

Dynamic Layer Management in Super-peer Architectures

Zhenyun Zhuang, Yunhao Liu, Li Xiao

*Department of Computer Science and Engineering,
Michigan State University, East Lansing, MI 48824, USA
{zhuangz1, liuyunha, lxiao}@cse.msu.edu*

Abstract

The emerging peer-to-peer (P2P) model has recently gained a significant attention due to its high potential of sharing various resources among networked users. Super-peer unstructured P2P systems have been found very effective by dividing the peers into two layers, super-layer and leaf-layer, in which message flooding is only conducted among super-layer. However, current super-peer systems do not employ any effective layer management schemes, which means the transient and low-capacity peers are allowed to act as super-peers. Moreover, the lack of an appropriate size ratio maintenance mechanism on super-layer to leaf-layer makes the system's search performance far from being optimal. We propose a Dynamic Layer Management algorithm, DLM, which can maintain the optimal layer size ratio, and adaptively adjust peers between super-layer and leaf-layer. DLM is completely distributed in the sense that each peer decides to be a super-peer or a leaf peer independently without the global knowledge. DLM could effectively help a super-peer P2P system maintain the optimal layer size ratio, and designate peers with relatively long lifetime and large capacities as super-peers, and the peers with short lifetime and low capacities as leaf-peers under highly dynamic network situations. We demonstrate that the quality of a super-peer system is significantly improved under DLM scheme by comprehensive simulations.

1. Introduction

Since the development of Napster, millions of peers join the network by connecting to at least one active peer in the peer-to-peer overlay network. Each peer acts as both a client who requests information and services, and a server who produces and/or provides information and services.

This work was partially supported by the US National Science Foundation (NSF) under grant ACI-0325760, by Michigan State University IRGP Grant 41114.

Today, both unstructured P2P systems, such as Gnutella [5], and structured P2P systems, such as Chord [16] and Cycloid [14], are under intensive study [3] [4] [7] [19]. While unstructured P2P systems are most commonly used in today's Internet, an early generation is *pure* unstructured P2P system. Gnutella is an example of pure P2P systems, where all the peers are involved in query flooding process. In these systems, any peer could be a query source issuing queries to its neighbors, and the peers receiving the queries may check its local storage, and response or further relay queries to their neighbors.

However, peers can be very different from each other in bandwidth, CPU power, duration times, shared files, and interests [6]. In pure P2P systems, all peers, regardless of their capacities, act equal roles and take the same responsibilities on all the operations. As the network size increases, the weak peers will seriously limit the scalability of P2P systems.

To address this problem, super-peer architectures were proposed, which have been attracting more and more users in unstructured P2P community. For example, KaZaA and early Morpheus, based on FastTrack P2P stack, have dominated the top downloads lists for most of 2001 and are still increasing in popularity in 2002 [6] [10]. Schemes are also proposed to introduce Ultra-peers into Gnutella protocol [2].

The advantages of super-peer systems are well discussed in [17]. However, current super-peer systems do not exploit any effective layer management schemes; thereby leave the following two problems unsolved. First, assume an optimal value of layer size ratio is given for a network, how to maintain this layer size ratio? Current super-peer approaches lack an appropriate size ratio maintenance mechanism, and make the system's search performance far from being optimal. Second, what types of peers should be elected to super-layer? As far as we know, no efficient algorithm is given to ensure that the super-peers have large-capacity and long-lifetime. Since no global knowledge exist in distributed P2P systems, it is impossible to tell what values can be "high" or "long" enough, especially under highly dynamic environments. This problem is even more difficult to solve when the network needs to consider more metrics besides capacity and duration-time.

In this paper, we propose a dynamic layer management algorithm to solve these two problems, called DLM, which is a fully distributed dynamic layer management algorithm; it can adaptively elect peers and adjust them between super-layer and leaf-layer. DLM is completely distributed in the sense that each peer determines to be a super-peer or a leaf-peer independently without global knowledge. DLM could effectively help a super-peer P2P system maintain a given layer size ratio, and designate peers with relatively long lifetime and large capacities as super-peers, and the peers with short lifetime and low capacities as leaf-peers under highly dynamic network situations.

The rest of this paper is organized as follows. In Section 2, we discuss the related work. In Section 3, we describe the importance of layer size ratio in super-peer systems and network assumptions. We then present the design rationale of DLM in Section 4. In Section 5, we evaluate the effectiveness of DLM through simulations. In Section 6, we discuss the side effects of DLM, and we conclude this paper in Section 7.

