Peer-to-Peer Netw. Appl. (2014) 7:497–510
DOI 10.1007/s12083-013-0229-z

A scalable P2P overlay based on arrangement graph with minimized overhead

Ssu-Hsuan Lu · Kuan-Ching Li · Kuan-Chou Lai · Yeh-Ching Chung

Received: 30 December 2011 / Accepted: 27 August 2013 / Published online: 18 September 2013

Abstract With innovations in the Internet, it is becoming increasingly relied upon. In the last decade, research on peer-to-peer (P2P) technology has become even more popular. As more people use P2P systems, the scalability and flexibility of the systems must be considered. In this study, an arrangement graph is used to form a P2P overlay, the Arrangement-Graph Overlay (AGO), to reduce system overhead and bind routing hops. The proposed AGO utilizes the properties of the arrangement graph, i.e., that each node has a unique ID and IDs between adjacent nodes differ by only one digit, to form the overlay network and develop a routing algorithm. The routing hops of the proposed AGO system can be bound within a certain number because of the diameter of the arrangement graph. Experimental results show that the proposed AGO system can greatly reduce system overhead and perform routing in a constant number of hops, even in a large-scale network environment. The experimental results also show that the AGO system consumes less bandwidth, which is an important consideration in P2P systems.

Keywords Peer-to-peer · Overlay network · Arrangement graph · Large scale

S.-H. Lu (✉) · Y.-C. Chung
Department of Computer Science,
National Tsing Hua University,
Hsinchu, Taiwan
e-mail: shlu@sslab.cs.nthu.edu.tw

Y.-C. Chung
e-mail: ychung@cs.nthu.edu.tw

K.-C. Li
Department of Computer Science and Information Engineering, Providence University,
Taichung, Taiwan
e-mail: kuancli@pu.edu.tw

K.-C. Lai
Department of Computer Science, National Taichung University,
Taichung, Taiwan
e-mail: kclai@ntcu.edu.tw

1 Introduction

The importance of the Internet has increased dramatically in the last 10 years because of its convenience in accessing both professional services and entertainment. People do many important activities that depend on the Internet and enjoy the convenience of the Internet. In such an environment, peer-to-peer (P2P) systems, in which any two nodes can directly share information without intermediate servers, are widely used and discussed. In P2P systems, each node can connect with other nodes by joining the system, without involving other physical connections. Each node in a P2P system acts as both server and client, so the bottleneck of a client-server architecture is eliminated. P2P systems have greatly increased the flexibility of networks, and resources can be easily, quickly, and conveniently accessed and shared through P2P systems.

P2P systems implement abstract overlay networks established on the application layer and on top of physical network topology. The overlay network is a type of virtual network that ignores the physical architecture of the network, treats each node on the network as a single node, and assumes that each node can connect with some other nodes in a specific way. In such a virtual environment, overlay networks do not consider the physical connection between nodes, but when nodes join or leave the system, it affects the stability of such a system. Thus, P2P systems must be more flexible in response to dynamic environments.

In P2P systems, the structure of overlay networks can be classified as *centralized* or *decentralized* [15]. In the *centralized overlay network*, there is a centralized server that manages files, such as Napster [15]. The other type is the *decentralized overlay network* that does not use a server to manage files. The *decentralized overlay network* can be *structured* or *unstructured* [15, 20] according to the topology of the overlay network. The *structured overlay network* tightly controls the locations of contents/files, locating contents/files at specific positions. It typically provides the mapping between contents/files and positions in the form of a distributed hashing table (DHT). Using the DHT, it is easy to locate contents/files because the DHT is designed to enable the system to quickly insert, query, and delete. Each node is assigned a key value for distributing contents/files. Chord [15, 21, 22] and Pastry [19] are some well-known examples of structured overlay networks.

The other type of *decentralized overlay network* is the *unstructured overlay network* [10–12]. In the *unstructured overlay network*, there is no relationship between contents/files and positions, so it cannot guarantee search results. Each node maintains a number of links with neighboring nodes to form the overlay network. Because each node only knows its neighboring nodes, the unstructured overlay network often applies the flooding approach for exhaustive searches.

Recently, several structured overlay networks have become available for P2P systems [8, 10–12, 15, 20–22]. However, these overlays continue to have some problems. Some of the networks are not suitable for large-scale environments, and others use large number of messages for maintaining neighbors and routing. This study proposes a new overlay network that resolves these problems, minimizes overhead, and maintains routing efficiency.

The structured overlay network proposed in this study involves an arrangement graph [3, 5–7, 9] and thus called the Arrangement Graph-based Overlay (AGO). The AGO inherits and employs characteristics of the arrangement graph into the P2P environment to achieve efficient routing and minimize system overhead. The arrangement graph is a generalized form of the star graph. It is denoted by $A_{n,k}$, where k denotes the number of digits of the node ID, n denotes the range of each digit of the ID, and $1 \leq k \leq n-1$. The number of nodes in the (n, k)-arrangement graph, $A_{n,k}$, is $\frac{n!}{(n-k)!}$, the degree is $k(n–k)$, and the diameter is $\left\lfloor \frac{3k}{2} \right\rfloor$.

The AGO uses the characteristics of the arrangement graph over the IDs of the nodes. Each node in the AGO system has an ID for identification, and such ID of any node in the AGO system differs by only one digit from any adjacent node. Based on the rule of the arrangement graph, the maximum number of neighbors for each node is limited, i.e., it is equal to the degree of the graph, which is $k(n–k)$.

In the proposed AGO system, a fixed number of node IDs are kept in the bootstrap node to provide to the nodes that want to join the AGO system. As a new node joins the AGO system, it asks for an ID of an existing node from the bootstrap node. Then, the new node sends a request to that existing node asking for an ID. The new node asks for an ID from the existing node received from the bootstrap to become its neighbor. When nodes want to leave the AGO system, they need to send messages to their neighbors to ask their neighbors to remove them from their neighbor tables.

In addition, a routing mechanism is also developed for searching, utilizing the properties of the arrangement graph. As a node tries to search for a specific file, the node needs to hash the filename and obtain the destination ID of the node containing that file. After that, the node that wants to search for a file sends requests to its neighbors with more than $\left\lfloor \frac{k}{2} \right\rfloor$ digits that are the same as the destination ID. Thus, the node can find the file by comparing node IDs. Experimental results show that the AGO can achieve performance improvements, particularly by reducing the number of generated messages.

This study proposes a new P2P overlay based on the arrangement graph and utilizes properties of the arrangement graph to route more efficiently. The AGO system also requires a lower system overhead than other P2P systems do, particularly in the large-scale environment. The proposed AGO system can be applied to cloud computing or home health care [16] because of its scalability and flexibility. Each node in the cloud system can be assigned an ID through the AGO system, and the routing algorithm of the AGO system can be used to obtain resources on other nodes in the cloud system. In everyday life, the proposed AGO system can be deployed in the home health care system. The AGO system can connect medical institutions, allowing medical resources to be shared and searched through the AGO system even though the resources are distributed in each medical institution. In addition, doctors can obtain medical records of patients through the AGO system to treat patients more efficiently.

The remainder of this paper is organized as follows. Section 2 presents some related work. Section 3 describes the proposed Arrangement Graph-based Overlay, and Section 4 shows some experimental results. Finally, conclusions and future work are presented in Section 5.

2 Related work

This section describes related research, such as the classification of P2P systems and the arrangement graph.

The structure of overlay networks can be classified into *centralized* and *decentralized* types according to their operations management. In the centralized overlay network, the most important property is that it offers a server to manage files in the overlay, such as Napster [15]. Napster uses a centralized overlay network for sharing mp3 files. As one user performs a search for mp3 files, a request is forwarded to the server and the user receives a list of nodes with needed files. Then, the user sends a request to those nodes for accessing the desired mp3 files.

