Second LACCEI International Latin American and Caribbean Conference for Engineering and Technology (LACCEI'2004)
"Challenges and Opportunities for Engineering Education, Research and Development"
2-4 June 2004, Miami, Florida, USA

Hierarchical P2P Systems in a Small World

Xiuqi Li
Instructor, Florida Atlantic University, Boca Raton, Florida, USA

Jie Wu, Ph.D
Professor, Florida Atlantic University, Boca Raton, Florida, USA

Abstract
Hierarchical organizations in general boost overall system scalability. Some existing research work organizes peers into different hierarchical structures. In these systems, the top-tier overlay is either a completely-connected graph or a CHORD ring. These overlays can achieve good routing latency ($O(1)$ or $O(\log^n)$), where n is the number of groups. However, each node has to keep a large number (n or $O(\log^n)$) of TCP connections to other nodes on the top-tier. More connections means more work on the underlying network and more interference between nodes. In this paper, we propose a novel small-world top-tier overlay that reduces the routing state to $O(1)$. The routing latency is $O(\log^2 n)$. In our approach, each peer on the top-tier overlay is equipped with some short links and some random long links. The simulation result demonstrates the effectiveness of our approach.

Keywords
Peer-to-peer networks, Hierarchical routing, Data management

1. Introduction

Peer-to-Peer (P2P) file sharing applications have been very popular in recent years. In a P2P file sharing system, files are stored distributedly in many independent nodes. These nodes form a virtual network (an overlay) on top of existing Internet networks. The main system function is *lookup* service: search in the system for a specific file. Many systems use a *Distributed Hash Table* (DHT) to realize the file lookup service. A DHT is a hash table of file entries that are maintained in a distributed fashion by all peers in the system. In such DHT-based ssystems, both files and nodes are hashed to the same ID space. A file is stored in a node whose node ID is closest to the file ID. Given a query for a file, the lookup service routes the query to the node responsible for that file. Chord (Stoica et al., 2001), CAN (Ratnasamy et al., 2001), Pastry (Rowstron and Druschel, 2001), and Tapestry (Zhao et al., 2001) are such systems. To implement the DHT, Chord uses a ring structure; Pastry and Tapestry employes a tree-like structure; CAN uses a d-dimensional cordinate space. Chord, Pastry, and Tapestry achieve $O(\log^n)$ routing latency (in terms of overlay hops) using $O(\log^n)$ routing state (in terms of the number of overlay neighbors). CAN achieves $O(d\sqrt[d]{n})$ lookup latency using $O(d)$ state. A recent system, Symphony (Manku et al., 2003), utilizes a small-world model overlay to implement the DHT. With only $k = O(1)$ links per node, a query can be resolved in $O(\log^2 n)$ hops.

Chord, CAN, Pastry, Tapestry, and Symphony employ flat overlay designs. Some hierarchical overlays have also been proposed in the literature. Hierarchical designs generally boost system scalibility. They can also take advantage of node heterogeneity. Typically, hierarchical systems divide nodes into different groups. Each group forms its own overlay architecture. Groups are arranged into different layers. A node may be in one or more layers. Kelips (Gupta et al., 2003) and the system in (Mizrak et al., 2003) arrange the nodes into a two-tier hierarchy. In the top-tier, each node keeps connections with all other nodes in the top-tier overlay. The system in (Garces-Erice et al., 2003) also organizes nodes into a two-tier hierarchy. The top-tier is a Chord ring. Coral (Freedman and Mazieres, 2003) and Hieras (Xu et al., 2003) adopt three-tier overlays. Each group overlay is a Chord ring.

It is observed that in existing hierarchical systems, the top-tier overlay is either a completely connected graph or a Chord ring. Each node on the top-tier overlay has to maintain $O(n)$ or $O(\log^n)$ routing state, where n refers to the number of nodes on the top-tier. This means that each top-tier node has to maintain TCP connections with all other nodes or $O(\log^n)$ other nodes on the top-tier. If one node leaves, it will affect all other nodes or $O(\log^n)$ other nodes. Therefore, we propose a two-tier overlay hierarchy. The top-tier nodes are organized into a small-world model. On the bottom tier, each group forms a Chord ring. In our model, each node on the top-tier only needs to maintain a small number ($k = O(1)$) of connections with other nodes. The routing latency on the top-tier is $O(\log^2 n)$.