2. Related Work

Applications based on Super-peer architectures are consuming a large portion of Internet traffic. For example, in June 2002, KaZaA consumed approximately 37% of all TCP traffic, which was more than twice the Web traffic on the University of Washington campus network [6]. Sandvine also estimates that 76% of P2P file sharing traffic is KaZaA/FastTrack traffic and only 8% is Gnutella traffic in the US [9]. Thus, super-peer architectures are attracting more and more attentions in P2P research communities.

To well understand the behavior of super-peer systems, authors in [17] examined the performance tradeoffs in super-peer systems by considering super-peer redundancy and topology variations. They also studied the potential drawbacks of super-peer networks and reliability issues. To make Gnutella network more scalable, Singla and Rohrs [2] describe how Ultrapeers work in an ideal network with a static topology and a handshaking mechanism based on Gnutella v0.6 protocol. Some requirements for super-peers were proposed, such as not fire walled, suitable operating system, sufficient bandwidth and sufficient uptime. Authors in [15] present some incentives to deploy super-peers and propose a topic-based search scheme to increase the effectiveness of super-peers.

However, because KaZaA, the most popular application based on super-peer architectures, is proprietary and uses encryption, little has been known about the protocols, architectures, and behaviors of KaZaA. With over millions of users at anytime, KaZaA

has neither been documented nor analyzed. Since super-peers and leaf-peers take different responsibilities, peers need to be assigned to appropriate layers through some well-designed layer management mechanism. But as far as we know, no such mechanism is proposed to date.

3. Impact of Layer Size Ratio

We first discuss the importance of maintaining an appropriate layer size ratio in a super-peer network.

In a super-peer network, the search tasks are mainly performed by super-peers, which actually form the “backbone” of the P2P network. Indeed, super-peer systems take advantage of peers’ heterogeneity by dividing peers into two layers- *super-layer* and *leaf-layer*, thereby scale better by reducing the number of query paths [17].

The peers in super-layer are called super-peers and are responsible for processing and relaying the queries from the leaf-peers and other super-peer neighbors. Each super-peer behaves like a proxy or agent of its leaf-peers, and keeps an index of its leaf-peers’ shared data. The peers in leaf-layer are called *leaf-peers*, and keep a small number of connections to super-peers for the purpose of reliability.

In super-peer systems, both super-peers and leaf-peers can submit queries, but only super-peers relay queries and query responses. A super-peer may forward an incoming query to its neighboring super-peers. When receiving a query, a super-peer first checks if the queried data is stored in local or in its leaf-peers (by checking the index of its leaf-peers’ objects). If some results are found in a peer, it will send a *QueryHit* message back to the query source along the inverse query path.

Compared with pure P2P systems, super-peer systems have higher search efficiency because instead of all the peers, only super-peers are involved in search processes. Intuitively, an appropriate layer size ratio, i.e. the ratio of the number of leaf-peers to the number of super-peers, is of great importance in the sense that, on one extreme, if most of the peers are in the super-layer, the system will be more like a pure P2P system since too many peers take part in searching; on the other extreme, if too few super-peers are available, the system is more like a centralized P2P system. A major drawback of a centralized P2P system is single point of failure.

Some layer management mechanisms use pre-configured values as the thresholds to select super-peers. For example, the Ultra-peer Proposal in Gnutella 0.6 [1] recommends at least 15KB/s downstream and 10KB/s upstream bandwidth. We argue that this approach cannot maintain an appropriate size ratio. Let us look at the example shown in Figure 1.

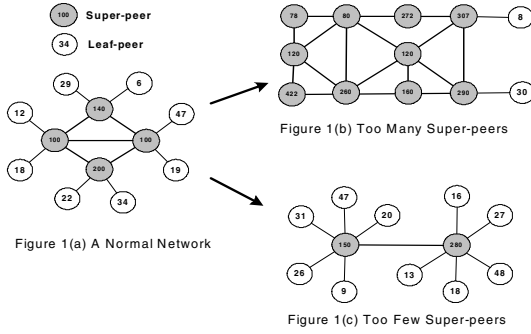


Figure 1 Inappropriate Layer Size Ratio

Suppose that a system sets the bandwidth threshold to 50KB/s. Peers with bandwidths larger than 50KB/s can be selected as super-peers. A network with good size ratio is shown in Figure 1(a), where the number inside a peer denotes its bandwidth value. However, after a certain time period, if most new joining peers have high bandwidths, the system will soon have too many super-peers, as shown in Figure 1(b). Or if most joining peers are weak ones with low bandwidths, the system will be more like a centralized P2P as shown in Figure 1(c).

