HP2P: A Hybrid Hierarchical P2P Network

Zhuo Peng*, Zhenhua Duan*, Jian-Jun Qi*, Yang Cao* and Ertao Lv*
*Institute of Computing Theory and Technology
Xidian University, Xi'an, 710071, P.R.China,
Email: {zhhduan,etlv}@mail.xidian.edu.cn

***Abstract*— Unstructured and structured P2P are two typical distributed models for P2P networking. The unstructured P2P uses flooding method, and has poor scalability and low efficiency, while the structured P2P uses DHT (Distributed Hash Table) method, and has low stability. This paper proposes a two-layer hybrid P2P network - HP2P, which combines flooding and DHT methods: Chord is used for the upper layer and flooding for the lower layer. The general framework of HP2P is presented and its upper and lower layers are described. By analyzing the performance of HP2P, it shows that HP2P has well scalability, high efficiency, and good stability.**

I. Introduction

P2P (Peer-to-Peer) networking is an important model that extends the limits of the traditional client/server architecture. In the P2P network, each node (or peer) is equivalent in functionality and capacity. They play both roles of client and server so called "servent". With its scalability, load balancing, self-organization, adaptation and fault tolerance, P2P applications enjoy better performance than traditional applications. The typical topologies of P2P networks are three types: central server P2P, unstructured P2P and structured P2P such as Napster [1], Gnutella [2] and Chord [3]. The unstructured P2P and structured P2P are both purely distributed topology and most of research work are based on them.

The unstructured P2P network is based on random graph and uses flooding [4], so it is called "unstructured". With the basic flooding method, queries are sent to all neighbors and then forwarded. The unstructured P2P model is scalable to nodes joining and leaving the system, which is called stability. However, the current searching mechanisms are unscalable and inefficient because of flooding, generating heavy loads on the network participants. The unstructured P2P network has been implemented in several systems including Gnutella, KaZaA [5] and Freenet [6]. Gnutella uses breadth-first traversal (BFS) with depth limit D (called controlled flooding or light flooding), where D is the system-wide maximum time-to-live (TTL) of a message in hops. Each node receiving a query will forward the message to its neighbors, unless the message has already travelled D hops [7] (reached the TTL value). In an N nodes unstructured P2P network, the typical time complexity of searching is $O(N)$. The unstructured network is good at system stability but needs to traverse all nodes in the system. This is a waste of bandwidth and may cause broadcasting storm.

The structured P2P network is based on DHT (Distributed Hash Table) [8]. By "structured" we mean that the P2P overlay topology is structured and controlled in a way in which resources are placed at specified locations. DHT is a large hash table which is maintained by wide area distributed nodes. Further, the large hash table is divided into many subtables and each node maintains a subtable. In the DHT, each item has a key and a value. The key for a data resource is assigned to the live node whose identifier is "closest" to the key (according to some metric). The lookup service determines the node that is responsible for a given key. This method is more efficient than flooding and has a better scalability. The structured P2P network has been implemented in a number of systems including Chord, CAN [9], Pastry [10] and Tapestry [11]. In these DHT based networks, overlays are highly structured and hashing is used for data placement. Each node in the structured P2P network has a routing table which is used for data searching. In an N nodes structured P2P network, the typical time complexity of searching is $O(\log N)$. So the advantage of structured network is its efficiency in searching. However, it is more frail when a large number of nodes continuously join and leave the system since this may cause routing table error and then system failure.

TABLE I
CONTRAST BETWEEN CHORD AND FLOODING.

The amount of nodes is N, each node has d neighbors.

	Chord	flooding
Time Complexity	$O(\log N)$	$O(N)$
Space Complexity	$O(\log N)$	$O(d)$
Stability	Low	High
Load Balancing	Yes	No
Scalability	High	Medium

The advantages and disadvantages of unstructured and structured P2P networks are shown in Table I.

Both of the unstructured and structured P2P networks are flat networks. This means that the architectures are not hierarchical. To improve the performance of P2P networks, hierarchical networks has been studied in the literatures [12], [13], [14]. However, most of researchers focus on theoretical aspect of system scalability and lookup latency. The existing hierarchical P2P networks are not good at system stability.

