A Multi-Level Super Peer Based P2P Architecture

Zhao Cao[#1], Kan Li[#2], Yushu Liu[#3]

[#] *School of Computer Science and Technology, Beijing Institute of Technology*
Beijing Institute of Technology Beijing, 100081, China
[1]zhaoyang@bit.edu.cn
[2]likan@bit.edu.cn
[3]liuyushu@bit.edu.cn

Abstract—**Most existing super peer based Peer-to-Peer (P2P) systems have only one level super peer, it can not support hierarchical P2P architecture, especially, the effectiveness, efficiency and scalability can not satisfy some application's requirements with the growth of the system. In this paper, we design a P2P system with hierarchical structure. First, we propose a Multi-Level Super Peer (MLSP) based P2P architecture for supporting hierarchical P2P system, where the upper level super peer summarizes its subordinates' meta information. Second, based on this architecture, we develop our query executing algorithm by only spreading query to the upper level super peer and flooding query between root level super peers. Finally, we design a protocol to handle the join and leave of peers, super peer selection, split and merge. By conducting experiments on a simulated large-scale network, we come to the conclusion that the effectiveness, efficiency and scalability improve prominently in the proposed system.**

I. INTRODUCTION

A central issue about the P2P system is assigning and locating resources among peers, which is called lookup services. There are three types of P2P lookup services implemented in P2P system: centralized directory model searching, unstructured searching and structured searching.

The centralized directory model was made popular by Napster[1]. It is based on a central directory server, where it publishes information about the content offered for sharing. The directory stores the meta information for searching. This model requires some management infrastructure (the directory server), which hosts information about all participants. This leads to the model to show some scalability limitations. Its biggest disadvantage is the single-point failure problem.

Unstructured P2P such as Gnutella[2], Freenet[3] use flooding mechanism to query data. Many popular P2P systems used in the internet are flooding systems. Each request from a peer is flooded (broadcast) directly to its connected peers, and these peers flood this request to their directly connected peers, until the request is answered or a maximum number of flooding steps is reached. Flooding is an expensive searching approach, thus many improved approaches are proposed, such as gossip flooding, cluster organization, ant algorithm[4][5], neural network[6], association analysis[7] and so on[8]. But the efficiency can not satisfy some large application, and the extensibility is not good enough in particular.

Structured P2P such as Chord, CAN[9], Pastry[10] and Tapestry[11] are designed for applications running on well-organized networks. It organizes peers to a flat overlay without topology-awareness. The searching process is completed in O(logN) hops. These systems usually ensure fast and efficient lookups, but do not support fuzzy queries.

Combining the advantage of the centralized directory model and the unstructured P2P system, many P2P systems use super peer based P2P architecture[12], which is a hybrid architecture, such as KaZaA[13], JXTA. There are two types of peers: the common peer which is called the "peer", and the "super peer". Every peer should connect to a super peer. The super peer acts as the centralized server in the central directory model system. Super peers connect to each other in the same way like in the pure P2P system.

In some recent applications, the one level super peer based P2P architecture can not satisfy the system's requirements. For example, the government data integration projects using P2P technology where the government data is distributed in a hierarchic manner, the peer in the superior department manages all the peers subordinate to it and only knows a summary of the data stored in its subordinates. A query from another department would need to be sent to the nearest common superior which manages both departments directly or indirectly. The superior just gives the request an authorization to query the corresponding peer in the appropriate department.

In this paper, we show how to change the current super peer based P2P architecture to our Multi-Level based (MLSP) P2P architecture. In Section II, we come up with a general framework for the design of multi-level system and show how to execute the query in this architecture. We demonstrate our approaches on how to maintain the P2P network and construct the multi-level super peers as outlined in Section III. In Section IV, we evaluate the effectiveness, efficiency and scalability of our architecture. In Section V, we conclude the paper with a summary and a brief discussion of future work.