Consider a P2P network with n participating peers, in which n_s peers are super-peers and n_l peers are leaf-peers. Assume each leaf-peer connects to m super-peers and each super-peer connects to k_s other super-peers and k_l leaf-peers, on the average. We use η to denote the layer size ratio of a super-peer network, and it is given by:

$$\eta = \frac{n_l}{n_s}.$$

Consider all the connections between super-layer and leaf-layer. The out-degree from super-layer to leaf-layer is $n_s k_l$, and the out-degree from leaf-layer to super-layer is $n_l m$. Since these two out-degrees are equal, so we have,

$$k_l = \frac{n_l m}{n_s} = m \eta. \quad (\text{Equation a})$$

Also, since $n_s + n_l = n$, we have

$$n_s = \frac{n}{1 + \eta}. \quad (\text{Equation b})$$

Therefore, in a network with n , n_s , n_l , m defined above, on the average each super-peer connects to $k_l = m \eta$ leaf-peers, and $n_s = \frac{n}{1 + \eta}$.

The size ratio of these two layers depends on the specific systems and applications, since each specific P2P system has its own targets and functions. Due to page limit, we leave the discussion of the appropriate size ratio to our other papers. In this paper, we just assume that the value of η is given by the protocol, and every participating peer of the network knows this value.

We also assume that new peers randomly select active peers as neighbors based on the bootstrapping and joining mechanisms currently used, although this randomness does not necessarily lead to totally random topology [11]. Some studies also show that keeping high-capacity peers with high degree can make the system more scalable [18], but it also imports unfairness among peers, and thereby hinders the peers accurately report the capacities, as well. In this paper, we assume that all the peers in the same layer prefer to be treated equally.

4. Design of DLM

We propose a Dynamic layer management algorithm (DLM) that intends to deal with the two problems described in Section 1. These problems are not trivial to deal with, since no peer knows the global knowledge of the network. In other words, DLM should achieve two goals: (1) maintains the size ratio of super layer to leaf layer; (2) keeps the peers with larger lifetimes and capacities as super-peers, and keeps the peers with shorter lifetimes and capacities as leaf-peers.

Unfortunately, these two goals are not always compatible. For example, at some time, the system may have a highly skewed layer size ratio, and more super-peers are heavily needed, but all the joining peers have low bandwidths. In this case, DLM needs to consider the two goals simultaneously, and adaptively promote some leaf-peers, though not "powerful" enough, to super-layer. Moreover, to perform well under both stable and highly changeable network situations, DLM needs to dynamically adjust the promotion and demotion policies according to the changing environments.

Ideally, the super-peers should be more powerful and with longer lifetime. To measure the eligibility of a peer, we define two metrics, *capacity* and *age*.

Definition 1: We define *capacity* as the ability of a peer to process and relay queries and query responses.

We use $capacity(d)$ to refer to the capacity value of a peer d . The value of capacity can be computed as,

$$capacity(d) = \sum_{i=1}^r w_i * v_i(d),$$

where r is the number of metrics affecting the peer's capacity, $v_i(d)$ is the value of i_{th} metric, and $w_i(d)$ is the weight of the corresponding metric. Examples of the metrics affecting the capacity are bandwidth, CPU powerfulness and storage space. We assume that a peer's capacity value does not change throughout its session and can be known when it connects the network. For simplicity, we omit the computation details of a peer's capacity, and just use the bandwidth of a peer as its

capacity, which does not affect the presentation of DLM algorithm.

Definition 2: We define *age* as the length of time up to now since a peer joins the network up to present.

The lifetime of a peer is the time the peer participates in the P2P network. It is the gap from the time the peer joins the network to the time it leaves the network. The *age* is less or equal to the lifetime of a peer by the definition. DLM will automatically promote the leaf peers with larger ages to super layer, and demote the super-peers with smaller ages to leaf layer. Although the *capacity* value can be set at the time when the peer joins the system, there are no means to know the lifetime of a peer, since a peer may leave the network at any time. One practical way to infer the lifetime is by monitoring its *age*: the longer the peer lives, more likely the peer will live in the future. So we use the *age* of a peer in DLM to denote its lifetime. We use *age(d)* to denote peer *d*'s *age* value.