In this paper, we propose a hierarchical network, namely HP2P, which is a hybrid hierarchical P2P network compared to other ones. Our HP2P network combines DHT and flooding methods in a full distributed topology. Within the HP2P, Chord is used for the upper layer while flooding is employed for the lower layer. Further, with upper layer, each node is

Proceedings of the First International Conference on the Digital Society (ICDS'07)
0-7695-2760-4/07 $20.00 © 2007 IEEE

called a virtual node since it represents a group of nodes in the lower layer. With the lower layer, each group of nodes are organized by means of flooding approach. To easily communicate between two layers, supernodes are introduced. In contrast with other hierarchical P2P networks, our HP2P is better in a number of aspects such as stability, efficiency, and scalability.

This paper is organized as follows: Section 2 discusses the related work. Section 3 gives the general framework of HP2P. Section 4 describes the details of upper and lower layers of HP2P. Section 5 discusses the performances of the HP2P network. Finally, the conclusion is drawn in Section 6.

II. Related Work

KaZaA [5] is similar to Gnutella except for some additional supernodes in the network which manage a group of nodes. Supernodes are connected through a top level overlay, so KaZaA can be recognized as hierarchical unstructured P2P network. [15] proposes a model called RLP2P which is a multi-layer unstructured model based on region. In these networks, searching efficiency is the most serious problem because of flooding.

Hierarchical schemes presented in those such as [12], [13], [14], [16] and Canon [17] are all structured DHT hierarchical networks. In [13], authors propose a generic framework for the hierarchical organization of P2P overlay network, and demonstrate the various advantages it offers over a flat organization. In [17], authors describe Canon, a general technique for constructing hierarchically structured DHT, and demonstrate the advantages of hierarchical DHT construction, such as routing in terms of fault isolation, bandwidth utilization, adaptation to the physical network etc.

The research works above on hierarchical DHT in general aim at improving overall system scalability and lookup latency. However, popular applications are all based on central server P2P and unstructured network (or some improved networks). The structured networks are not used widely in applications because of the low stability.

Therefore, we are motivated to create a general framework which combines the advantages of both unstructured and structured P2P networks to improve system stability and scalability but still with low lookup latency.

III. The HP2P Framework

A. Basic Description

First, we introduce some notations. *Cluster (CL)* denotes a group of nodes, in which nodes are aggregated by some strategies. In our network, the nodes are aggregated according to geography location proximity. *Virtual node (VN)* is a node in the DHT and does not exist actually. It represents the Cluster indeed. *Supernode (SN)* is a node in the Cluster, which acts as an organizer and transmitter of the Cluster.

Nodes in the HP2P network are first organized in the Cluster in which messages are forwarded by plain flooding (or other modified flooding methods such as Directed BFS [7], Intelligent BFS [18], Expanding Ring [4] etc.). Each Cluster has some supernodes which manage the Cluster. The Cluster does not disappear if there are nodes in it. So it functions stably for a long time.

The Clusters are organized in DHT manner (here we use the typical and simple protocol Chord). Each Cluster is a virtual node in the Chord and has a routing table. The actual executors of virtual nodes are supernodes in the Cluster. Communications among and in the Clusters are forwarded by the supernodes. The hybrid hierarchical P2P network is shown in Fig. 1.

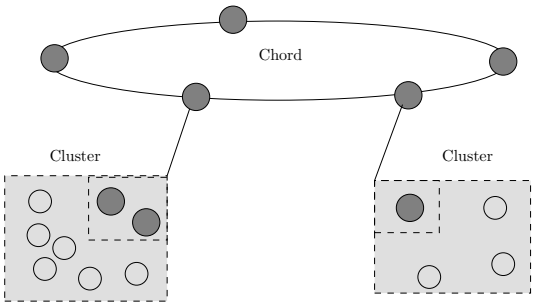

Fig. 1. The HP2P framework.

B. Lookup Service

Routing in the HP2P network is different from other networks. When a node searches, queries are first sent to the Chord to locate the Cluster which stores the data, then queries are forwarded in the Cluster. However, flooding searching can be performed within the Cluster if a node wants to get neighbors' resources.