II. MULTI-LEVEL SUPER PEER BASED P2P ARCHITECTURE

A. *MLSP Based P2P Architecture*

The architecture consists of several trees. All trees' root peers connect to each other and form an overlay network. Similar to many other super peer based P2P architectures, it contains two types of peers: super peers and common peers (often called just "peers"). But in our architecture we label the super peers in different level.

Definition 1: A multi-level super peer based P2P architecture is a triple $T = (F, R, E)$, where F is a set of trees, R is the set of all trees' roots and E is the connection relation between all these root peers.

There are bidirectional connections (v, w) for every pair of peers v and w for which there is an edge $(v, w) \in E$ with $v \in R$ and $w \in R$.

The level number of root peer is 1. Level number increases as the depth of tree increases. Lower level super peers are children of upper level super peer. Every peer has a super peer to manage it and every super peer manages at least one children peer.

We define the grand super peer **grandSup(v)** of peer v as the super peer of v's super peer. For every peer v, it stores its super peer and grand super peer; for root level super peers, they do not have super peer and grand super peer, but they have neighbours, this information can help the system to reconstruct itself in case some peers leave the system accidentally.

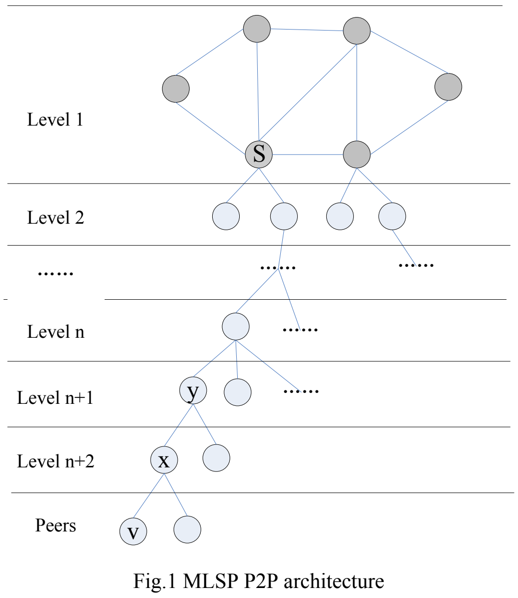

Fig.1 MLSP P2P architecture

As illustrated in Fig.1, the peers filled with gray color are root level super peers. Peer x is the super peer of v, and peer y is the grand super peer of v. The gray peers connected to peer S are neighbours of peer S.

Common peers aggregate into a group and connect to a super peer, this super peer is the lowest level super peer. With the growth of the system scale, the number of super peers grows gradually, when the number reaches a threshold, it cluster the lowest super peers into a small number of groups, and the super peers clustered into one group connect to an elected super peer as an upper level super peer. This process iterates with the scale-up of the system.

In the Fig.2, the blank peers are common peers, the peers filled with black color are super peers, the black peers not contained in the big circle are lowest super peers, the peers between the big circle and the little circle filled with gray color are second-level super peers. In the system, there are three levels of super peers, so the third-level super peers in the circle filled with gray color are the root super peers of the system.

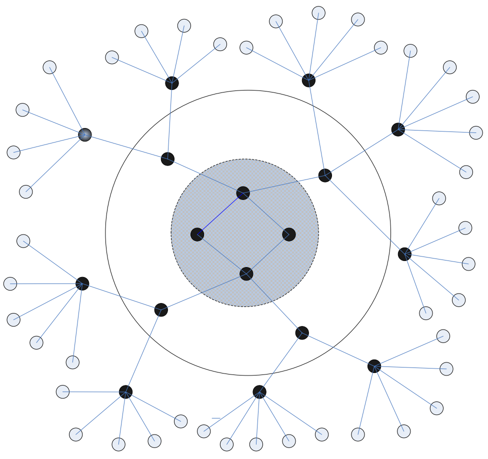
Fig.2 Example of three levels super peer architecture

B. *Query processing*

Query from a peer is first submitted to the super peer it connects to, if the super peer can not answer the query, the query is sent to the upper level super peers iteratively until reach the root level. In the root level, the query is flooded as in a pure P2P system.