There is no server to manage files in a decentralized overlay network. The decentralized overlay network can be classified as *structured* or *unstructured* depending on the connecting topology [15]. In the structured overlay network, each resource has a key value based on the DHT and is assigned to a certain corresponding node using the hashing function. In DHTs, nodes are the hash buckets. Files can be assigned to certain responsible nodes by using the hash function. Each node is expected to be responsible for a number of files to achieve load balance. Nodes can join and leave the system, but the links must be reconstructed. In this manner, the system is more scalable than the centralized overlay network.

The other type is the unstructured P2P system, in which each node maintains a number of links with neighboring nodes to form the overlay network. Because each node only knows its neighboring nodes, the unstructured overlay network often applies the flooding approach for exhaustive searches. In this manner, queries must be flooded through the entire overlay network, producing many redundant messages and resulting in heavy network traffic and load.

Chord [15, 21, 22] uses a consistent hashing function to assign each node a key value. It adopts the hashing function to hash the IP address of each node to obtain a node ID. Chord also adopts the hashing function to obtain key values of files. Using a consistent hashing function, node IDs and keys are assigned *m*-bit identifiers. The identifier is a circle that can have at most 2^m nodes. Each node has a predecessor that is the previous node on the identifier circle and a successor that is the next node on the identifier circle. This configuration can balance the load on the system because there is roughly the same number of keys on each node. To speed up the file searching process, each node maintains a fingertable with the information of its related successors. When a node searches files, it routes requests by comparing the key values of files with successors in the fingertable.

Pastry [19] is a P2P overlay that is similar to Chord and uses two different hashing functions to convert IP addresses and file names into *128*-bit node IDs and object IDs. It dynamically adds/removes nodes through a bootstrap node and establishes a routing table. Because the properties are redundant and distributed, each node can leave immediately without notifying other nodes, rarely losing data. The key space in Pastry's DHT is circular, based on 2^b, similar to the key space of Chord, which is an unsigned *128*-bit ID. The IDs of the nodes are chosen randomly so that adjacent physical nodes are likely to be located at different positions in the Pastry. Each node needs to maintain IP addresses for nodes in its leaf set (*L*), which contains *L*/2 of the numerically closest larger node IDs and *L*/2 of the numerically closest smaller node IDs.

Kademlia [17] is a key-based routing (KBR) protocol developed for decentralized P2P networks that also uses the DHT. Kademlia has three parameters: *alpha*, *B*, and *k*. *Alpha* is a small number representing the degree of parallelism in network calls, *B* is the size in bits of the keys used to identify nodes and store and retrieve data, and *k* is the maximum number of contacts stored in a bucket. Nodes in Kademlia communicate using UDP protocol and use node IDs for identification. It has a simple structure and the XOR metric and is deployed with some well-known applications, such as eMule and several BitTorrent clients, using its features as a DHT.

R/Kademlia [8] uses recursive overlay routing instead of the iterative key lookup used in Kademlia. It can thus achieve low latency and consume low bandwidth. R/Kademlia is more efficient than Kademlia because it utilizes Proximity Routing (PR) and Proximity Neighbor Selection (PNS).

The arrangement graph [3, 5–7, 9] is denoted by $A_{n,k}$, where k is the number of digits in the node IDs, n is the range of each digit, and $1 \leq k \leq n-1$. According to the definition of the arrangement graph, the sets $\langle n \rangle$ and $\langle k \rangle$ are $\langle n \rangle = \{1, 2, \ldots, n\}$ and $\langle k \rangle = \{1, 2, \ldots, k\}$, respectively. Let P_k^n be the set of permutations of k elements taken from $\langle n \rangle$.

Definition 1 The $(n,\ k)$-arrangement graph, $A_{n,k}$, is an undirected graph (V, E), where $V = \{p_1 p_2 \ldots p_k | p_i \in \langle n \rangle$ and $p_i \neq p_j$ for $i \neq j\} = P_k^n$, and $E = \{(p, q) | p, q \in V$, and for some i in $\langle k \rangle\, p_i \neq q_i$ and $p_j = q_j$ for $j \neq i\}$.

Basic properties The $(n,\ k)$-arrangement graph is represented as $A_{n,k}$, where the number of nodes is $\frac{n!}{(n-k)!}$, the degree is $k(n–k)$, and the diameter is $\left\lfloor \frac{3k}{2} \right\rfloor$.

$G = (V, E)$ is the set of nodes, such as participating nodes, and edges, such as overlay links. Each node of $A_{n,k}$ is an arrangement with k digits out of n elements of $\langle n \rangle$, and the edges connect nodes with only one different digit. A (*4, 2*)-arrangement graph is shown in Fig. 1. The degree of each node is four, so each node has four neighbors. The node 34 is connected to the nodes 31, 32, 14, and 24. For node 34, there is only one digit different from the four nodes 31, 32, 14, and 24. In other words, all nodes that have one different digit are connected.

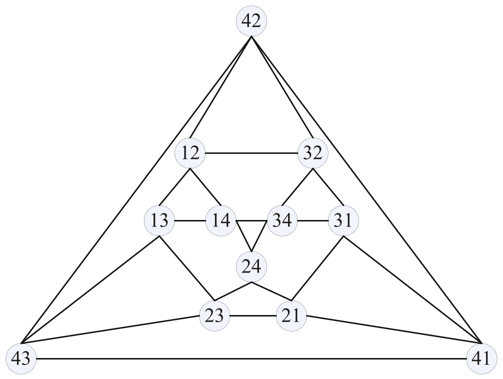


Fig. 1 (*4*, *2*)-arrangement graph

Some studies [4, 13, 14, 23] have adopted the arrangement graph to improve the efficiency of multimedia streaming, such as the topology-aware hierarchical arrangement graph (THAG) [23]. THAG improves the application-layer multicast (ALM) service. The arrangement graph is used to construct node-disjoint multicast trees for each description.

3 System structure

In this study, the arrangement graph is applied to form a structured overlay system, the AGO. The proposed AGO is a P2P overlay network with minimized overhead and efficient routing. The architecture and some algorithms illustrating the mechanics of the AGO are described in this section.

3.1 System architecture

The proposed AGO system is an abstract overlay network based on the physical network. Because the AGO system is formed according to the rules of the arrangement graph, the AGO system inherits characteristics from the arrangement graph, such as, the IDs of any adjacent nodes differ by only one digit and the degree of each node is $k(n–k)$. The AGO system consists of three main processes, i.e., joining the system, departing the system, and routing algorithm.

When a new node wants to join the AGO system, it needs to process the joining step to join the AGO system. However, all nodes in the AGO system must process the departing step to inform all neighbors before leaving the AGO system. The routing algorithm allows nodes to route for resources on other nodes in the AGO system. Properties of the arrangement graph are exploited to improve the efficiency of the routing algorithm.

Because of the development of the Internet, resources distributed on different nodes can exist in large-scale network environments around the world. In such an environment, it is necessary to develop an efficient algorithm to locate resources. The ID single-digit difference property in the arrangement graph is used to achieve efficient routing. Furthermore, because the diameter of the arrangement graph is $\left\lfloor \frac{3k}{2} \right\rfloor$, the number of routing hops can be limited.

3.2 AGO algorithms

In the proposed AGO system, a bootstrap node is used as the portal node, similar to other P2P systems. Several available IDs are stored in the bootstrap node. There are three main processes in the AGO system: joining the system, departing the system, and the routing algorithm. Joining and departing are used to build and maintain the proposed AGO system. Routing is the algorithm used for communication on the AGO system and is used to verify the efficiency of the AGO system during the routing process. Following are descriptions of the AGO system based on the three processes.