In DLM, each peer first collects its neighboring peers' information, which includes the *capacities*, *ages*, and the number of neighboring leaf-peers (for a super-peer neighbor). After processing the collected information, the peer will determine its own status. We describe DLM in four phases.

- **Phase 1: Information Collection**

Peers first exchange information with its neighbors using messages. We design two pairs of messages, which are employed by a pair of leaf-peer and super-peer. The formats of these two pairs of messages are shown in Table 1.

Table 1 Message Formats

| Messages | Value | |
|--------------------|----------|------|
| Neigh_num_request | Null | |
| Neigh_num_response | l_{mn} | |
| Value_request | Null | Null |
| Value_response | capacity | age |

The first pair of the messages includes *neighbor_num_request* and *neighbor_num_response* Messages; *neighbor_num_request* is sent from a leaf-peer *d* to a super-peer *s* to request the leaf neighbor number of *s*, and we use $l_{mn}(s)$ to refer to this number. Message *neighbor_num_response* is the response from the super-peer. The second pair of the messages is *value_request* and *value_response*, which are sent between a super-peer and a leaf peer to query and respond the latter peer's *capacity* and *age*. Note that these two pairs of messages can also be piggybacked in other messages in some P2P protocols.

One issue deserves some words here is how often that the peers exchange this information. Obviously, higher

frequency means higher accuracy, while more traffic overhead as well. In this design we employ an event-driven policy, in which information exchange is invoked whenever a peer finds that a new connection is created. In simulation, we have also evaluated other policies, such as time interval based policy where peers exchange information periodically. Our results show that event-driven performs the best in the sense that it incurred smaller overhead when having the same performance. Also, the amount of this overhead is trivial compared with other traffic costs in P2P systems, which we will discuss in Section 6.

- **Phase 2: Maintaining Appropriate Layer-Size-Ratio**

One of the goals of DLM is to maintain appropriate layer size ratio. To achieve this goal, DLM first needs to estimate the extent of appropriateness of current layer size ratio.

The basic idea of DLM is based on three observations. First, current layer size ratio can be easily calculated if the global knowledge is known, but in reality no peer knows this global knowledge. Second, it is trivial for a peer to collect its neighbors' information. Third, due to the randomness of the neighbor selection mechanism in super-peer systems, to some extent, the current numbers of leaf neighbors of super-peers can reflect the current layer size ratio. That is, if the super-layer size is too small, the average number of leaf neighbors of a super-peer will be larger than k_l , the "optimal" leaf neighbor number; otherwise it will be less than k_l . Since each peer knows the optimal value of η , then the value of k_l can be computed using Equation a. Therefore, l_{mn} , the leaf neighbor number of some super-peer, can be used as a flag of the current layer size ratio when compared with the value of k_l .

To illustrate the comparison method, we now define the *related set* of a peer.

Definition 3: we use G to represent the *related set* of a peer in the other layer. For a super-peer s , we define $G(s)$ as the set of its current neighboring leaf-peers. While for a leaf-peer l , we define $G(l)$ as the set of super-peers that it has connected within a period of time T_l .

The definitions of G on super-peers and leaf-peers are different as a super-peer normally keeps connections to many leaf-peers, while a leaf-peer normally only connects to a few super-peers. To more accurately estimate the network condition, we hope that the size of G is large enough, so that we consider all the super-peers that a leaf-peer has contacted in a recent period of time. In simulation, G contains all the super-peers that a leaf-peer has connected since it joins the network.

For a super-peer s , since it connects to some leaf-peers, it can directly use $l_{nm}(s)$. While for a leaf-peer l , it uses the average l_{nm} value of the super-peers in $G(l)$. We let μ to denote the extent of inappropriateness of current layer size ratio compared to the optimal layer size ratio η , and it is computed as,

$$\mu = \log(l_{nm}/k_l).$$

We can see that a positive value of μ , where l_{nm} is larger than k_l , means that there are too few super-peers in the system, since super-peers have more leaf-peer neighbors than normal. While a negative value of μ , where l_{nm} is smaller than k_l , means that there are too many super-peers. Furthermore, a larger absolute value of μ means a larger degree of inappropriateness. Thereby the value of μ reflects the system requirement, and it is used to adjust some other parameters, which determine the possibility of a peer's being promoted or demoted.

