthesis are locations on the 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 node properties.

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 evenly to each node.

In a Waon network, many nodes might be located in a narrow area of the identifier ring because nodes can set their location freely. DHT routing algorithms subsume a uniform distribution of nodes on the identifier ring. For that reason, Waon can not use existing DHT routing algorithms because many messages are concentrated in a few nodes if Waon uses a DHT routing algorithm. Therefore, Waon uses a hop-based routing algorithm, which has been proposed as a routing algorithm of Chord# (Schütt et al., 2008). The hop-based routing algorithm achieves load balancing of messages even if the nodes are located non-uniformly on the identifier ring.

Waon does not subsume a 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 a uniform distribution of nodes on the identifier ring. Waon can therefore reflect the physical network topology into nodes' locations on the identifier ring. Waon can reduce the physical network traffic between nodes.

3.2 Waon's identifier ring

Fig. 1(a) portrays a Waon identifier ring. The Waon identifier ring is a circle that is 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 shown below.

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

In this definition, r represents the number of neighbour 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 a clockwise direction from the node, and property *predecessor* is a list of nodes lying in an anticlockwise 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 node properties.

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

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

For example, in the circumstances depicted in Fig. 1, node *N1* has object *K1* and *K2* because these objects are located between node *N1* and *N1*'s first successor node *N2*.

3.3 Dynamic load balancing

Nodes in a Waon network change their location on the identifier ring for dynamic load balancing. Figure 2 shows pseudo-code for changing node 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. (at line: 5–7)

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 node n is greater than the average of the number of objects assigned to the predecessor nodes. (at line: 9–10)

```
01: // update location of a node
02: n.updateLocation() {
03:   sum = count(n.object);
04:
05:   for (i = 0; i < r; i = i + 1) {
06:     sum = sum + count(n.predecessor[i].object);
07:   }
08:
09:   ave = sum / (r + 1);
10:   num = count(n.object) - ave;
11:
12:   if (num > 0) {
13:     object = sortById(n.object);
14:     n.location = object[num].id;
15:
16:     for (i = 0; i < num; i = i + 1) {
17:       n.predecessor[0].addObject(object[i]);
18:       n.removeObject(object[i]);
19:     }
20:   }
21: }
```

Figure 2 Pseudo-code for modifying node location.

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. Subsequently, node n transfers some objects to its first predecessor node. (at line: 12–20)

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

3.4 Hop-based 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 the distribution of node locations on the identifier ring is non-uniform. 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 a node property *route*. Each node creates a routing table under the following rule:

$$\text{node.route}[i+1] = \text{node.route}[i].\text{route}[i]$$

This rule means that node *route*[i] is the $2^i th$ successor node. Figure 3 portrays 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 independent of the distribution

```
// update routing-table
n.updateRoute() {
  node = n.successor[0];
  c_distance = distance(n, node);
  p_distance = 0;

  for (i = 0; p_distance < c_distance; i = i + 1) {
    n.route[i] = node;
    node = node.route[i];
    p_distance = c_distance;
    c_distance = distance(n, node);
  }
}
```

Figure 3 Pseudo-code for updating a routing table.

```
// search an object named k
n.searchObject(k) {
  target = distance(n, k)

  if (target < distance(n, n.successor[0])) {
    foreach (o : n.object) {
      if (k = o.name) {
        return o;
      }
    }

    return null;
  }
  else {
    next = n.successor[0];

    for (i = 0; i < length(n.route); i = i + 1) {
      if target < distance(n, n.route[i])) {
        break;
      }
      else {
        next = n.route[i];
      }
    }

    return next.searchObject(k);
  }
}
```

Figure 4 Pseudo-code for searching an object.

of nodes' locations on the identifier ring. Therefore, Waon achieves efficient object searching even if the node locations on the identifier ring are distributed non-uniformly.