3.2.1 Joining the system

The proposed AGO system utilizes a bootstrap node as a portal, and there is a *waiting join pool* in the bootstrap. This pool keeps several node IDs that already exist in the AGO system. Each node enters the AGO system through the bootstrap and keeps some information, such as its ID and a neighbor table with its neighbors' IDs.

Joining refers to the actions that a new node performs when it enters the AGO system. When a new node enters the AGO system, it obtains an ID of an existing node from the bootstrap node and acquires an ID from that existing node. The new node also can discover information about its neighbors through this existing node.

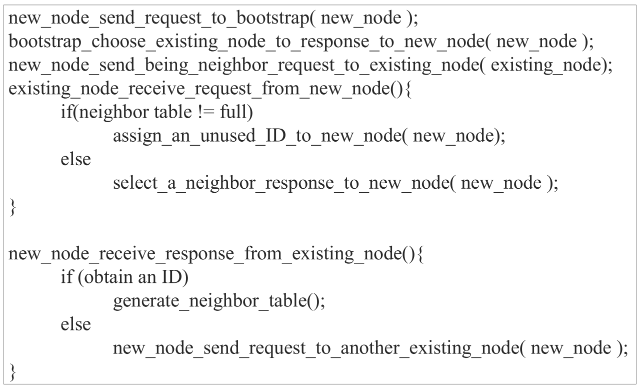
After the new node obtains the ID of the existing node from the bootstrap node, the new node sends a request to that existing node and asks for an ID. The existing node receiving the request checks whether its neighbor table is full and responds in one of two different ways. If its neighbor table is not full, it chooses an unused ID and assigns the ID to the new node. However, if the neighbor table is already full, the existing node chooses one of its neighbors' IDs and sends it back to the new node, which then sends an ID request to that neighbor.

After the new node receives the response, it also has two different response behaviors depending on the returning message. One response is to obtain the ID and become a neighbor of the responding node. The other option is to send a request to the node received from the responding node and ask for an ID again. After the new node obtains its ID, it generates its neighbor table with its neighbors' IDs according to its ID, which is received from the responding node. The new node then explores whether its neighbors already exist by sending a request to the responding node. After the responding node receives the request, it checks its neighbor table and helps the new node to generate its neighbor table. If the responding node knows any of the neighbors, it sends their information to the new node. In addition, the responding node sends the request to its neighbors to help explore the neighbors.

Figure 2 shows the pseudo-code of joining the system, and Fig. 3 describes the processes of joining the system. The following explains the processes when a new peer attempts to join the AGO system with $A_{4,3}$. According to the property of the arrangement graph, each node has at most three neighbors, which is calculated as $k(n-k) = 3*(4-3) = 3$.

1. A new node joins the system and sends a request to the bootstrap node to ask for an existing node ID.
2. The bootstrap node randomly chooses node 123 of the existing node from the waiting join pool and sends it to the new node.
3. The new node sends a request to the existing node 123 to be its neighbor.
4. The existing node 123 finds that its neighbor table is full and sends neighbor node 124 to the new node.
5. The new node sends a request to node 124 to be its neighbor.
6. Node 124 finds its neighbor table is not full and assigns the unused node ID 134 from its neighbor table to the new node. The new node becomes one of this node's neighbors.
7. The new node generates its neighbor table according to its node ID, 134, and records information of the existing node 124, who assigned it the node ID.

After the new node obtains its node ID and generates its neighbor table, it begins to establish whether its other neighbors already exist. The new node 134 sends a request to node 124 about finding the new node's neighbors. When node 124 receives the request, it follows the rules of the routing algorithm to help the new node 134 discover neighbors. The routing algorithm will be described in Section 3.2.3.



```
new_node_send_request_to_bootstrap( new_node );
bootstrap_choose_existing_node_to_response_to_new_node( new_node );
new_node_send_being_neighbor_request_to_existing_node( existing_node);
existing_node_receive_request_from_new_node(){
        if(neighbor table != full)
                assign_an_unused_ID_to_new_node( new_node);
        else
                select_a_neighbor_response_to_new_node( new_node );
}

new_node_receive_response_from_existing_node(){
        if (obtain an ID)
                generate_neighbor_table();
        else
                new_node_send_request_to_another_existing_node( new_node );
}
```

Fig. 2 Pseudo-code for joining the system

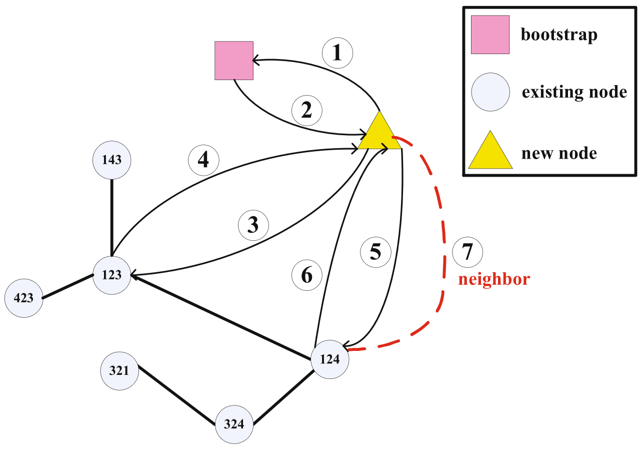


Fig. 3 Processes of the joining step

3.2.2 *Departing the system*

Departing refers to when a node leaves the AGO system. Each node needs to inform all of its existing neighbors in its neighbor table that it is leaving before it departs the AGO system. In this manner, the leaving node's neighbors can remove the leaving node from their neighbor tables after receiving the leaving information and assign this newly available ID to another node if necessary. As shown in Fig. 3, when node 321 departs the AGO system, it notifies its neighbor node 324. If any of the leaving node's neighbors finds that all of its neighbors have departed, it will join again by requesting a new node ID.

Sometimes, nodes leave the AGO system in abnormal ways and are unable to inform their neighbors about their leaving before they go. Nodes in the AGO system can repair their neighbor tables automatically. Nodes send messages to their neighbors periodically to check whether their neighbors still exist. If nodes do not receive their neighbors' responses, they remove the neighbor from their neighbor tables. Therefore, nodes can make their neighbor tables correct, and the AGO system can work more smoothly. This regular maintenance keeps the cost of maintaining neighbor tables low.

3.2.3 *Routing algorithm*

In addition to the above two major pieces used to build the AGO system, the AGO system also supplies a function, i.e., the routing algorithm. The AGO system employs one of the important properties of the arrangement graph, i.e., the single-digit difference in the node IDs, to construct the routing algorithm. All files in the AGO system are distributed to corresponding nodes using a hashing function. Then, if a node wants to access data, it also can use the hashing function to obtain the ID of the destination node that has the data. Then, the requesting node uses the routing algorithm to reach the destination node.

The routing algorithm can be divided into two components: comparing and transmitting. The comparing action is performed when a receiving node receives a routing request with a destination ID and the receiving node compares its ID with the destination ID. If its ID is equal to the destination ID, it returns its information to the requesting node. However, if its ID is not equal to the destination ID, it performs the transmitting action. In the transmitting action, there are three possible cases. First, the receiving node compares its neighbors' IDs with the destination ID. If one of its neighbor's IDs is equal to the destination ID, it transmits this routing request to that neighbor and is finished addressing the routing request.

If neither the receiving node's ID nor any of its existing neighbors' IDs matches the destination ID, the receiving node will loosen the transmitting rule. The receiving node compares its neighbors' IDs again and transmits the routing request to any neighboring node with an ID that is only one digit different from the destination ID.

In the worst case, the receiving node needs to transmit the routing request to its neighbors whose IDs have at least $\lfloor \frac{k}{2} \rfloor$ digits in common with the destination ID. In this manner, the AGO system avoids sending unnecessary messages and increasing system overhead. The routing algorithm is presented in Fig. 4.