The algorithm for query executing is described below:

```
Algorithm: executeQuery(q, p)
1  if(p can answer q)
2      It is a local query, do it in the local peer
3  else
4  {
5          while( sup(p1) != null)
6          {
7              p1 = sup(p1)
8              send q to p1
9               if(p1 can answer q)
10              {
11                  Send result to p
12                  return
13              }
14         }
15         Set s = neighbours(p1)
16         for(every peer n in s)
17         {
18              send q to n
19               if (n can answer q)
20               {
21                  send result to p
22                  return
23                }
24         }
25    }
```

In this algorithm, sup(p1) means getting the super peer of peer p1, neighbours(p1) means getting all the neighbours of p1.

III. Network Construction and Maintenance

A. Peers join the system

If there is a new peer N applies to participate in the system, it first sends an advertisement containing the peer description to the system, if an existing super peer receive this advertisement, it computes the distance between peer N and itself based on the meta information stored in this super peer and sends this distance to the newly joined peer.

This decision-making function varies in different applications. For example, in file sharing system, the description of a peer is a set of keywords representing the files shared by this peer, given all these keywords, the super peer can easily compute the distance between two keywords vectors.

The newly joined peer N receives several responses from existing super peers; peer N selects a super peer as its entry point to the system. This super peer selection varies in different systems; it is decided by the super peer distribution criteria and peer-to-peer factor in [14]. In our system, we add the semantic distance to the selection criteria; all these criteria and factors have their weight in the selection decision-making function.

$$\max\{\sum_{i=1}^{n} w_i f_{ij} \quad ,1 \le j \le m\}$$

Where n is the number of the factors, f_{ij} is the value of the factor i received from super peer j, w_i is the weight of factor i, and m is the number of responses received.

If a peer N that used to participate in the system reconnects to the system, it first consults its local cache for super peer candidates (every participated peer stores a super peer list). If there are no suitable candidates in the cache, it selects a super peer as new entry point.

A common peer can check the trust level information in the certificate presented by a super peer and can also choose a super peer based on trust-based routing criteria by evaluating the trust level of the peers along the path to the potential super peer. For fault tolerance, a given common peer can connect to several super peers, verifying whether its connections to those super peers traverse disjoint paths.

B. Peers leave the system

1) Common peer leaves the system: If a common peer S leaves the system, its super peer sets the peer S's status to be inactive; if this peer does not rejoin the system for a long time, the super peer will delete the meta information of S from the super peer, and notify the upper level super peer to delete the meta information until reaching the root level super peer.

2) Super peer leaves the system: If a super peer S leaves the system, it is much more complex than a common peer leaves, because it must select an existing super peer to manage the children of S or elect a super peer from the common peers; then construct the meta information of S's children.

If a super peer S leaves the system unexpectedly, the information stored in S will be entirely lost. In this case, we should select/elect a super peer for the children of S. If we have selected/elected a super peer S1 as the alternative super peer, now, S2 which is the grand super peer of S informs its grandchild (children of peer S) to connect to the newly selected/elected super peer S1 and sends all of its descriptions to S1. Then, the children of S update theirs super peer information and S2 update its children information.

If a super peer S leaves the system on its own initiative, S will run the super peer selection/election algorithm to select/elect a new super peer S1, transform all the meta information stored in it to S1, and update the children and parent peer meta information of S as described previously.

C. Super peer maintenance

During the running of the system, peers join and leave the system frequently, which results in that some super peers manage very large number of common peers, but the others manage few common peers; as a result, some peers are very busy and response slowly while the others are light loaded. To make the system load balance, we split the weight loaded super peer into two super peers, and merge some light load super peers into one super peer.

1) Split super peer: If the depth of current tree is less than the system's specified depth, we add two children S1 and S2 in S, and split the children of S into two sections according to the factors mentioned previously, S1 and S2 manage one section respectively.