Fig. 4 portrays pseudo-code for searching for an object. Waon's search process is fundamentally identical

```
// join to a network
// m and n are physically nearby nodes
n.join(m) {
  n.location =
    (m.location + m.successor[0].location) / 2;

  n.predecessor[0] = m;
  n.successor[0] = m.successor[0];
  n.predecessor[0].addSuccessor(n);
  n.successor[0].addPredecessor(n);

  foreach (o : m.object) {
    if (distance(m,n) < distance(m,o)) {
      m.removeObject(o);
      n.addObject(o);
    }
  }

  n.updateRoute();
}
```

Figure 5 Pseudo-code for joining to a network.

to Chord's search process. Therefore, the cost of an object search is $O(\log n)$, where n denotes the number of nodes in the Waon network. Therefore, Waon is a scalable structured P2P network.

3.5 Join process

To reduce the load of a physical network, Waon builds a proximity-aware overlay network, which has been studied in recent years (Zhu and Hu, 2005; Shen and Xu, 2008; Shen, 2009). A node often communicates with neighbour nodes on the identifier ring. Therefore, if the physical distance between a node and neighbour nodes is short, then the load of the physical network is small.

Fig. 5 portrays pseudo-code for joining to a Waon network. Before the joining process, a node finds a nearby node on the physical network using proximity-aware technique such as GPS or a landmark technique (Ratnasamy et al., 2002). After finding the nearby node, the node joins as a successor node of the nearby node. Subsequently the node receives objects located in the responsibility region of the node and performs *updateRoute* to build a routing table. The cost of the joining process is $O(\log n)$, where n is the number of nodes in the Waon network.

3.6 Leave process

Fig. 6 portrays pseudo-code for leaving a Waon network. At the leaving process, no procedure is necessary except transferring objects. After this process, routing tables of other nodes are rebuilt through an updating process *updateRoute*, which is performed by the other nodes at regular time intervals.

```
// leave a network
n.leave() {
  n.predecessor[0].removeSuccessor(n);
  n.successor[0].removePredecessor(n);

  foreach (o : n.object) {
    m.addObject(o);
  }
}
```

Figure 6 Pseudo-code for leaving a network.

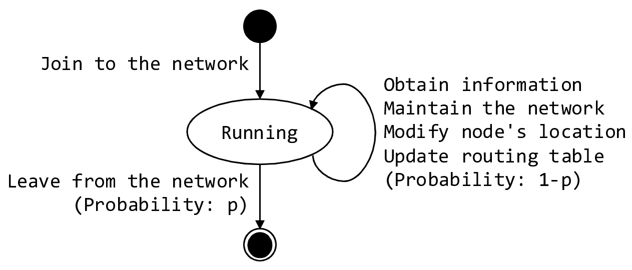


Figure 7 State chart of nodes in the simulation.

4 Evaluation

4.1 Simulation environment

To confirm Waon's advantages over an existing scheme, we evaluated Waon's performance through a simulation implemented with Java. In this simulation, nodes perform actions independently and mutually communicate at every step. Packet loss and latency are not simulated in this simulation.

Fig. 7 portrays a state chart of nodes in this simulation. In Fig .7, p denotes the probability of logout, which is a churn parameter. After a node joins a Waon network, nodes perform the following actions at every step until the node leaves the Waon network.

1. All nodes send *ping* messages to all predecessor nodes and to all successor nodes to obtain information of the nodes and maintain the network.
2. As described in 3.3, all nodes modify their respective locations on the identifier ring for dynamic load balancing.
3. As described in 3.4, all nodes update their routing tables.

In addition, for measurement of communication cost at searching process, each node searches all objects at every five steps. When a node leaves a Waon network, a new node joins the Waon network, so the number of nodes in the Waon network remains unchanged.