In Fig. 5, there are some nodes in the proposed AGO system with $(n, k) = (4, 3)$. This figure is used as an example to illustrate the routing algorithm. If node 324 needs to access data that is on node 412, node 324 sends a routing request to node 314. Although node 324 has two existing neighboring nodes, i.e., node 321 and node 314, node 324 only sends the routing request to node 314 because node ID 314 has one digit that is the same as the destination node ID 412. Node ID 321 does not have any number in common with node ID 412, so node 324 does not send the routing request to node 321. After node 314 receives the routing request, it compares its ID and its neighbors' IDs with the destination ID. Then, node 314 only sends the routing request to node 312 because node ID 312 has only one digit different from the destination ID. Following this routing rule, node 412 will receive the routing request next and reply with its information to node 324. In addition, each node filters routing requests to avoid handling the same routing request multiple times.

```
Algorithm: Searching File
Input: An $ID$ that is hashed from the filename
Output: The information of the node whose ID is equal to $ID$
Begin
1: When node $A$ receives an $ID$ that is needed to search
2:  if $A$'s ID matches $ID$
3:       return its information;
4:  else if any of $A$'s existing neighbors' IDs matches $ID$
5:       sends the request to that neighbor;
6:       else compares $ID$ with $A$'s existing neighbors digit by digit
7:            if any of $A$'s existing neighbors' IDs is only one digit different from the $ID$
8:                 transmit this request to the neighbors;
9:            else sends the request to $A$'s existing neighbors whose IDs have at least
floor(k/2) digits the same as $ID$;
End
```

Fig. 4 The routing algorithm

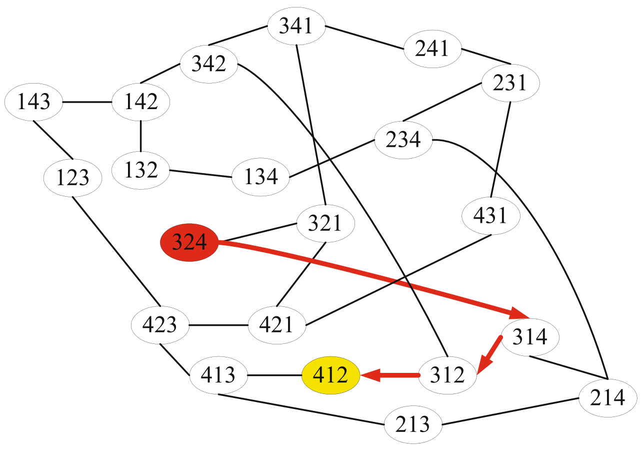


Fig. 5 An example of the routing algorithm

3.3 Theorem analysis

According to the theorem of the arrangement graph, the maximum number of nodes that the AGO system can accommodate is

$$N_{max} = \frac{n!}{(n-k)!},$$

and the diameter is $\left\lfloor \frac{3k}{2} \right\rfloor$. The diameter is the longest distance between any two nodes.

In general, the distance between any two nodes is related to the number of different digits between these two nodes, and the distance can be calculated by considering the following two cases.

Case 1 If there are x different digits between node A and node B, then node A needs x hops to reach node B, where $x \leq k$.

For example, if node A's ID is 123456, and node B's ID is 127856, there are two different digits between node A and node B. Node A can reach node B via

$$123456 \rightarrow 127456 \rightarrow 127856.$$

Therefore, node A needs two hops to reach node B, which is equal to the number of digits between nodes A and B.

Case 2 Consider all numbers of all digits of node IDs. If all numbers of all digits in nodes A and B are the same but there are x digits in different positions, node A may need more than x hops to reach node B.

For example, if node A's ID is 123456 and node B's ID is 124356, then node A can reach node B via

$$123456 \rightarrow 123756 \rightarrow 124756 \rightarrow 124356.$$

There are two digits different between nodes A and B, but node A needs three hops to reach node B. Therefore, from Cases 1 and 2, the following is deduced:

$$x \leq Hop_{search}, \tag{1}$$

where Hop_{search} is the number of hops that node A needs to take to reach node B. This result indicates that nodes may need to take more hops to reach other nodes.

According to the theorem of the arrangement graph, the following is also true:

$$Hop_{search} \leq \left\lfloor \frac{3k}{2} \right\rfloor. \tag{2}$$

From Eqs. 1 and 2, the relationship is derived as follows:

$$x \leq Hop_{search} \leq \left\lfloor \frac{3k}{2} \right\rfloor. \tag{3}$$

Then, consider the expected average number of hops nodes take to reach other nodes. The number of nodes that have t digits different from node A is expressed as follows:

$$diff(t) = \binom{k}{t} \times \left[\prod_{n-k+1}^{n-k+t} + \sum_{i=1}^{t-1} \left((-1)^i \times \binom{t}{i} \times \prod_{n-k+1}^{n-k+t-i} \right) + (-1)^t \right]. \tag{4}$$

Then, the expected average number of hops in Case 1 is as follows:

$$E(x) = \frac{\sum_{t=1}^{k} (diff(t) \times t)}{N_{max}}. \tag{5}$$

However, to account for Case 2, the relationship must be included as follows:

$$E(x) \leq E(Hop_{search}) \leq \left\lfloor \frac{3k}{2} \right\rfloor. \tag{6}$$

In functioning P2P systems, nodes often join/depart the system. Therefore, the AGO system will often have some IDs that are not assigned to any node. When a node tries to find another node, it may need to make a detour to avoid vacant nodes. In this case, more hops would be required to reach other nodes, even higher than $\left\lfloor \frac{3k}{2} \right\rfloor$.

Thus, the following may occur:

$$Hop_{search} > \left\lfloor \frac{3k}{2} \right\rfloor.$$

Although this situation may occur, it does not occur often. It occurs when the number of nodes in the AGO system is low and the arrangement graph is incomplete. As the number of nodes in the AGO system increases, the AGO system becomes more complete and Hop_{search} can be limited to within $\left\lfloor \frac{3k}{2} \right\rfloor$.

4 Experimental results

In this section, some experimental results of the proposed AGO system are presented to illustrate the system performance. OverSim [1, 2, 18, 24] is used to evaluate the performance of the proposed AGO system. The AGO is compared with some well-known P2P overlay networks, such as Chord, Pastry, and Kademlia. These overlay networks are used and discussed in many research papers.

4.1 Experimental environment

4.1.1 OverSim

This study uses OverSim as the simulation environment. OverSim is an open-source simulation framework for building overlay and P2P networks on top of OMNeT++ simulation environment [26]. It is a powerful and widely used simulator, and it provides several modules for investigations of P2P overlay networks. OverSim is flexible and allows for both structured and unstructured overlay networks. The AGO system is developed in the OverSim modules because of these many advantages of OverSim. OverSim is also scalable, which allows for the simulation of large-scale environments for real-world applications. Furthermore, several structured and unstructured P2P systems and overlay protocols are contained in OverSim, such as Chord, Kademlia, Pastry, and GIA.

4.1.2 Experimental setup

To evaluate the performance of the proposed AGO system, simulations of the AGO were executed under $(n, k) = (8, 6)$. The overlay framework OverSim was deployed to simulate overlay networks with 1,000–10,000 nodes, which are approximately 5–50 % of the maximum size of $A_{8,6}$. The percentage is calculated as follows:

$$P = \frac{N}{\frac{n!}{(n-k)!}},$$

where P is the percentage of the maximum nodes, N is the number of current nodes, and $\frac{n!}{(n-k)!}$ is the maximum number of nodes that $A_{n,k}$ can accommodate. According to the characteristics of the arrangement graph, the diameter is

Table 1 Parameters settings

Overlay	Parameters
AGO	$n = 8, k = 6$
Chord	$m = 14$
Pastry	$b = 4, l = 16$
Kademlia	Alpha $= 3$, $B = 160$, $k = 20$
R/Kademlia	Alpha $= 3$, $B = 160$, $k = 20$

equal to $\left\lfloor \frac{3k}{2} \right\rfloor = 9$, and the degree of each node is equal to $k(n-k) = 12$.