The detailed searching process of HP2P is as follows (see Fig. 2):

1) A node $P_i \in CL_i$ sends the query to $SN_i \in CL_i$.
2) SN_i finds the successor (a Cluster CL_j) through the Chord.
3) SN_i forwards the query to the successor CL_j.
4) In CL_j, its supernode SN_j floods the query.
5) When found, the node $P_j \in CL_j$ return a successful response.

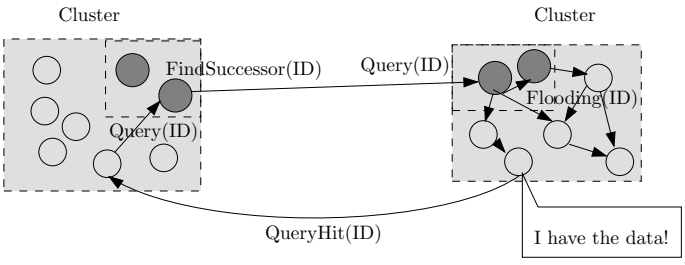

Fig. 2. Information searching process.

IV. The Upper and Lower Layers

The HP2P network combines both unstructured and structured P2P networks. The upper structured layer is based on Chord (or other DHTs) and the lower unstructured layer is based on Cluster (using flooding).

A. The Upper Chord[19]

The upper layer can be any kind of DHTs. We choose Chord because it is a typical protocol and simple enough for applications. Chord is a scalable peer-to-peer lookup service for internet applications. Given a key, Chord can find out the node responsible for storing the key's value which is "closest" to the key. Each node in the Chord maintains a routing table called a finger table [3]. In our network, Chord is used for upper layer routing and also for organizing the Clusters.

We use most of the characters of the Chord but modify the finger table to support the Cluster. Meanwhile, data are not stored in this layer anymore. The successor field of the finger table is also modified to point to a Cluster (virtual node). Further, a supernode table, which records the virtual nodes, is added according to the alternation of the finger table. In the upper Chord, the Cluster appears as a virtual node and the nodes in the supernode table together act as a virtual node (see Fig. 3).

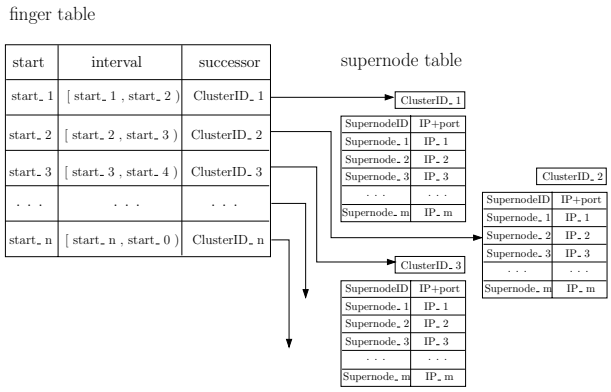

Fig. 3. Finger table and supernode table.

So the modified joining process of virtual node is as follows:

1) A new virtual node VN_i initiates its $finger[1].start$ field of finger table and its corresponding supernode table.
2) VN_i sends locating message to the existing VN_m to get the successor of $finger[1].start$ and its supernode table.
3) VN_i gets the predecessor field of its successor as its predecessor, then updates the predecessor field of its successor to itself.
4) VN_i updates other $start$ fields.
5) VN_i updates other virtual nodes' finger table and supernode table.
6) VN_i sends message to its successor VN_j to delete the metadata.

The modified leaving process is as follows:

1) A virtual node VN_i deletes all the metadata it maintains.
2) VN_i informs the new incomers(both metadata and nodes) to locate its successor VN_j to join.
3) VN_i informs other virtual nodes to leave and updates their finger tables. Further, VN_i updates $finger[i].successor$ to corresponding supernode table.
4) VN_i leaves.