Fig. 8 portrays some objects used in the simulation. This simulation assumes that shared objects in the

```
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 8 Objects used for the simulation.

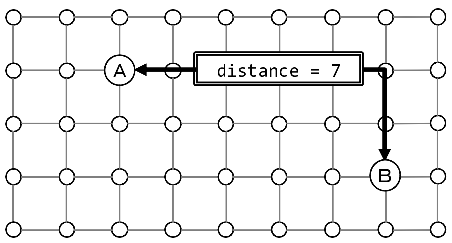


Figure 9 Physical network assumed in the simulation.

P2P network are files in a UNIX operating system. Consequently, in this simulation, nodes share some 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 node transfers an object to another node. This simulation assumes that shared objects are pointers such as URLs. The shared objects are put on Waon's identifier ring based on the file paths. Therefore, file objects contained in the same directory are located nearby on the identifier ring in this simulation.

Fig. 9 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 use the following parameters in this simulation:

- Parameter m, which is the size of Waon's identifier ring, is 64. (the maximum of the identifier is 2^{64})
- Parameter r, which is the size of neighbour node list: 7.

We performed the simulation described above. This section shows Waon's advantages over Chord (Stoica et al., 2003), a famous structured P2P network, using simulation results.

4.2 Dynamic load balancing

Fig. 10 shows the variance of the number of objects assigned to each node, where the number of nodes is 1024 and the number of objects is 10240. 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

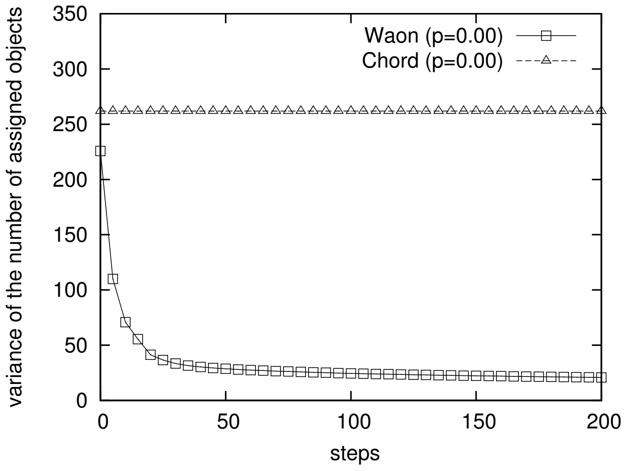


Figure 10 Variance of the number of objects assigned to a node.

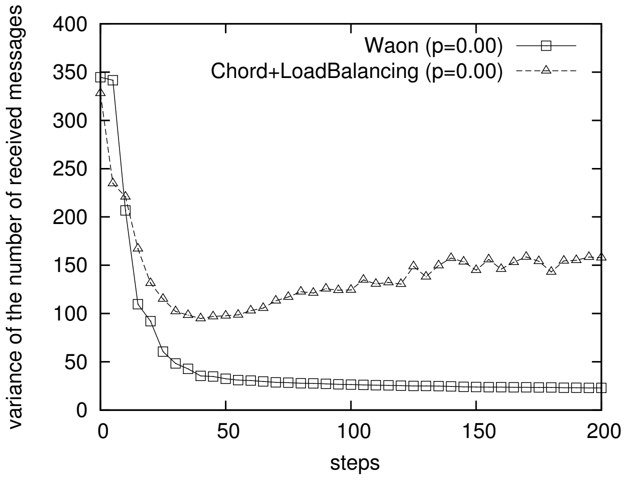


Figure 11 Variance of number of message received by a node.

change the nodes' location on the identifier ring to control the nodes' load as described in 3.3. Consequently, Waon achieves a uniform distribution of objects.

Fig. 11 shows the variance of the number of communication messages received by each node at each step, where the number of nodes is 1024. In Fig. 11, *Chord+LoadBalancing* is Chord with a load balancing mechanism. Chord+LoadBalancing achieves load balancing of objects. However, 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. However, Waon uses a hop-based routing algorithm described in 3.4 for load balancing of messages. Therefore, the variance of Waon is less than the variance of Chord+LoadBalancing. Waon achieves not only uniform distribution of objects but also load balancing of messages.