• **Phase 3: Scaled Comparisons of Capacity and Age**

DLM automatically promotes the *leaf-peers* with large capacities and longer lifetimes to super-layer, and demotes the super-peers with small capacities and shorter lifetimes to leaf-layer. The decision of promotion or demotion is based on the comparison results with other peers.

One straightforward way of comparison is directly comparing the metric values of a peer with other peers. However, comparing in such a direct way may fail to maintain the layer size ratio successfully. Consider a scenario in which the system needs more super-peers, but the leaf-peers all have larger metric values than the current super-peers. The results of simple comparisons would forbid any leaf-peer to be promoted; thereby the system cannot adjust the layer size ratio at all.

DLM improves the direct comparison by using "scaled-comparison". Since the capacity and age metrics are disjoint, that is, a peer with high-capacity not necessarily has larger age and vice versa, we analyze these two metrics individually. In scaled-comparison, DLM introduces two scale parameters, X_{capa} and X_{age} , corresponding to the two metrics respectively. These two scale parameters are adjusted dynamically by DLM based on the value of μ to reflect the system requirements.

For each peer that runs DLM, it uses two counting variables, Y_{capa} and Y_{age} , corresponding to Capacity and Age metrics, respectively. A peer sets the value of these two counting variables by comparing its metric values with the peers in its related set, G . For a peer d , the pseudo codes of the scaled-comparison are listed below.

for all peer d_j in $G(d)$
 if (capacity(d_j) * X_{capa} > capacity(d))
 $Y_{capa} += 1 / (\text{size of } G(d));$

if (age(d_j) * X_{age} > age(d))
 $Y_{age} += 1 / (\text{size of } G(d));$

We can see that Y_{capa} and Y_{age} store the fractions of peers that have "larger" metric values than those of d in $G(d)$. Y_{capa} reflects the relative capacity value of a peer compared to the peers in the other layer; while Y_{age} reflects the relative *age* value of one peer compared to the peers in the other layer. These two variables will be used to determine the eligibility of the peer's promotion or demotion.

The values of X_{capa} and X_{age} are adjusted according to the value of μ . For a super-peer, if it finds that the system needs more super-peers, it will decrease the possibility of its demotion by decreasing the two scale parameters. Otherwise, it will increase the possibility of its demotion by increasing the scale parameters. While for a leaf-peer, if it finds that more super-peers are needed, it will decrease the scale parameters in hoping to increase the promotion possibility; otherwise it will increase the scale parameters to decrease the promotion possibility.

• **Phase 4: Promotion or Demotion**

For a leaf-peer l , if Y_{capa} and Y_{age} are small enough, it means that many super-peers in $G(l)$ have metric values "smaller" than it. Thus, l may assume that it has comparatively large metric values and may decide to be promoted to super-layer. While for a super-peer s , if Y_{capa} and Y_{age} are large enough, it means that many leaf-peers in $G(l)$ have metric values "larger" than it. Thus, s may assume that it has comparatively small metric values and may decide to be demoted to leaf-layer.

We use two threshold variables, Z_{capa} and Z_{age} , in the determination. For a leaf-peer l , if Y_{capa} and Y_{age} are smaller than Z_{capa} and Z_{age} , respectively, it will be promoted to be a super-peer. In promotion, the leaf-peer keeps its current connections to other super-peers. The scenarios before and after peer l 's promotion are illustrated in Figure 2(a) and 2(b).

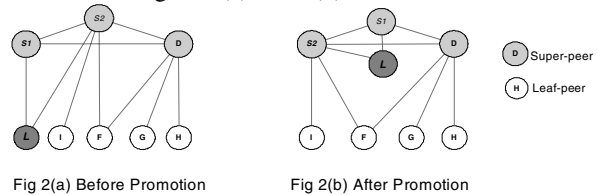


Figure 2 Promotion of a Leaf Peer

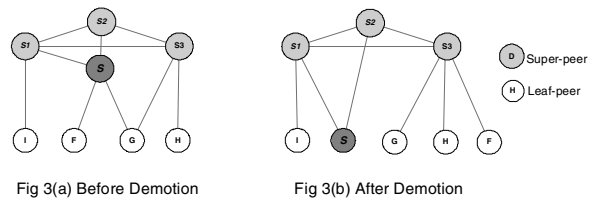


Figure 3 Demotion of a Super-peer

5. Performance Evaluation

For a super-peer s , if Z_{capa} and Y_{age} are larger than Z_{capa} and Z_{age} , respectively, it will be demoted to be a leaf-peer. In demotion, the super-peer only keeps m of its current connections to other super-peers and drops the connections to leaf-peers. The scenarios before and after peer s 's demotion are illustrated in Figure 3(a) and 3(b).