The stabilization process is given below (acting on the entry i):

- finger table stabilization
  1) A virtual node VN_i finds CL_i which is pointed by the entry i. If all the supernodes in CL_i are failed, the action moves to the next step; if not, VN_i turns to supernode table updating process.
  2) VN_i updates the finger table and finds the successor of $start[i]$, then substitutes the old supernode table. Finally, the action turns to entry $i+1$.
- supernode table stabilization
  1) If the number of the supernodes is greater than k (threshold of supernodes), the action turns to entry $i+1$; if not, the action moves to the next step.
  2) VN_i gets supernode list, then updates the supernode table.
- finger table checking
  1) successor checking
     - VN_i checks whether the predecessor of its successor is itself.
     - If this predecessor is greater than itself, then VN_i updates its successor to this value; if not, VN_i updates its successor's predecessor to itself.
  2) predecessor checking
     - VN_i checks whether the successor of its predecessor is itself.
     - If the successor is less than itself, then VN_i updates its predecessor to this value; if not, VN_i updates its predecessor's successor to itself.
  3) random checking
     - VN_i finds a successor in a random entry.
     - If the value is less than its successor, then VN_i updates its successor to this value.

Other processes such as failure, notify, and so on are modified too. We omit them here. The general Chord operations are shown in Fig. 4.

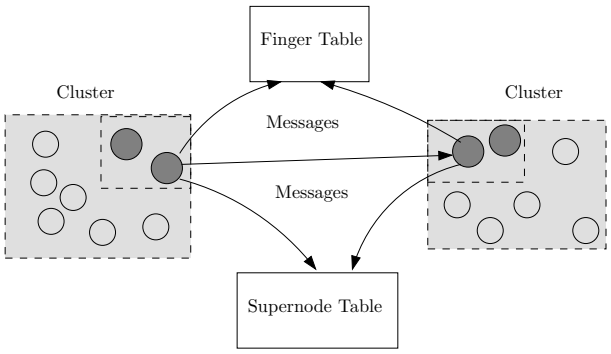

Fig. 4. Chord operations.

B. The Lower Cluster

Our Cluster is a small unstructured P2P network which is similar to Gnutella or KaZaA. Each Cluster has a Cluster ID

(identity) which is determined by the first node (using some hash functions). Nodes join the Cluster according to geography location proximity. Metadata are placed in their successors (Clusters) through Chord protocol.

The size of the Cluster is predefined and m denotes the size of the Cluster. When a peer joins a full Cluster (has m nodes already), the Cluster checks if it has been split. If not, it will cause the splitting of the Cluster and the peer joins in a new Cluster, otherwise the peer will join in the Cluster that has already been split from the original Cluster. The parameter m is determined by the capability of the Cluster supernodes in applications.

Supernode is elected from ordinary nodes, and is estimated as follows:

- Is under NAT/firewall or not.
- Online time.
- Bandwidth.
- Speed of CPU.
- The size of memory.
- The number of TCP connections supported

Most of massages are flooded in the Cluster except that directed massages are sent to supernodes. Supernodes play the role of interactions between the upper and the lower layers. They function both in the Chord and Cluster. The supernodes, as virtual nodes, perform the Chord protocol in the Chord while execute the management characters in the Cluster.

Each node has two roles in the Cluster. On one hand, acting as a *Metadata node*, it stores the metadata of resources. On the other hand, acting as a *Resources node*, it provides with the resources to the network.

In the traditional unstructured networks, there are no metadata because queries are sent directly towards resources. In our Clusters, metadata and resources are stored separately. The metadata are stored in the Cluster where their hashing values are less than or equal to the Cluster ID but greater than the predecessor's Cluster ID.

We cannot find resources if the corresponding metadata are not online. To improve fault tolerance, the redundance of metadata is required. Each piece of metadata is simply dispensed by b times in the Cluster. The resources nodes have to check their corresponding metadata nodes to keep b backups online. When the number of backups is less than b, resources nodes will dispense them again.

After joining the Clusters, a node can act as a ordinary node and is possibly promoted to a supernode in later actions. The general Cluster operations are shown in Fig. 5. The joining process is as follows:

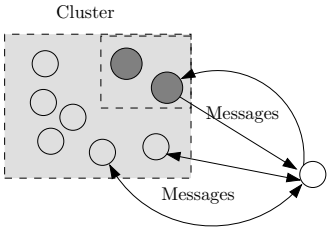

Fig. 5. Cluster operations.