4.3 Load balancing in churn environment

Fig. 12 shows the variance of the number of objects assigned to each node, where the number of nodes is 1024

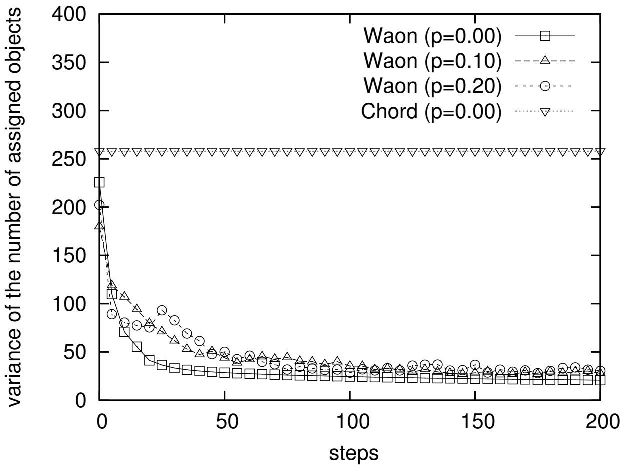


Figure 12 Variance of the number of objects assigned to a node.

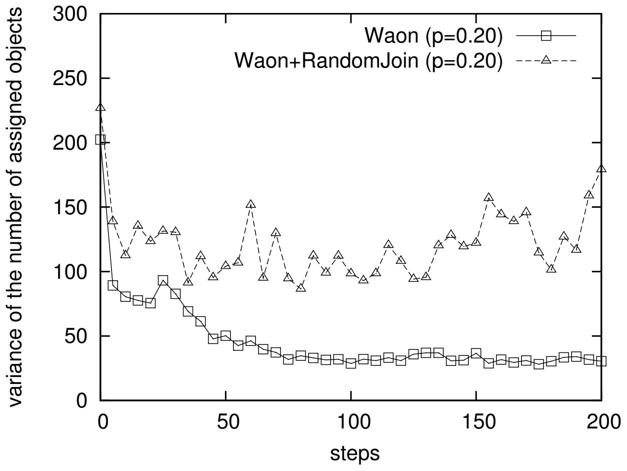


Figure 14 Variance of the number of messages received by a node.

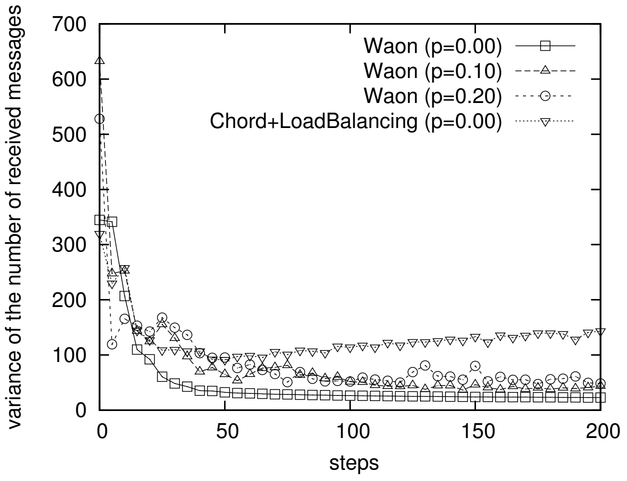


Figure 13 Variance of the number of messages received by a node.

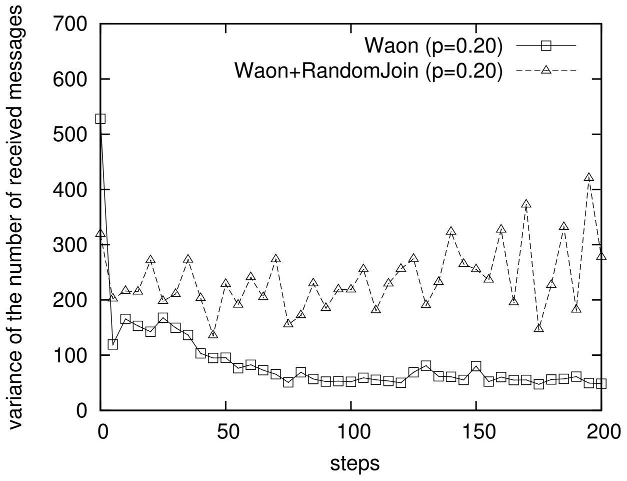


Figure 15 Variance of the number of message received by a node.

and the number of objects is 10240. Additionally, Fig. 13 presents the variance of the number of communication messages received by each node at each step, where the number of nodes is 1024. Fig. 12 shows that the variance of the number of objects assigned to each node increases with the churn parameter, and Fig. 13 also shows that the variance of the number of messages received by each node increases with the churn parameter. These results indicate that Waon's dynamic load balancing in a churn environment is less effective than in a static environment. These results, however, also indicate that Waon's dynamic load balancing is functional in a churn environment.

Fig. 14 portrays the variance of the number of objects assigned to each node, where the number of nodes is 1024 and the number of objects is 10240. Furthermore, Fig. 15 shows the variance of the number of communication messages received by each node at each step, where the number of nodes is 1024. In these figures, *Waon+RandomJoin* is Waon without the join process described in section 3.5. Nodes in *Waon+RandomJoin* are located randomly on the identifier ring at the join process. These results show that Waon is better than *Waon+RandomJoin* in a churn environment. Therefore, Waon's joining process makes dynamic load balancing more effective in a churn environment.

4.4 Path length of object search

Fig. 16 shows the average of path length of seeking an object on the overlay network. In Fig. 16, *Chord+LoadBalancing* is Chord with a load balancing mechanism based on the *partition split* concept, which uses an id-based routing algorithm. In contrast, Waon uses a hop-based routing algorithm. Fig. 16 shows that the number of messages required for Waon's object search is less than the number of Chord+LoadBalancing messages. This result demonstrates that a new routing algorithm described in 3.4 enables more efficient object searching than the existing DHT algorithm.