A series of simulations were performed to collect data and establish the system performance. In the P2P environment, nodes join/leave frequently, and therefore, the different churn rates help the experiments model the real world more closely. Nodes are assigned different lifetimes using the Weibull distribution [25] to simulate the different churn rates. The mean lifetime was varied between 1,000 and 10,000 s for simulations with different churn scenarios.

Bandwidth consumption is another important issue in the P2P environment. The bandwidth consumption is expressed as the average sent and received message bytes per second of a node. This issue is often argued and discussed because the dynamic environment in P2P systems often creates a large number of messages. In experiments, the test application sent a 100-byte message every 60 s to several live current nodes with a normal distribution. Each protocol and parameter was simulated 10 times, and the average data were calculated. The experimental results are more accurate under this protocol. Table 1 shows some related settings of parameters; definitions of the parameters are presented in Section 2.

4.2 Joining the system

In P2P systems, each node is an independent individual node. Therefore, to join the system, each node must take a series of steps to inform other nodes who are already in the system. Each node also needs to obtain a correct position, explore its neighbors, and maintain its neighbor table. All of these actions produce many messages and may cause additional system overhead. This study aims to help nodes join the system quickly and to reduce the number of created messages.

Figure 6 presents the number of messages created in building the AGO system compared with Chord, Pastry, Kademlia, and R/Kademlia. It shows the *log* of the number of created messages because the number of messages Pastry created is much larger than the number created by the other overlay systems. Therefore, to visualize the relationship between all of the different overlay systems, the results must be viewed on a *log* scale. Table 2 displays the complete data of the number of created messages. As shown in Table 2, the proposed AGO system creates the least number of messages despite the number of nodes.

From Table 2, the AGO only created 20–40 % of the messages that Chord created. The AGO only created 1 %, 15–33 %, and 20–45 % of the messages created by Pastry, Kademlia, and R/Kademlia, respectively. These findings indicate that the proposed AGO system can reduce the number of created messages in building the system. Furthermore, the AGO system can reduce system overhead in building and maintaining the system.

4.3 Average routing hops

In distributed systems, such as P2P systems, searching for resources efficiently is an important issue, particularly since

Fig. 6 Number of created messages

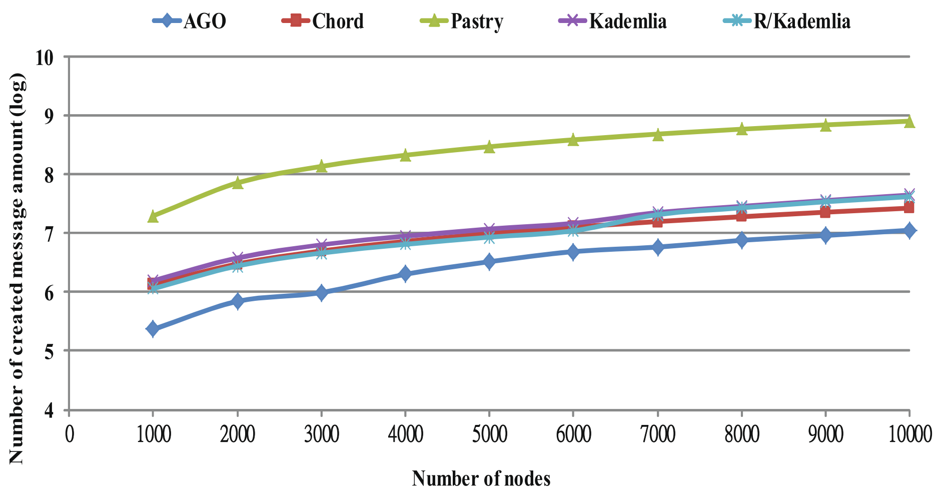
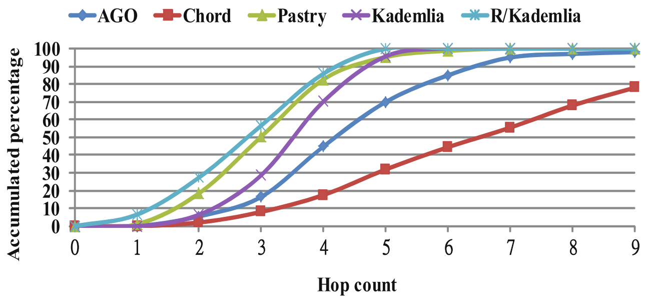
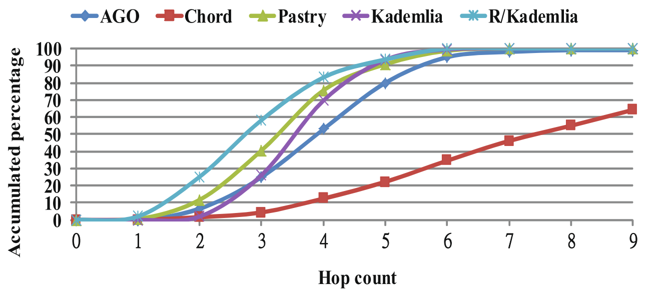


Table 2 Number of created messages

Nodes	AGO	Chord	Pastry	Kademlia	R/Kademlia
1000	232,797	1,359,830	19,863,140	1567,417	1,148,930
2000	697,350	3,031,425	72,058,328	3,820,027	2,793,551
3000	982,692	5,045,875	138,043,748	6,365,253	4,634,724
4000	2,018,987	7,332,288	213,784,198	8,981,605	6,525,248
5000	3,292,855	9,908,696	298,445,765	11,904,636	8,619,766
6000	4,890,596	12,737,604	389,899,887	14,854,760	11,033,261
7000	5,837,528	15,866,716	488,145,456	22600149	20,875,733
8000	7,616,313	1,9193,316	591,390,248	28,758,395	27,146,847
9000	9,165,380	22,878,497	699,317,417	36,224,712	34,403,991
10000	11,347,705	26,818,547	810,098,957	45,037,570	42,046,195



a N = 2,000



b N = 8,000

Fig. 8 Cumulative percentage of nodes

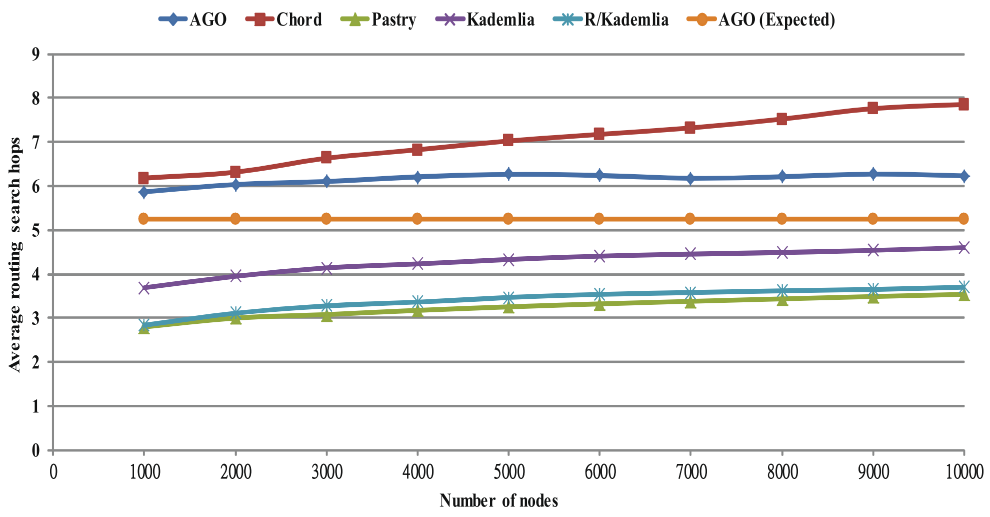
the development of the Internet, as more and more people search for resources on the Internet. Therefore, a method of increasing the efficiency of the routing without increasing system overhead is the goal of the AGO routing system.