This paper is organized as follows. In Section 2, small-world models and existing hierarchical systems that are related to our approach will be described. In Section 3, the proposed overlay hierarchy will be presented in detail. We will focus our discuss on the small-world top-tier overlay and the file placement and retrieval. In Section 4, the system performance will be analyzed and the experimental result will be illustrated. We conclude this paper in Section 5.

2. Related work

Kelips, Coral, Hieras, and the systems in (Mizrak et al., 2003) and (Garces-Erice et al., 2003) are hierarchical P2P systems. Kelips is composed of k groups. In each group, a constant number of nodes are selected as group contacts. These contacts provide entrance to files stored in that group. Nodes are hashed to their belonging groups. Each node keeps information about all other nodes in the same group, and the contacts in all other groups. Each node also keeps the indexes of files stored in its own goup. A file is hashed to a group and then stored in a randomly selected node. This node is called the home node of that file. To look for a file, the querying node hashes the file to a group. If the group is its own group, the file is found from the query node's index. If not, the query is forwarded to one contact in the group to which the file is hashed. This contact finds the file's homenode by checking its file indexes. The information about nodes and files indexes at each node are made current through periodic update. The message update between a node and contacts in other groups are conducted using a uniform gossip protocol with a latency of $O(\log^n)$. The message update among nodes in the same group is carried out through a spatially weighted gossip protocol with a latency of $O(\log^2 n)$. Kelips achieves $O(1)$ query latency through increased background traffic for routing state maintenance.

The system in (Mizrak et al., 2003) is also designed for $O(1)$ routing latency. The overlay is a two-tier hierarchy. All nodes are part of the bottom tier: a Chord ring. A Chord ring is a ring where each node keeps connections to other nodes that are 1/2-ring away, 1/4-ring away, 1/8-ring away, etc. The Chord ring is divided into a number of arcs. In each arc, some node is selected as the superpeer for that arc. All superpeers form the top-tier. Each superpeer keeps a peer table and a superpeer table. The peer table contains the node ids and addresses of nodes in an arc. The superpeer table consists of the node ids, addresses, and arc ranges of all superpeers. Files are stored similarly to Chord. To query for a file, the

querying node sends the query directly to its superpeer. If the file key is in its arc, the superpeer locates the responsible node in its peer table. If not, the superpeer lookups its superpeer table and forwards the query to the superpeer responsible for the arc containing the file key. That superpeer finds the responsible node from its peer table. To maintain the superpeer tables consistent, keep-alive messages are sent periodically among superpeers.

The system in (Garces-Erice et al. 2003) is a two-tier overlay. At the bottom tier, different groups of nodes form their own overlays that may or may not be the same. The top-tier is a Chord ring that includes all superpeers chosen from each group. For fault tolerance purpose, there is more than one superpeer per group. Coral is a 3-tier hierarchy of Chord rings. All nodes in the same ring have similar round-trip time. All rings in the same tier have similar round-trip time. Each node belongs to one ring in each tier and has the same node ID in all its belonging rings. Queries are resolved first at the bottom tier, and then higher tiers. Hieras is very similar to Coral. The difference is on how to estimate round-trip time. Coral uses ping-pong messages while Hieras uses distributed binning.

Symphony organizes all nodes into a flat small-world network. All nodes have ids in the rang [0,1). Each node keeps a short link to its immediate predecessor and a short link to its immediate successor in the node ID ring. Each node also keeps k long links to nodes probabilistically selected from the ID ring. Each node is responsible for the arc between its node id and the id of its immediate predecessor. Files have m-bit keys. A file with key f is stored in a node whose arc contains $f/2^m$. On receiving a query for the key f, each node forwards a query to its neighbor (close or distant neighbor) that minimizes the distance to $f/2^m$. Both unidirectional routing and bidirectional routing can be used. In bidirectional routing, each node treats the nodes at the other end of its incoming links as its distant neighbors for query forwarding. Unidirectional routing minimizes the clock-wise distance at each node while bidirectional routing minimizes the absolute distance at each node. The look-ahead approach is proposed to speed up the query processing. In 1-lookahead, each node forwards the query to the neighbor whose neighbor is closest to $f/2^m$.