If the depth of current tree is larger than the system specified depth, we select a new super peer S1, and pass some children of S to S1.

The algorithm for super peer splitting is described below:

```
Algorithm: splitSuperPeer(S)
1    Cs = children(S)
2    if(the depth of the system> system's specified depth)
3    {
4        Sp = electSuperPeer(Cs)
5        Set Cs1 = split(Cs-Sp)
6        manage(Sp,Cs1)
8    Else
9    {
10       Sp1 = electSupePeer(Cs)
11       Sp2 = electSuperPeer(Cs-Sp1)
12       manage(S,Sp1+Sp2)
13       Set Cs1 = spilt(Cs-Sp1-Sp2)
14       Set Cs2 = Cs-Sp1-Sp2-Cs1
15       manage(Sp1,Cs1)
16       manage(Sp2,Cs2)
17   }
```

In this algorithm, children(S) means getting all the peers managed by S; electSuperPeer(Cs) means electing a super peer from peer set Cs; Split(Cs) means splitting a section of peers from set Cs, it returns a peer set; manage(Sp,Cs1) means setting the peer set Cs1's super peer as Sp, saving the meta information of Cs1 to Sp and spreading all these meta information to Sp's super peer iteratively until reaching the root level super peer.

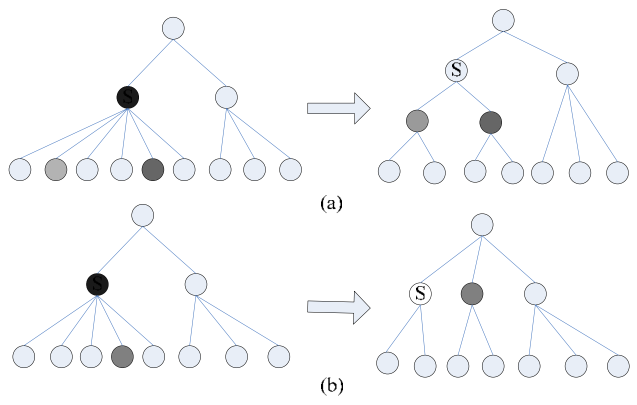

Fig.3 Split of super peer

As described in Fig.3 (a), the peer S filled with black color is heavy loaded, so we select two super peers (peers filled with light gray and deep gray color) from its children as super peers to manage the rest of peers respectively. This increases the depth of the system. Another case illustrated in Fig.3 (b) is that we select a peer from the children of S as super peer, which manages a portion of peers managed by peer S previously.

2) Merge super peers: If super peer S1 and S2 manage only a few peers, we can merge peer S1 and S2 into one super peer. There are two cases: decrease the depth of the system and keep the depth of the system.

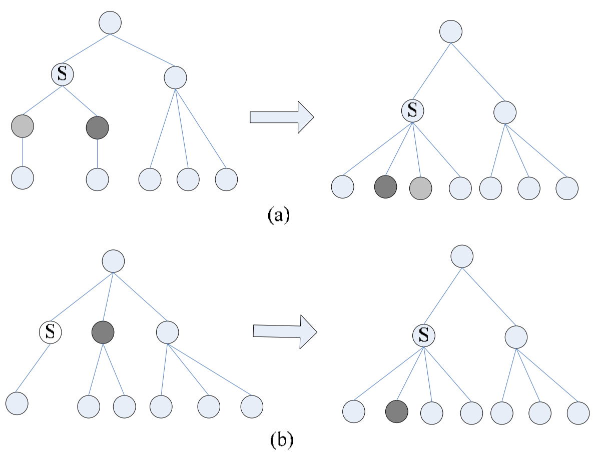

Fig.4 Merge of super peers

If the depth of current system is larger than the system specified, then we degrade the light loaded super peers to lower level or, in extreme cases, convert them to common peers. As depicted in Fig.4 (a), we degrade the super peer filled with light gray and deep gray color to the common peer level, the meta information stored in these two peers is transformed to peer S.