The values of threshold variables, Z_{capa} and Z_{age} , are also adjusted according to the value of μ . When more super-peers are needed, super-peers will increase the values of the threshold variables to reduce the demotion tendencies, and leaf-peers will reduce the values of the threshold variables to increase the promotion tendencies. For the cases that there are too many super-peers, inverse measures will be taken accordingly.

Essentially, to achieve the two goals we set in the beginning of Section 4, DLM needs to consider two factors: the layer-size-ratio factor (μ), and the capacity-age factor. The first factor reflects the requirement of keeping the optimal layer size ratio, and the second factor reflects the requirement of keeping large-capacity and large-age peers in the super-layer. Since the two scale parameters, X_{capa} and X_{age} , are adjusted based on the values of the first factor, DLM can achieve the first goal: maintaining the optimal layer size ratio. Similarly, by using scaled comparisons method, DLM achieves the second goal: keeping large-capacity and large-age peers on the super-layer.

We use simulation to evaluate DLM and compare it to preconfigured algorithms. To collect real data, we implement two Gnutella clients based on the publicly available Mutella [8]. One client participates into the network as an ultra-peer, and the other acts as a leaf-peer. Using these two clients we collect some first-hand data, such as the lifetimes of ultra-peers and leaf-peers, the number of open connections and average query frequencies of peer. The collected data are consistent with the data presented in previous studies [6, 12, 13]. We configure our simulation environments based on these

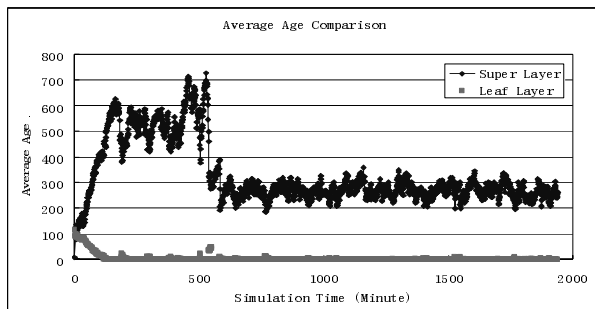


Figure 4 Average Age

observations. The values of the parameters are shown in Table 2.

Table 2. Simulation Parameters

| Parameter | Value | Description |
|-----------|-------|------------------------------------|
| n | 50000 | # of peers in the network |
| n_l | 48780 | # of preferred leaf-peers |
| n_s | 1220 | # of preferred super-peers |
| η | 40.0 | Layer size ratio |
| m | 2 | Super-peer neigh. # of a leaf-peer |
| k_l | 80 | Aver. leaf neigh. # of super-peers |
| k_s | 3 | Aver super neigh. # of super-peers |

We evaluate DLM under various simulation environments. We first simulate a *stable* network by assigning new peers with capacity and lifetime values based on previous studies [12]. We then simulate a *dynamic* network with dynamic means of the distributions. For these two cases, the network sizes are not changed. We also evaluate DLM on networks with different sizes. The results are consistent, so we only present the results in dynamic environments, in Figures 4, 5, and 6.

In a *stable* network, the simulation starts “cold”, i.e. without any peer. The size of the network increases with new peers joining until reaches the designated size. Then with time going, whenever a peer dies, a new peer is created and joins the network, thereby the network size does not change. The new peer is always assigned to leaf layer first, and DLM will promote eligible leaf peers to super layer. The lifetime and capacity values of a new peer are generated based on the distributions we observed and shown in previous studies.

We simulate the *dynamic* networks by varying the means of the capacity and age distributions for new joining peers. Starting from the 300th time unit, the lifetime values of new joining peers are generated using the method as before, but with half mean values. Thus from that time, the new peers have smaller lifetime values. Figure 4 plots the average ages of each layer. As expected, the age of super-layer is much larger than that of leaf-layer, regardless the changing environments.