Figure 7 illustrates the average routing search hops in each P2P system with different numbers of nodes. As shown in Fig. 7, the line of "AGO (Expected)" is the expected average number of routing hops calculated using the previously mentioned equations with $(n, k) = (8, 6)$. According to Eqs. 4 and 5, the expected average number of routing hops is equal to 5.25. From Fig. 7, we can observe that the average routing search hops in the AGO system is slightly higher than the theoretical expected value of average search hops from Case 2, as mentioned in the Section 3.3.

Files are evenly distributed to all nodes to evaluate average routing hops. The average routing search hops in the AGO system is nearly constant despite the number of nodes. Therefore, regardless of the number of nodes, average routing search hops is nearly constant. The average routing search hops in the AGO system are better than in Chord, but a few hops worse than in other P2P systems. However, average routing search hops in those P2P systems slightly increase when the number of nodes increases, and they also generate a large number of messages and consume large bandwidth, which will be shown in the next subsection.

Figure 8 shows the cumulative percentage of nodes using different routing hops. Two cases, in terms of number of nodes, are analyzed ($N = 2,000$ and $N = 8,000$) to illustrate results. As shown in Fig. 8, even though average routing search hops in the AGO system are worse than for the other three P2P systems, most of the nodes in the AGO system can find resources within six or seven hops, and

Fig. 7 Average routing search hops



the other three P2P systems need five or six hops. When there are 8,000 nodes, most nodes can find resources within six hops regardless of the system being employed. Therefore, when there are more nodes, the routing performance of the AGO system is more efficient, implying that the AGO system is more suitable for a large-scale environment. Furthermore, experimental results show that routing hops of over 90 % of the nodes can be bound within diameter hops because the join mechanism assigns IDs from existing nodes and groups around existing nodes. Therefore, nodes will not be sparsely distributed in the system, and routing is more efficient.

Table 3 Message quantity and message size of each node in each P2P system

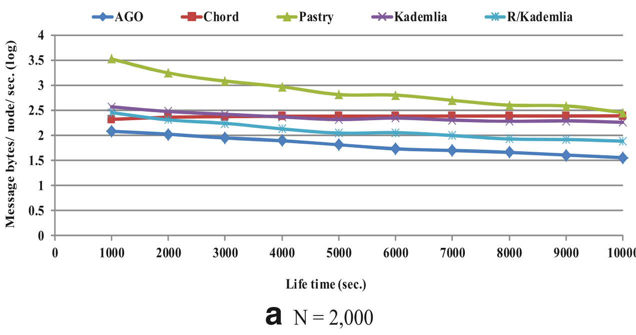
LifeTime	N	message	AGO	Chord	Pastry	Kademlia	R/Kademlia
1000	$N = 2000$	Bytes	120.33	208.19	3323.30	370.85	284.50
		Amounts	0.65	2.57	38.92	3.06	1.58
	$N = 8000$	Bytes	165.76	233.82	4771.25	575.91	509.75
		Amounts	0.96	2.85	66.4	5.45	4
2000	$N = 2000$	Bytes	105.65	229.64	1751.11	297.70	203.63
		Amounts	0.53	2.78	19.57	2.87	1.1
	$N = 8000$	Bytes	153.98	275.02	2546.73	474.01	419.57
		Amounts	0.90	3.25	32.17	5.42	3.78
3000	$N = 2000$	Bytes	88.68	236.03	1205.94	263.57	173.86
		Amounts	0.51	2.84	13.4	2.75	0.9
	$N = 8000$	Bytes	138.77	287.59	1649.73	443.47	384.20
		Amounts	0.83	3.37	20.68	5.4	3.73
4000	$N = 2000$	Bytes	78.36	240.00	930.77	231.85	134.77
		Amounts	0.48	2.88	10.15	2.72	0.72
	$N = 8000$	Bytes	120.33	293.01	1236.52	427.44	372.52
		Amounts	0.76	3.43	15.5	5.36	3.69
5000	$N = 2000$	Bytes	65.12	241.14	649.88	207.26	110.96
		Amounts	0.45	2.88	7.43	2.69	0.62
	$N = 8000$	Bytes	116.89	296.58	963.98	407.93	353.38
		Amounts	0.71	3.46	12.37	5.33	3.66
6000	$N = 2000$	Bytes	53.76	241.28	634.89	211.48	103.25
		Amounts	0.45	2.89	7.01	2.67	0.60
	$N = 8000$	Bytes	112.96	299.64	831.49	397.82	344.53
		Amounts	0.67	3.49	10.59	5.31	3.63
7000	$N = 2000$	Bytes	49.99	242.38	499.81	201.67	99.53
		Amounts	0.44	2.9	5.74	2.65	0.54
	$N = 8000$	Bytes	110.32	301.00	686.07	393.54	337.11
		Amounts	0.62	3.5	8.92	5.28	3.61
8000	$N = 2000$	Bytes	45.78	243.33	398.98	190.54	84.76
		Amounts	0.43	2.91	4.6	2.64	0.50
	$N = 8000$	Bytes	108.33	301.45	594.98	385.32	326.17
		Amounts	0.60	3.51	7.85	5.27	3.57
9000	$N = 2000$	Bytes	40.52	243.78	386.79	184.87	82.50
		Amounts	0.39	2.91	3.68	2.63	0.48
	$N = 8000$	Bytes	105.65	303.30	536.66	383.06	326.05
		Amounts	0.56	3.52	7.1	5.25	3.55
10000	$N = 2000$	Bytes	35.98	244.43	284.51	182.00	76.64
		Amounts	0.33	2.92	3.32	2.62	0.45
	$N = 8000$	Bytes	103.56	303.33	473.58	379.50	323.35
		Amounts	0.49	3.55	6.25	5.19	3.51

4.4 Bandwidth consumption

In a network environment, bandwidth often affects the performance of message transmission. Therefore, bandwidth consumption is an important issue in P2P systems. In this study, reducing bandwidth consumption is one of the goals of the AGO system. The bandwidth consumption is the average sent and received message rate (bytes per second) of a node. Evaluating the message size and quantity of each node, the bandwidth consumption of the AGO system and other P2P systems can be identified and compared. The size and number of messages produced by routing or maintaining neighbor tables can greatly affect performance in terms of bandwidth.

Table 3, Figs. 9 and 10 show the average message size and number under different churn rates to evaluate the bandwidth consumption. The message size and number are compared on a *log* scale because the number of messages created by Pastry is significantly higher than those created by the other overlay systems. In Fig. 10, because the original values of the AGO system are less than one, the values after taking the *log* are less than zero. Table 3 shows the complete data for Figs. 9 and 10.

From the above results, the average message size (in bytes) and quantity (amount) decrease as the lifetime increases because the P2P systems become more stable. When the mean lifetime is short, nodes receive and send more messages in P2P systems, increasing the overhead of the system.