- first joining
  1) A node P_i locates CL_i it belongs to. Then P_i gets the supernode list of CL_i.
  2) P_i requests $SN_i \in CL_i$ for joining.
  3) SN_i gets a random neighbor list in CL_i then sends it to P_i as its neighbor list.
  4) P_i informs all the nodes in the neighbor list to add it and update their statuses.
  5) P_i publishes its resources.
  6) Normal actions.
- usual joining
  1) P_i tries to join CL_i again. If failed, P_i relocates CL_j and turns to first joining process.
  2) P_i checks its resources and its metadata nodes.
  3) Normal actions.

Then, let's see the leaving process of nodes in the Cluster:

1) A node P_i informs $SN_i \in CL_i$ to leave.
2) P_i informs its neighbors to delete it from their neighbor list.
3) As a resources node, P_i informs its metadata nodes to modify the flags of metadata and add time stamps.
4) As a metadata node, P_i deletes all the metadata.
5) P_i leaves.

Other processes such as notify, failure, stabilization and so on are omitted here.

V. Performance

In the HP2P, we concern with Chord and Cluster. In the Chord, the operations such as joining, leaving and failure of virtual nodes as well as information publishing and searching are permitted. In the Cluster, the operations including joining, leaving and failure of nodes as well as information storing are legal.

Since the HP2P combines unstructured and structured methods together, it obtains some new characters.

a) Stability: In the upper layer, the Chord is more stable since the Clusters function for a longer time in contrast with the original Chord. This can reduce network churn of the Chord. Clusters do not disappear until all the nodes fail simultaneously. So the upper layer has a slight network churn by replacing normal peers in the Chord DHT by cluster of nodes which are less prone to disappear.

b) Scalability: Each Cluster is a small network and all Clusters together are a large network. With the hierarchical architecture, more nodes can be accommodated in the HP2P network compare to the pure Chord.

c) Reducing Loads: The metadata of resources are stored in the Clusters, so the Chord layer does not have metadata. Accordingly, it reduces the loads of storage and transferring in the Chord.

d) Light Flooding: In the lower layer, the Cluster is smaller than regular unstructured networks. So flooding is performed in a certain area, which can avoid the broadcasting storm in the whole network.

We now discuss time and space complexities. Suppose there are N nodes in total in the system, and each Cluster has m (m is not a large number) nodes in maximum. So there are at least $\frac{N}{m}$ Clusters in the system (it means $\frac{N}{m}$ virtual nodes in the Chord). Suppose each node has typical d neighbors in the Cluster. If N nodes are all in the Chord, the time and space complexities are $O(\log N)$ and $O(\log N)$. If N nodes are all in the Cluster, they are $O(N)$ and $O(d)$. In our network, the time and space complexities are $O(\log \frac{N}{m} + m)$ and $O(d)$.

We now focus on the lookup latency. In the P2P network, lookup latency is usually measured by hops which means the number of nodes reached by queries. The average lookup latency of the pure Chord network is $\log N$ hops. Before we discuss lookup latency of the unstructured network, we have the following conclusion[20].

Theorem 1: In an acyclic graph,

$$n_{nodes} \approx \sum_{i=1}^{n_{hops}} (d-1)^i$$

Where n_{nodes} is the number of nodes reachable, n_{hops} the number of hops, and d the average number of nodes reachable from each node (degree of connectivity).

Thus, the average lookup latency of the pure unstructured P2P network is about $\log_{d-1}(\frac{(d-2)\cdot N}{d-1} + 1)$ hops.

So we can get the average lookup latency of the HP2P network is about $\log \frac{N}{m}$ + $\log_{d-1}(\frac{(d-2)\cdot m}{d-1} + 1)$ hops.

It can be concluded that the average lookup latency of our network is less than the pure Chord (see Fig. 6) and the average finger table entries are much smaller than the pure Chord.

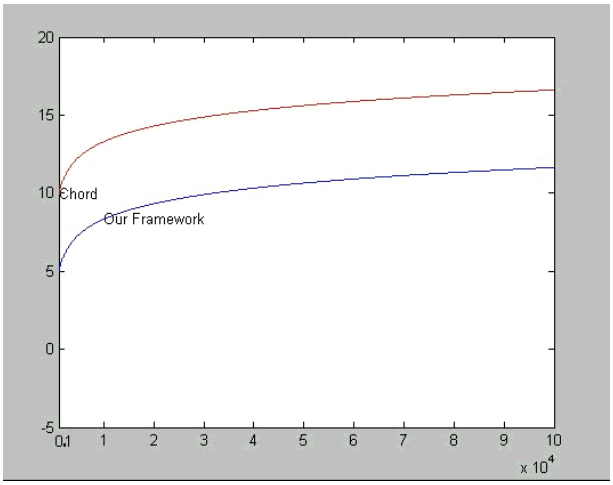

Fig. 6. Average lookup latency, where $m = 1000$, $d = 5$.