3. The proposed P2P approach

Figure 1 illustrates the proposed P2P approach. It is a two-tier hierarchy. The bottom tier consists of many independent Chord rings. Each Chord ring includes all nodes in the same group. The top tier is a small-world network. This top tier contains all superpeers in all groups. Each group has a constant number of superpeers. The top tier overlay and the bottom tier overlays are independent. Nodes are hashed to their belonging groups. Inside each group, nodes are assigned random node ids to form the Chord ring. The Chord ring in each group operates in the same way as in (Stoica et al., 2001).

3.1 The small-world top-tier overlay

We adapted the flat small-world model in Symphony to our top-tier overlay. Our small-world top-tier overlay is constructed by all superpeers based on their group ids. In the top-tier, each node corresponds to a group. Node ids are actually group ids. The information about each top-tier node includes the group id, the addresses and heartbeat counts of all group superpeers. Group ids form a ID ring. Each group has a short link to its immediate predecessor group and a short link to its immediate successor group. In addition, each group also keeps long links to k distant groups. These distant groups are selected based on a probabilistic density function that are associated with the number of groups and the distance between groups. Figure 1 shows two short links and a long link. The solid directed lines are actual links. The dashed directed lines are not real links. They denote that there are other groups in between. Double-arrowed lines all together denote the basic group ID ring. In this overlay, each group has two short links and one long link. Group b has short links to both group c and group a, and a long link to group i. Group ids are in the range of [0,1). A group manages the values between its group id and the group id of its predecessor group.

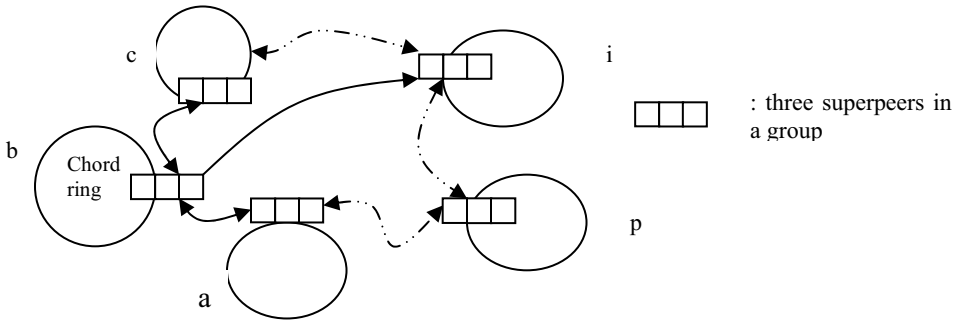

Figure 1: The proposed two-tier overlay

3.1.1 Long links to distant groups

Each group has long links to k distant groups. The distant groups are selected based on the following probabilistic desnsity function:

$$f(y) = \frac{1}{y \ln n}, \text{ where } y \in [1/n, 1]$$

In the formula, n refers to the total number of groups; y denotes the distance between groups.

To select a long link to a distant group, a superpeer in a group first draws a random number y based on the function $f(y)$ above. The group whose group id is the smallest number larger than y is then chosen as the distant group. The information about this group is discovered through the top-tier routing. To restrict the maximum number of connections per group on the top-tier, each group is allowed to have at most $2*k$ incoming long links. If this limit is reached at some distant group, the group making the selection has to try another distant group. After a maximum number of retries, if the selection still fails, the group making the selection will give up. Therefore the number of long links at a group may be smaller than k.

The current number of groups can be estimated similarly to Symphony. Three groups are chosen. Their segment lengths are sumed up to *seg_sum*. The closest integer to $3/seg\_sum$ is the estimate of the number of groups. It is expected that the number of groups is dynamic during the system initialization. When the system is stabilized, the number of groups should remain about the same. However, the number of nodes per group will still be dynamic. The specific number of long links per group: the value of k can be selected experimentally. The simulaion shows that k can be as small as 4 for a good performance.