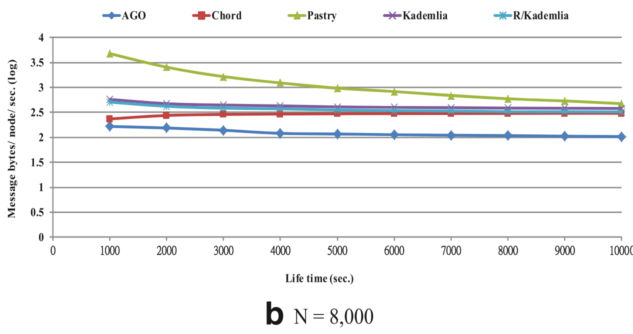
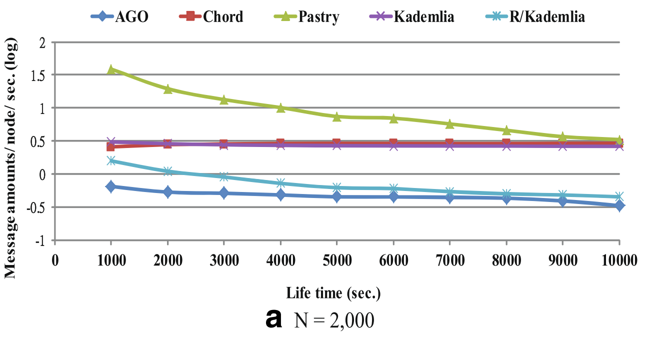


Fig. 9 Average message size of each node (bytes)



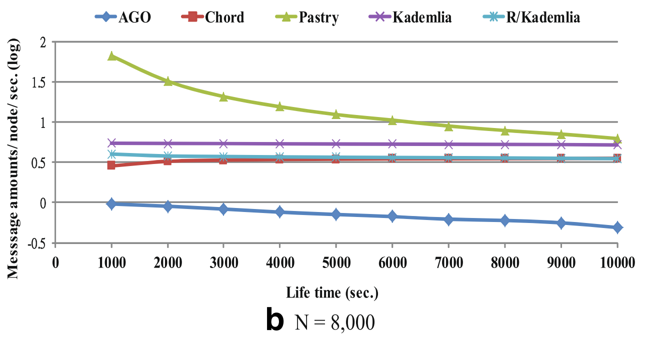


Fig. 10 Average number of messages for each node

Table 3 shows that the AGO system consumes the least bandwidth. When there are 2,000 nodes, the AGO system only consumes 15–58 % of the bandwidth compared with Chord and consumes approximately 4–13 %, 20–35 %, and 42–60 % of the bandwidth compared with Pastry, Kademlia, and R/Kademlia, respectively. Overall, the AGO system showed great performance compared with other P2P systems. In addition, the AGO system creates the fewest messages for system routing and maintenance. When there are 2,000 nodes, the AGO system creates 11–25 % of the messages created by Chord and creates 2–10 %, 13–21 %, and 41–86 % of the messages produced by Pastry, Kademlia, and R/Kademlia, respectively. From these results, the AGO system consumes the least bandwidth and reduces system overhead compared to the other P2P systems.

4.5 Message latency

Message latency can also affect system performance. To determine whether the AGO system spends too much time sending packets, the time spent sending packets between two nodes was measured. If the time spent sending messages between two nodes is too long, nodes must wait a long time for the response. In particular, when transmitting requests to other nodes, the original requesting node must wait a long time for the return response. Thus, the time of one-way latency was used to compare the AGO system with the other P2P systems.

Table 4 One-way latency for each P2P system

Nodes	AGO	Chord	Pastry	Kademlia	R/Kademlia
1000	0.23	0.41	0.11	0.46	0.21
2000	0.26	0.45	0.11	0.51	0.22
3000	0.26	0.47	0.11	0.54	0.23
4000	0.26	0.5	0.11	0.56	0.24
5000	0.27	0.51	0.11	0.58	0.25
6000	0.27	0.51	0.11	0.6	0.25
7000	0.28	0.52	0.11	0.6	0.26
8000	0.28	0.53	0.11	0.6	0.26
9000	0.28	0.54	0.11	0.61	0.26
10000	0.28	0.55	0.11	0.62	0.27

Table 4 and Fig. 11 show the one-way latency of the AGO system and other P2P systems. As shown in Table 4 and Fig. 11, the latency of the AGO system is better than those of Chord and Kademlia and slightly worse than those of R/Kademlia and Pastry. Although the latencies of the AGO system increase slightly as the number of nodes increases, the degree of increase is not as great as those of Chord and Kademlia.

From the experimental results discussed above, the AGO system performs very well. The proposed AGO system can build the system without producing large system overhead and has an efficient routing algorithm that is limiting within the diameter value. Although some overlay networks have fewer average routing hops than the AGO system, those overlay networks also produce a larger number of messages than the AGO system. The AGO system may need one to three more routing hops than the other overlay networks, but the AGO system also produces 20 % fewer messages, as shown in Fig. 3. The AGO does not construct as many connections with other peers and does not produce as large a number of messages for routing. In terms of bandwidth consumption, the AGO system has a significantly lower consumption than the other P2P systems. These experimental results show that the AGO system is also suitable for large-scale environments because its performance does not deteriorate as the number of nodes increases.

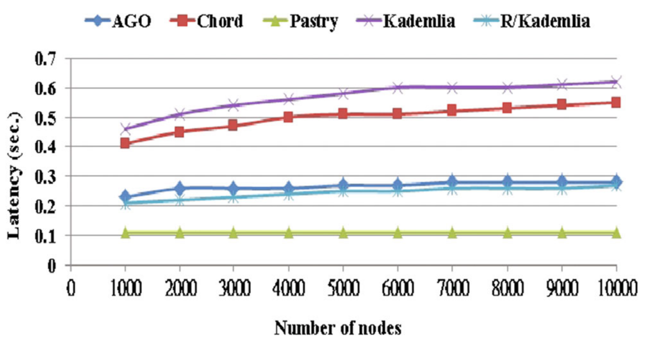


Fig. 11 One-way latency

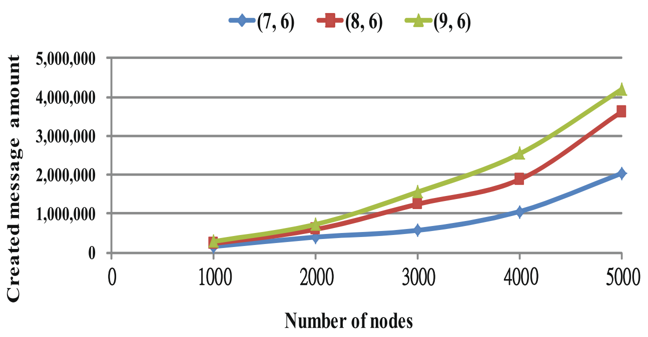


Fig. 12 Number of created messages using different (n, k) values

4.6 Comparisons under different parameters

The maximum number of nodes that the AGO system can accommodate is affected by parameters (n, k). It is hoped that AGO can construct the system with lower system overhead for different values of (n, k).

Some preliminary experimental results are presented in this study to determine the effects on the system overhead. As shown in Fig. 12, the parameter k is constant at six, and only the parameter n is varied. In this manner, the maximum number of nodes can be increased without changing the diameter. In Fig. 12, parameters (n, k) are $(7, 6)$, $(8, 6)$, and $(9, 6)$. The experimental results show that the smaller parameter n creates fewer messages when the parameter k is also small. In the future, the AGO system can be constructed with smaller parameters (n, k) and be dynamically adjusted according to the number of nodes in the system.

5 Conclusions and future work

In this study, the arrangement graph is adopted to form a P2P overlay network, and the property of the single-digit difference of node IDs is utilized to provide an efficient routing algorithm and reduce system overhead. Experimental results have shown the efficiency of building the system and routing algorithm. The AGO system is also suitable for large-scale environments without resulting in excessive system overhead.

In future studies, the routing algorithm in the AGO system will be improved by adding a replica mechanism, which can reduce the average routing hops and improve the routing efficiency. The (n, k) parameters of the AGO system will also be dynamically adjusted according to the number of nodes. This dynamic adjustment can make the arrangement graph more complete, thus limiting the routing hops within the diameter number.