4.5 Traffic load on the physical network

Fig. 17 and Fig. 18 show the load of the physical network caused by maintenance packets of the overlay network. This simulation assumes that Waon puts a node on

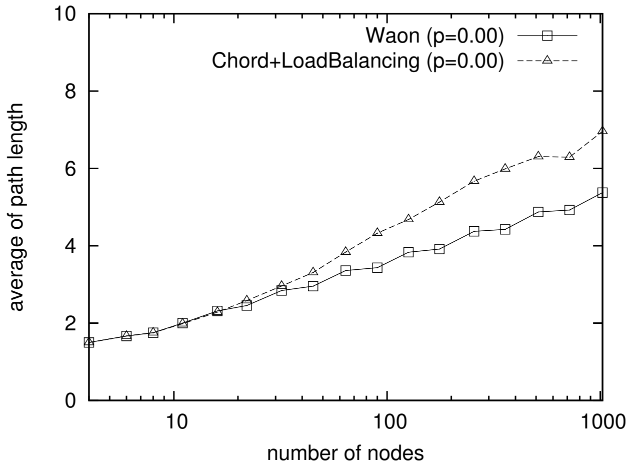


Figure 16 Average path length of object search.

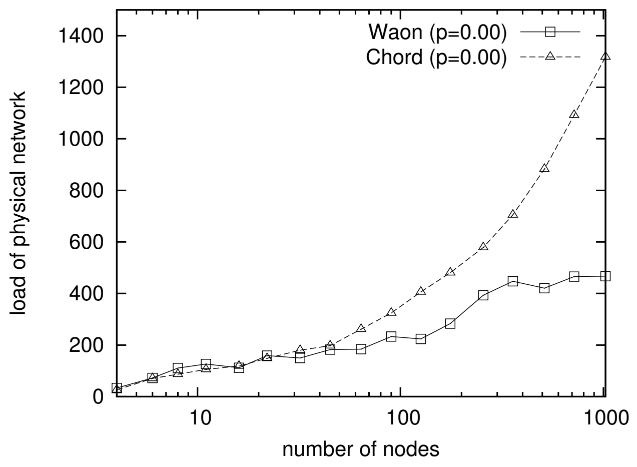


Figure 17 Load on the physical network for maintaining an overlay network.

the identifier ring as a successor node of another node that is the nearest on the physical network. In contrast, Chord determines the node locations on the identifier ring using a one-way hash function. In addition, this simulation assumes that $d \times m$ traffic load is emplaced on the physical network when a node sends m messages to a node that is d away. For example, in Fig. 9, when node A sends four messages to node B, the traffic load placed on the physical network is 28.

Fig. 17 portrays the load on the physical network caused by maintaining the overlay network. Fig. 17 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 neighbour nodes, which are predecessor nodes and successor nodes. Therefore, the load traffic caused by Waon's maintenance packets is less than the load traffic attributable to Chord's maintenance packets because neighbour nodes in a Waon network are close nodes on the physical network. When the number of nodes is 1024, the load of the physical network caused by Waon is

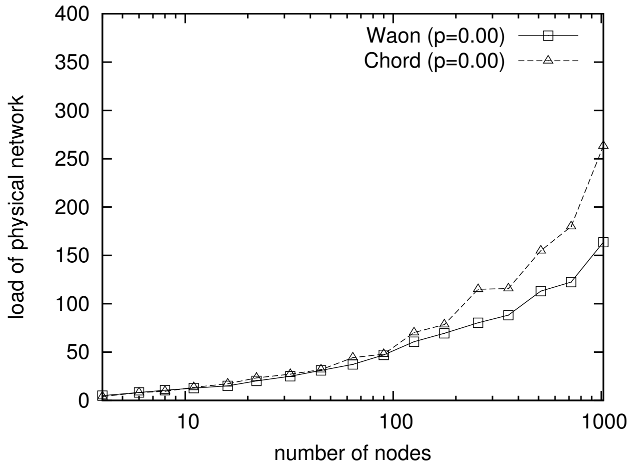


Figure 18 Load on the physical network for searching an object.

reduced by more than 50% compared with the load on the physical network caused by Chord.

Fig. 18 shows the load on the physical network caused by searching an object. Fig. 18 shows that the Waon's traffic load on the physical network is less than Chord's traffic load on the physical network. When the number of nodes is 1024, the load of the physical network caused by Waon is reduced about 40% compared with the load on the physical network of Chord.

5 Conclusion