If the depth of current system is less than the system specified, then we keep the system depth, degrade one light loaded super peer to the lower level and transform the meta information stored in this peer to another super peer. As depicted in Fig.4 (b), the peer filled with gray color is degraded and transform its meta information to peer S.

The algorithm for merging super peers is described below:

```
Algorithm: MergeSuperPeer(p1,p2)
1    If(the depth of the system> system's specified depth)
2    {
3          Sp = sup(p1&p2)
4          Set Cs1 = children(p1)
5          Set Cs2 = children(p2)
6           Manage(Sp,p1+p2+Cs1+Cs2)
7    }
8    Else
9    {
10          Sp1 = selectSuperPeer(p1)
11          Set Cs1 = children(p1)
12          Manage(Sp1,Cs1+p1)
13          Sp2 = selectSuperPeer(p2)
14          Set Cs2 = children(p2)
15          Manage(Sp2,Cs2+p2)
16   }
```

IV. Experimental Evaluations

A preliminary simulation was performed comparing multi-level super peer based P2P mode with other P2P architecture. Each peer in a 10000 peers network was randomly assigned 3 documents from a pool of 10000. Each document was randomly assigned 3 keywords from a pool of 1000. The initial TTL was set to 16 and the forwarding subset M was set to be 2. The children number managed by each super peer is 5. The simulation runs for 100,000 searches, with each search being started at a randomly selected peer.

In the simulation, if the number of levels is zero, it is a pure P2P architecture; if the number of levels is one, the system is in a one level super peer based P2P architecture, which is the same as the architecture in [11]; if the number of levels is larger than one, it is a multi-level super peer based P2P architecture we proposed.

We examine the correctness and effectiveness of our architecture and query executing algorithm.

Correctness is usually measured in terms of the classic information retrieval notions of precision and recall. In our system, it is exact keywords matching, so the precision is 1, but the recall is different in different level settings.

Effectiveness is usually measured in average TTL of first match and average number of message transferred for each query. The less of average TTL of first match make the user get the portion of result faster. The decrease of average message transferred for each query can improve the traffic efficiency of the system and decrease the overhead of the network.

Fig.5 depicts that the recall increases quickly with the increasing of super peer levels. Especially, when the number of levels increased from 2 to 4 the recall increases greatly. In our simulation settings, when the super peer level reaches 5, it is obviously that the root level has only one peer, so the recall is 1.

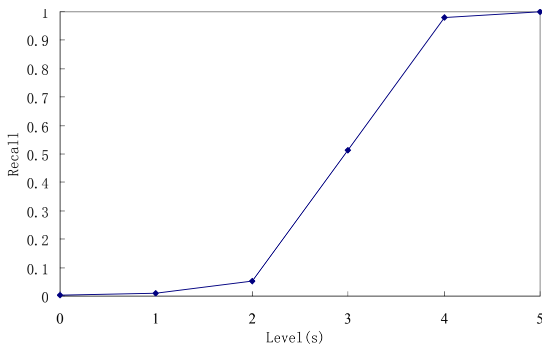

Fig.5 Level effect on recall

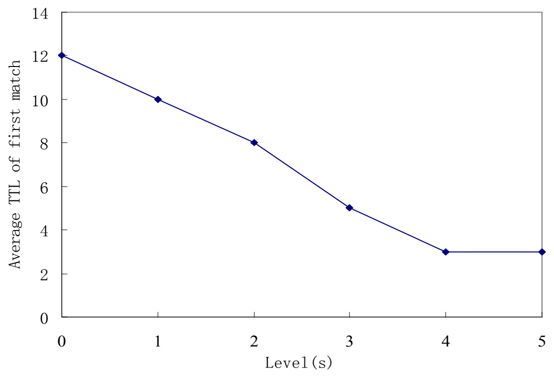

Fig.6 Level effect on average TTL of first match

Fig.6 depicts that the average TTL of first match decreases quickly with the increase of super peer levels, especially between 2 and 4.