From the preliminary experimental results, with the same number of nodes under different parameters (n, k), the AGO system produces different overheads. When parameters (n, k) are small, the AGO system produces less system overhead. Therefore, it is hoped that the AGO system can be constructed under smaller parameters (n, k) and that the parameters (n, k) can be dynamically adjusted according to the number of nodes in the system. The AGO system can also be used on other distributed systems, such as grid or cloud systems, or on remote home care systems by assigning each node a unique ID to share resources.

References

1. Baumgart I, Heep B, Krause S (2007) OverSim: a flexible overlay network simulation framework. In: The proceedings of 10th IEEE global internet symposium (GI '07) in conjunction with IEEE INFOCOM 2007. Anchorage, pp 79–84, doi:10.1109/GI.2007.4301435
2. Baumgart I, Heep B, Krause S (2009) OverSim: a scalable and flexible overlay framework for simulation and real network applications. In: The proceedings of IEEE ninth international conference on peer-to-peer computing (P2P'09). Seattle, pp 87–88
3. Chiang W-K, Chen R-J (1998) On the arrangement graph. J Inf Process Lett 66(4):215–219
4. Chen Y-S, Juang T-Y, Shen Y-Y (2000) Multi-node broadcasting in an arrangement graph using multiple spanning trees. In: The proceedings of the seventh international conference on parallel and distributed systems, pp 213–220
5. Day K, Tripathi A (1992) Arrangement graphs: a class of generalized star graphs. Inf Process Lett 42:235–241
6. Day K, Tripathi A (1993) Embedding of cycles in arrangement graphs. J IEEE Trans Comput 42(8):1002–1006
7. Day K, Tripathi A (1993) Embedding grids, hypercubes, and trees in arrangement graphs. In: The proceedings of international conference on parallel proceeding (ICPP 1993), vol 3. USA, pp 65–72
8. Heep B (2010) R/Kademlia: Recursive and topology-aware overlay routing. In: The proceedings of australasian telecommunication networks and applications conference, pp 102–107
9. Hsieh S-y, Chen G-H, Ho C-W (1997) Fault-tolerant ring embedding in faulty arrangement graphs. In: The proceedings of international conference on parallel and distributed systems. Seoul, pp 744–749
10. Haribabu K, Reddy D, Hota C, Yla-Jaaski A, Tarkoma S (2008) Adaptive lookup for unstructured peer-to-peer overlays. In: The proceedings of 3rd international conference on communication systems software and middleware and workshops 2008 (COMSWARE 2008). Bangalore, pp 776–782
11. Jiang S, Guo L, Zhang X (2003) LightFlood: an efficient flooding scheme for file search in unstructured peer-to-peer systems. In: The proceedings of the 2003 international conference on parallel processing (ICPP 2003). Kaohsiung, pp 627–635
12. Jiang S, Guo L, Zhang X (2008) LightFlood: minimizing redundant messages and maximizing scope of peer-to-peer search. J IEEE Trans Parallel Distrib Syst (TPDS) 19(5):601–614
13. Kobayasi M, Nakayama H, Ansari N, Kato N (2007) NHAG: network-aware hierarchical arrangement graph for application layer multicast in heterogeneous networks. In: The proceedings of 2007 global telecommunications conference (GLOBECOM '07), pp 1993–1997
14. Kobayashi M, Nakayama H, Ansari N, Kato N (2009) Robust and efficient stream delivery for application layer multicasting in heterogeneous networks. J IEEE Trans Multimedia 1(1):166–176
15. Lua EK, Crowcroft J, Pias M, Sharma R, Lim S (2005) A survey and comparison of peer-to-peer overlay network schemes. J IEEE Commun Surv Tutor 7(2):72–93
16. Lu S-H, Lai K-C, Yang D-L, Li K-C, Chung Y-C (2010) Pervasive health service system: insights on the development of a Grid-based personal health service system. In: Proceedings of the 12th international conference on e-health networking, application and services (Healthcom2010). Lyon, pp 61–67
17. Maymounkov P, Mazieres D (2002) Kademlia: a peer-to-peer information system based on the XOR metric. In: The proceedings of the first international workshop on peer-to-peer systems. USA, pp 53–65
18. Munoz-Gea JP, Malgosa-Sanahuja J, Manzanares-Lopez P, Sanchez-Aarnoutse JC, Martinez-Rojo AM (2009) Simulation of a P2P application using OverSim. In: The proceedings of the first international conference on advances in future internet (AFIN 2009). Athens, pp 53–60
19. Rowstron A, Druschel P (2001) Pastry: scalable, distributed object location and routing for large-scale peer-to-peer systems. In: The proceedings of the IFIP/ACM international conference on distributed systems platforms Heidelberg (Middleware 2001), lecture notes in computer science, vol 2218/2001, pp 329–350
20. Ratnasamy S, Francis P, Handley M, Karp R, Shenker S (2001) A scalable content addressable network. In: The proceedings of conference on applications, technologies, architectures, and protocols for computer communications (ACM SIGCOMM 2001), pp 161–172
21. Stoica I, Morris R, Karger D, Frans Kaashoek M, Balakrishnan H (2001) Chord: A scalable peerto-peer lookup service for internet applications. In the proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications, pp 149–160. San Diego
22. Stoica I, Morris R, Liben-Nowell D, Karger D, Frans Kaashoek M, Dabek F, Balakrishnan H (2003) Chord: a scalable peer-to-peer lookup service for internet applications. J IEEE/ACM Trans Netw 11(1):17–32
23. Tian R, Zhang Q, Xiang Z, Xiong Y, Li X, Zhu W (2005) Robust and efficient path diversity in application-layer multicast for video streaming. J IEEE Trans Circ Syst Video Technol 15(8):961–972
24. The OverSim P2P Simulator - http://www.oversim.org/
25. Weibull distribution - http://en.wikipedia.org/wiki/Weibull_distribution
26. OMNeT++ Network Simulation Framework - http://www.omnetpp.org

Ssu-Hsuan Lu received a B.S. and M.S. degree in Information Management from Providence University in 2002 and 2005, respectively. She is currently working toward the Ph.D. degree in the Department of Computer Science at the National Tsing Hua University, Taiwan. Her research interests include parallel and distributed processing, cluster systems, peer-to-peer computing, grid computing, distributed shared memory, cloud computing, healthcare, and IoT.

Kuan-Chou Lai received his M.S. degree in Computer Science and Information Engineering from the National Cheng Kung University in 1991, and the Ph.D. degree in Computer Science and Information Engineering from the National Chiao Tung University in 1996. Currently, he is an associate professor in the Department of Computer and Information Science at the National Taichung University, Taiwan. His research interests include parallel processing, parallel compiler, system architecture, P2P computing, grid computing, and cloud computing. He is a member of the IEEE and the IEEE Computer Society.

Kuan-Ching Li is currently a Professor in the Department of Computer Science and Information Engineering at the Providence University, Taiwan. He has served in a number of journal editorial boards and guest editorship, as also served many international conference chairmanship positions as steering committee, advisory committee, general and program committee chairs and member of program committees. In addition, he has also named as Guest Professor at Lanzhou, Xiamen and Shanghai Universities in China. His research interests include networked computing (Clusters, Grids, P2Ps, Clouds), parallel software design, and performance evaluation and benchmarking. He is a senior member of the IEEE and a Fellow of the IET.

Yeh-Ching Chung received a B.S. degree in Information Engineering from Chung Yuan Christian University in 1983, and the M.S. and Ph.D. degrees in Computer and Information Science from Syracuse University in 1988 and 1992, respectively. He joined the Department of Information Engineering at Feng Chia University as an associate professor in 1992 and became a full professor in 1999. From 1998 to 2001, he was the chairman of the department. In 2002, he joined the Department of Computer Science at National Tsing Hua University as a full professor. His research interests include parallel and distributed processing, cluster systems, grid computing, multi-core tool chain design, and multi-core embedded systems. He is a member of the IEEE computer society and ACM.