A structured P2P network is a scalable network that enables accurate object search. In pursuit of the development of those features, 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 uniform distribution of objects but also dynamic load balancing of messages. Each node in a Waon network modifies its location on the identifier ring for uniform distribution of objects dynamically. In addition, each node in a Waon network uses a hop-based routing algorithm for load balancing of messages. We confirmed Waon's advantages over an existing structured P2P network through simulation results, which show that Waon achieves not only uniform distribution 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 comparing Waon with existing algorithms for dynamic load balancing. We expect that Waon is less efficient than some of existing algorithms. However, we believe that Waon has advantages over these algorithms in traffic reduction. Now, we are implementing other algorithms on our simulator. We will report the comparison between

Waon and these algorithms in the future. Additionally, dynamic load balancing in a heterogeneous environment is also a future work. We assumed a homogeneous environment for this study. However, a real P2P network includes both low-spec nodes and high-spec nodes; these nodes share both popular objects and unpopular objects. We are developing a mechanism of dynamic load balancing for use with Waon in a heterogeneous environment in future studies.

Acknowledgment

This work was supported by a Grant-in-Aid for Challenging Exploratory Research (No.23650032) of the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

Aspnes, J. and Shah, G. (2007) 'Skip graphs', *ACM Transactions on Algorithms*, Vol. 3, No. 4.

Bharambe, A. R., Agrawal, M. and Seshan, S. (2004) 'Mercury: Supporting scalable multi-attribute range queries', *Proc. ACM SIGCOMM*, Vol. 34, pp.353–366.

Bienkowski, M., Korzeniowski, M. and auf Heide, F. M. (2005) 'Dynamic load balancing in distributed hash tables', *Lecture Notes in Computer Science*, Vol. 3640, pp.217–225.

Clarke, I., Miller, S. G., Hong, T. W., Sandberg, O. and Wiley, B. (2002) 'Protecting free expression online with freenet', *IEEE Internet Computing*, Vol. 6, No. 1, pp.40–49.

Guerraoui, R., Handurukande, S. B., Huguenin, K., Kermarrec, A.-M., Fessant, F. L. and Riviere, E. (2006) '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.

Karthik, B. G., Lakshminarayanan, K., Surana, S., Karp, R. and Stoica, I. (2004) 'Load balancing in dynamic structured p2p systems', *Proceedings of INFOCOM 2004*, Vol. 4, pp.2253–2262.