3.1.2 Node join and leave

On the top-tier, node join refers to group join. Only group superpeers are part of the top-tier overlay. Therefore, group join occurs when the first peer in a group joins the network. This first peer becomes the group superpeer. Specifically, when a new peer joins the network, it is hashed to its group with group id g and its node id n in the group. Then it contacts some well-known group superpeer. This superpeer looks up the new peer's group id on the top-tier. In general, the query returns the group whose group id is greater than or equal to g. If the group id returned g' is not the same as the new peer's group id, it means that this new peer is the first peer in the group. This new peer becomes a superpeer in its group and joins the top-tier overlay. This new superpeer sets g' as its immediate successor group. It also sets the immediate predecssor group of g' as its own immediate predecessor group. Next, this new superpeer informs a superpeer in group g' that the new group g is the new immediate predecessor group of g'. Then the new superpeer adds long links to k distant groups based on the probabilistic density function in Subsection 3.1.1. If the group id returned g' is the new peer's group id, the new peer is a regular peer. This new peer reaches one superpeer in its group. This group superpeer will get the successor of the new peer in the group Chord ring and return it to the new peer. The new peer then joins the group Chord ring in the same way as in (Stoica et al., 2001).

On the top-tier, node leave means group leave. A graceful group leave refers to the scenario when all superpeers in a group gracefully leave the network. This should seldomly occur because there should be at least one superpeer in operation unless all peers in the group leave. A group failure refers to the sceneario when all group superpeers fail. This should also seldomly occur because the system maintains a constant number of superpeers per group. When one superpeer fails, another superpeer is selected to replace the failed one. The first *c* number of peers in a group are all superpeers. Later on, the superpeers are selected based on the connection time. The most stable one is chosen as the new superpeer. All superpeers periodically notifies all neighbor groups about their existence. Failed superpeers are detected through time-out and removed from the superpeer list.

3.2 File placement and query routing

When a new file is inserted, it is hashed to a group *fg* and the file key *fk* in that group. The file is stored in the node in the group *fg* whose node id is the successor of *fk*.

When a node *n* looks up a file, the node *n* first hashes the file to a group *fg* and a file key *fk*. If *fg* is the same as the node *n*'s group, it routes the query according to the regular Chord. Otherwise, the querying node forwards the query to one superpeer in its group. The query is then routed on the top-tier from the superpeer in the query source group to a superpeer in the destination group *fg*. The query routing on the top-tier can be carried out in different ways depending on the performance requirement. The basic routing is clockwise unidirectional routing. In this method, each group on the query path routes the query to a group (close or distant group) that has the smallest clockwise distance from *fg*. The second method is bidirectional routing. In this routing, in addtion to the regular distant neighbor groups, each group also treats the groups having long links to it as its distant neighbor groups for query forwarding. A query is forwarded to a neighobr group (close or distant) that has the smallest absolute distance from the destination group. A particular query may be forwarded clockwise or counter-clockwise along the group ID ring. To further reduce the query latency, the lookahead approach can be combined with bidirectional routing. In this approach, each group knows the group ids of the neighbors of its own neighbor groups (close of distant). Each group forwards the query to its neighbor whose neighbor has the smallest absolute distance from the destination group.

4. Analysis and experimental result

Because the major improvement of our appraoch is on the top-tier, this section will focus on the performance analysis of the top-tier overlay. We evaluate our small-world top-tier overlay against the completedly-connected graph top-tier overlay and the chord-ring top-tier overlay. The routing performance is in terms of the routing latency in the number of overlay hops and the routing state in the number of neighbors per node. Our small-world top-tier overlay has the minimum routing state: $O(1)$ (*i.e. 2k+2, k: a constant*). Chord top-tier overlay has the routing state: $O(\log^n)$. The completedly-connected graph top-tier overlay has the routing state: $O(n)$. Smaller routing state means less number of TCP connections on the underlaying network. It also means that when one node joins or leaves, less number of of other nodes are affected.