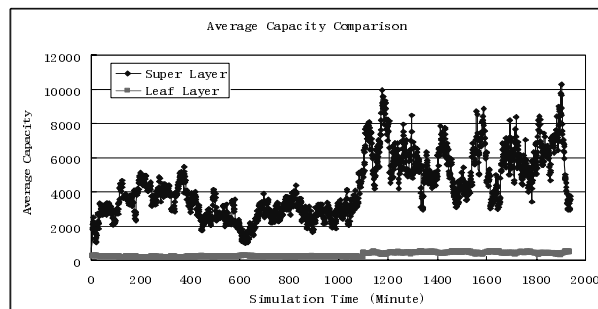


Figure 5 Average Capacity

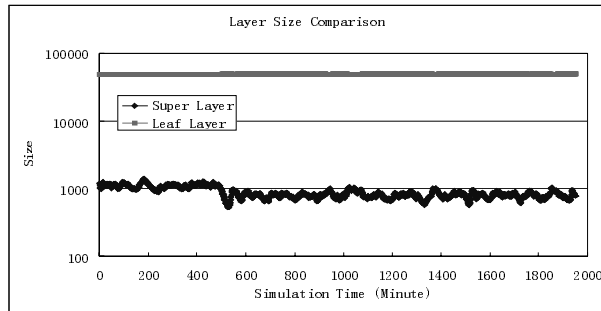


Figure 6 Layer Sizes

Starting from the 1,000th time unit, the capacities of new joining peers are generated with doubled mean values; thereby the new peers are more powerful than previous peers. The results in Figure 5 indicate that DLM adaptively promotes the peers with large-capacities to super-layers and the average capacity value of super-layer is always larger than that of leaf-layer.

The layer sizes of the network are shown in Figure 6, we can see that an almost constant ratio is maintained throughout the simulation process, even the network environment is changing. Note that the Y-axis of Figure 6 is logarithmic.

We also compare DLM with pre-configured algorithms under dynamic network situations where the new peers' mean capacity values are periodically changed. We compare the sizes and ages of two layers in DLM and a preconfigured algorithm.

The results are shown in Figures 7 and 8. We can see that the DLM maintains the layer size ratio very well, while in the preconfigured algorithm, the layer size ratio changes periodically. For the average ages of super-layer and leaf-layer, in DLM, they are sharply divided and the average age of super-layer is much larger than that of the preconfigured algorithm, as shown in Figure 8. The comparisons of capacity values show similar results.

6. Discussion on Side Effects of DLM

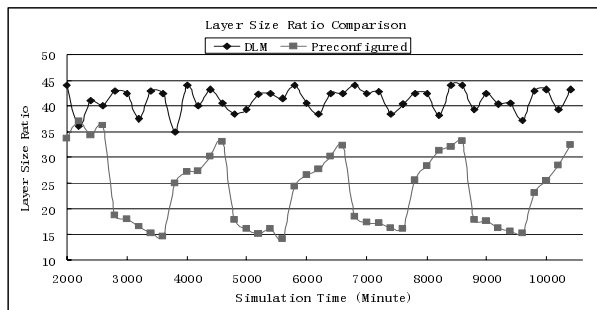


Figure 7 Layer Size Ratios on Same Success Rate

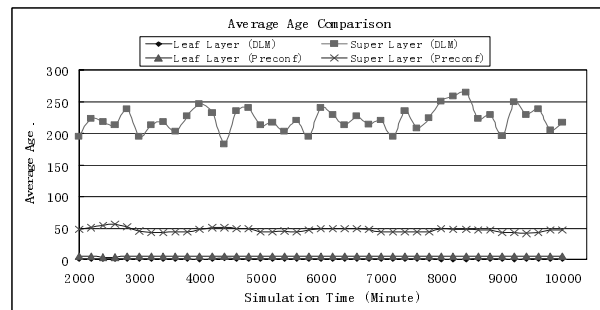


Figure 8 Average Age Comparisons

Introducing DLM to super-peer systems does incur some traffic overhead, such as the traffic overhead for information exchanging among neighbors, and the peer adjustment overhead. In this section we discuss these issues in details.

- **Overhead for Information Exchanging**

To implement DLM, we propose to add two pairs of messages described in Section 4 to the existing super-peer P2P protocol. We argue that the additional overhead on delivering these two pairs of messages is negligible compared to the search traffic costs in a P2P system for two reasons. First, these messages are only transferred between directly connected neighbors, so they can have very simple formats and only need few bytes. As a result, these two pairs of messages are quite light-weighted. Second, these messages are only sent when new connections are created. Moreover, these two pairs of messages may be piggybacked in other messages available, thus reducing the traffic overhead even more. Our collected data and studies in [12] found that each connection can keep active for at least several minutes on the average. Therefore, the frequency of DLM message transferring is quite low compared with that of other messages.