Kenthapadi, K. and Manku, G. S. (2005) 'Decentralized algorithms using both local and random probes for p2p load balancing', *Proc. 17th Annual ACM Symposium on Parallelism in Algorithms and Architectures*, pp.135–144.

Ledlie, J. and Seltzer, M. (2005) 'Distributed, secure load balancing with skew, heterogeneity, and churn', *Proc. IEEE INFOCOM 2005*, Vol. 2, pp.1419–1430.

Pugh, W. (1990) 'Skip lists: A probabilistic alternative to balanced trees', *Communications of the ACM*, Vol. 33, No. 6, pp.668–676.

Rao, A., Lakshminarayanan, K., Surana, S., Karp, R. and Stoica, I. (2003) 'Load balancing in structured p2p systems', *Lecture Notes in Computer Science*, Vol. 2735, pp.68–79.

Ratnasamy, S., Francis, P., Handley, M., Karp, R. and Shenker, S. (2001) 'A scalable content-addressable network', *Proc. ACM SIGCOMM*, pp.161–172.

Ratnasamy, S., Handley, M., Karp, R. and Shenker, S. (2002) 'Topologically-aware overlay construction and server selection', *Proceedings of INFOCOM 2002*.

Rowstron, A. and Druschel, P. (2001) '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.

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

Schütt, T., Schintke, F. and Reinefeld, A. (2008) 'Range queries on structured overlay networks', *Computer Communications*, Vol. 31, No. 2, pp.280–291.

Shen, H. (2009) 'Efficient and effective file replication in structured p2p file sharing systems', *Proceedings of P2P '09*, pp.159–162.

Shen, H. and Xu, C.-Z. (2008) 'Hash-based proximity clustering for efficient load balancing in heterogeneous dht networks', *Journal of Parallel and Distributed Computing*, Vol. 68, No. 5, pp.686–702.

Stoica, I., Morris, R., Liben-Nowell, D., Karger, D. R., Kaashoek, M. F., Dabek, F. and Balakrishnan, H. (2003) 'Chord: A scalable peer-to-peer lookup protocol for internet applications', *IEEE/ACM Transactions on Networking*, Vol. 11, No. 1, pp.17–32.

Zhao, B. Y., Huang, L., Stribling, J., Rhea, S. C., Joseph, A. D. and Kubiatowicz, J. D. (2004) 'Tapestry: A resilient global-scale overlay for service deployment', *IEEE Journal on Selected Areas in Communications*, Vol. 22, No. 1, pp.41–53.

Zhu, Y. and Hu, Y. (2005) 'Efficient, proximity-aware load balancing for dht-based p2p systems', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 16, No. 4, pp.349–361.