As for the routing latency, our approach has $O(\log^2 n)$ while Chord has $O(\log^n)$ and the completedly-connected graph has $O(1)$. At the surface, $O(\log^2 n)$ may seem much larger than $O(\log^n)$ with the same number of nodes *n*. However, the experimental result shows that with a good *k* value (number of long-links per node) , the bidirectional routing combined with lookahead can have routing latency comparable to Chord. Figure 2 displays the routing latency of Chord and the small-world overlay with different *k* ($k \in [1,4]$). The horizontal axis is the number of nodes on the overlay. The vertical axis is the average number of overlay hops per query. The figure clearly indicates that with a small number (3 or 4) of long

links, the small world overlay can be as good as Chord. It is also observed that more long links does reduce the routing latency but the reduction slows down when k reaches some value. Therefore, too many long links may not be good.

5. Conclusion

In this paper, we propose a new P2P overlay hierarchy consisting of two tiers. The top-tier is a small-world overlay. The bottom tier is many Chord rings. All nodes are divided into independent groups. Peers in the same group form one Chord ring. A small number of peers in each group is selected as superpeers for that group. All superpeers form the top-tier overlay. On the top-tier, each group has a group ID. Group IDs form an ID ring. Each group has short links to its immediate predecessor group and its immediate successor group. Each group also has long links to k distant groups that are probabilistically selected. Our small-world top-tier overlay achieves $O(\log^2 n)$ routing latency with only a small number of TCP connections.

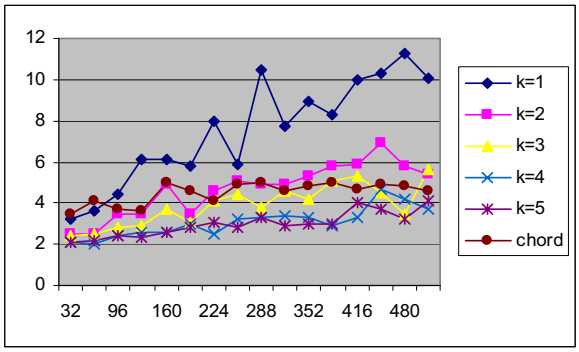

Figure 2: Performance comparison of chord and small-world top-tier overlay

References

Stoica, I., Morris, R., Karger, D., Kaashoek, M., and Balakrishnan, H., "Chord: a scalable peer-to-peer lookup service for internet applications", *Proceedings of ACM SIGCOMM*, 2001

Ratnasamy, S., Handley, M., Karp, R., and Shenker, S., "A scalable content-addressable network", *Proceeding of ACM SIGCOMM*, 2001

Rowstron, A., Druschel, P., "Pastry: scalable, distributed object location and routing for large-scale peer-to-peer systems", *Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, 2001

Zhao, B.Y., Kubiatowicz, J., and Joseph, A.D., "Tapestry: an infrastructure for fault-tolerant wide-area location and routing", Technical report, University of California, Berkeley, 2001

Manku, S.G., Bawa, M., Raghavan, P., "Symphony: distributed hashing in a small world", *Proceeding of 4th USENIX Symposium on Internet Technology and Systems (USITS'03)*, 2003

Gupta,I., Birman,K., Linga,P., Demers, A., Renesse, R.V., "Kelips: building an efficient and stable P2P DHT through increased memory and background overhead", *Proceeding of IPTPS*, 2003

Mizrak, A.T., Cheng,Y., Kumar, V., and Savage,S., "Structured superpeers: leveraging heterogeneity to provide constant-time lookup", *Proceeding of IEEE Workshop on Internet Applications,* 2003

Garces-Erice, L., Biersack, E.W., Felber, P.A., Ross, K.W. and Urvoy-Keller, G., "Hierarchical peer-to-peer systems", *Proceeding of INFOCOM'03*, 2003

Freedman, M.J., Mazieres, D., "Sloppy hashing and self-organizing clusters", *Proceeding of IPTPS'03*, 2003

Xu, Z., Min R., Hu, Y., "HIERAS: A DHT based hierarchical P2P routing algorithm", *Proceeding of International Conference on Parallel Processing*, 2003.