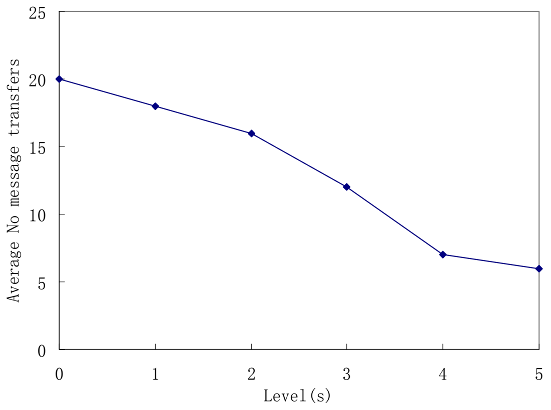

Fig.7 Level effect on average No message transfers

Fig.7 shows that the architecture decreases the number of message transferred with the increase of the super peer levels. This result in the query latency decreases prominently.

As showed by the experiments results, it is obviously that we improve the recall and efficiency of the searching, which make the system more scalable.

V. Conclusions

This paper proposed a multi-level super peer (MLSP) based P2P architecture. We have presented the query executing approach, network construction and maintenance method in this architecture. A prototype and a simulated large-scale network have been designed to evaluate the system performance. Our experiments showed that such an architecture and algorithms improve the query efficiency, effectiveness and scalability of the system prominently. It can also be applied to many applications with hierarchical structure.

In future work, we will add learning mechanism to the architecture, this would flood query message to the peer that contains the information most possibly, and avoid doing the same query repeatedly.

Acknowledgment

We would like to thank the anonymous reviewers for their valuable comments and suggestions. I am grateful to Robert Zotti, Sun Pei, Song Fei and Zhou Shibin's helpful discussion and advice.

References

[1] Napster homepage. http://www.napster.com/.

[2] Gnutella homepage. http://gnutella.wego.com/.

[3] Freenet homepage. http://freenet.sourceforge.net/.

[4] Kwang Mong Sim and Weng Hong Sun. "Ant Colony Optimization for Routing and Load-Balancing: Survey and New Directions," *IEEE Transactions on systems, man, and cybernetics*, vol.33, pp.560-572, Sep. 2003.

[5] Ozalp Babaoglu, Hein Meling and Alberto Montresor. "Anthill:a framework for the development of agent-based peer-to-peer systems." In *Proceedings of the 22nd International Conference on Distributed Computing Systems,* 2002, pp.15-22.

[6] Taskin Kocak, Jude Seeber and Hakan Terzioglu. "Design and implementation of a random neural network routing engine". *IEEE transaction on neural networks*. vol 14, pp.1128 – 1143, Sept. 2003.

[7] Brian D. Connelly, Christopher W. Bowron., Li Xiao, Pang-Ning Tan, and Chen Wang. "Adaptively Routing P2P Queries Using Association Analysis." In *Proceedings of the 2006 International Conference on Parallel Processing*, 2006, pp.281-288.

[8] B. Yang and H. Garcia-Molina, "Improving Efficiency of Peer-to-Peer Search," In *Proceedings of 28th International Conference on Distributed Computing Systems*, 2002.

[9] S.Ratnasamy, P.Francis, e.t.al. "A scalable content addressable network." In *Proceedings of the 2001 Conference on applications, technologies, architectures, and protocols for computer communications*, 2001.

[10] A. Rowstron and P. Drusche, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems." *International conference on distributed systems*, 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] B. Yang and H. Garcia-Molina, "Designing a Super-Peer Network," In *Proceedings of International Conference on Data Engineering (ICDE)*, 2003, pp.49-60.

[13] KaZaA. http://www.kazaa.com.

[14] Virginia Lo, D. Z., et.al. "Scalable Super node Selection in Peer-to-Peer Overlay Networks." In *Proceedings of the 2005 Second International Workshop on Hot Topics in Peer-to-Peer Systems*, 2005.