VI. Conclusion

In this paper, we proposed a hybrid hierarchical P2P network called HP2P, which combines unstructured and structured networks. The HP2P is better in system stability as well as scalability and efficiency compared to other networks. Some interesting characters are also introduced. However, several aspects such as namespace, detailed operations of both layers are omitted in this paper because of space limitation. There are still many mechanisms such as Cluster optimization, lookup cache in the upper and lower layers need to be improved in the future researches.

Acknowledgements

The authors gratefully acknowledge the support of NSFC Grant 60373103 and 60433010 and SRFDP Grant 20030701015.

References

[1] Napster. http://www.Napster.com/.
[2] Gnutella. http://www.Gnutella.com/.
[3] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan: Chord: A scalable peer-to-peer lookup service for internet applications. In Proc. ACM SIGCOMM, August 2001.
[4] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker: Search and Replication in Unstructured Peer-to-Peer Networks, in Proc. Of the ACM ICS, 2002.
[5] KaZaA. http://www.KaZaA.com/.
[6] Freenet. http://freenet.sourceforge.net/.
[7] B. Yang and H. Garcia-Molina: Improving Search in Peer-to-Peer Networks, in Proc. of the 22nd International Conference on Distributed Computing Systems (ICDCS'02), June 2002.
[8] W. Litwin, M.-A. Neimat, and D. A. LH Schneider: A Scalable, Distributed Data Structure. ACM Transactions on Database Systems (TODS) 21, 4 (1996), 480C525.
[9] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker: A scalable content-addressable network. In Proc. ACM SIGCOMM, August 2001.
[10] A. Rowstron and P. Druschel: Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In Proc. Middleware 2001, November 2001.
[11] B. Zhao, J. Kubiatowicz, and A. Joseph: Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, Computer Science Division, U. C. Berkeley, April 2001.
[12] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker: Topologically-aware overlay construction and server selection, in Proceedings of Infocom02, (NewYork City, NY), 2002.
[13] L. Garcés-Erice, E.W. Biersack, P.A. Felber, K.W. Ross, and G. Urvoy-Keller: Hierarchical Peer-to-Peer Systems, Euro-Par 2003, LNCS 2790, pp. 1230C1239, 2003.
[14] M. Castro, P. Druschel,Y. C. Hu, and A. Rowstron: Exploiting network proximity in peer-to-peer overlay networks, Tech. Rep. MSR-TR-2002-82, Microsoft Research, One Microsoft Way, Redmond,WA 98052, 2002.
[15] G.X. Yue, R.F. Li, and Z.D. Zhou: A P2P network model with multi-layer architecture based on region, Journal of Software, 2005,16(6):1140.1150. DOI: 10.1360/jos161140, 2005.
[16] G. Doyen, E. Nataf, and O. Festor: A Hierarchical Architecture for a Distributed Management of P2P Networks and Services, DSOM 2005, LNCS 3775, pp. 257C268, 2005.
[17] P. Ganesan, K. Gummadi, and H. Garcia-Molina: Canon in G Major: Designing DHTs with Hierarchical Structure, Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS04) 1063-6927/04, 2004.
[18] V. Kalogeraki, D. Gunopulos, and D. Zeinalipour-Yazti: A Local Search Mechanism for Peer-to-Peer Networks. In CIKM, 2002.
[19] Y. Cao, Z.H. Duan, J.J. Qi, Z. Peng and E.T. Lv: Implementing Chord for HP2P Network, RDDS 2006, LNCS 4278, pp. 1480-1489, 2006.
[20] M. Miller, Jozef Siran: Moore graphs and beyond: A survey of the degree/diameter problem, Mathematics Subject Classifications: 05C88, 05C89, 2005