- **Peer Adjustment Overhead**

When a super-peer is demoted to be a leaf-peer, it needs to cut the connections to the leaf-peers and some super-peers because a leaf-peer only keeps a small number (m) of links to super-peers. The leaf-peers disconnected by the demoted super-peer need to connect to another super-peer instead. We call this kind of connection overhead as *Peer Adjustment Overhead (PAO)*. Note that the promotion process does not cause PAO because no peers are disconnected during the process. We analyze this overhead and find that the peer adjustment overhead is quite small. The results are shown in Table 3.

Table 3 Peer Adjustment Overhead Analysis

| Network size | # of new leaf-peers | # of demoted super-peers | # of disconnected leaf-peers | PAO/NLCO (%) |
|--------------|---------------------|--------------------------|------------------------------|--------------|
| 5,000 | 21.19 | 0.55 | 44.56 | 104.65% |
| 20,000 | 84.76 | 1.19 | 95.26 | 56.19% |
| 80,000 | 339.21 | 2.06 | 157.73 | 23.25% |

We count the number of new peers, the number of super-peers demoted as leaf-peers, and the number of disconnected leaf-peers caused by the demotions per unit time. The disconnected leaf-peers need to connect other super-peers and incur the PAO. This process behaves like a new joining leaf-peer making connections to super-peers. The difference is that each disconnected leaf-peer only needs to create one new connection to another super-peer, while each new joining leaf-peer needs to create m new connections to super-peers. Thus, the PAO for a disconnected leaf peer is only $1/m$ of the connection overhead for a new joining peer. We call the connection overhead caused by new leaf-peers as *New Leaf-initiated Connection Overhead (NLCO)*. We also calculate the ratio of PAO and NLCO in Table 3.

In Table 2, we can see that as the network size increases, the ratio of PAO to NLCO decreases. The reason is that as the network size increases, the number of leaf-peers each super-peer connects to is more close to k_i due to the randomness of connections between peers. Therefore, the probability of misjudgments is also decreased. It is safe to expect that in the real-world P2P networks with millions of peers, the ratio of PAO to NLCO is even smaller.

7. Conclusion

In this paper, we propose a dynamic layer management algorithm, DLM, which can adaptively elect peers and adjust them between super-layer and leaf-layer. Our simulation results show that DLM can maintain a given size ratio of super-layer to leaf-layer. It also designates peers with long lifetime and large capacities as super-peers, and the peers with short lifetime and low capacity as leaf-peers under highly dynamic network situations. With wide deployment of DLM, the quality of a super-peer system can be significantly improved.

References

[1] The Gnutella protocol specification 0.6. 2002.
 [2] Anurag Singla, C.R., Ultrapeers: Another Step Towards Gnutella Scalability, in *Working draft, available from the Gnutella Developers Forum at http://groups.yahoo.com/group/the_gdf/*.

[3] Bustamante, F.E. and Y. Qiao. Friendships that last: Peer lifespan and its role in P2P protocols. In *Proceeding of International Workshop on Web Content Caching and Distribution*. 2003. NY, USA.
 [4] Ganesan, P., K. Gummadi, and H. Garcia-Molina. Canon in G Major: Designing DHTs with Hierarchical Structure. In *Proceeding of the 24th International Conference on Distributed Computing Systems (ICDCS 2004)*. 2004. Tokyo, Japan.
 [5] Gnutella, <http://gnutella.wego.com/>.
 [6] Gummadi, K.P., et al. Measurement, Modeling, and Analysis of a Peer-to-Peer File-Sharing Workload. In *Proceeding of the 19th ACM Symposium on Operating Systems Principles (SOSP)*. October 2003. Bolton Landing, NY.
 [7] Liu, Y., et al. A Distributed Approach to Solving Overlay Mismatching Problem. In *Proceeding of IEEE ICDCS 2004*. 2004. Tokyo, Japan.
 [8] Mutella, <http://mutella.sourceforge.net/>. 2003.
 [9] P2P, R.c.o., <http://www.sandvine.com>.
 [10] Peter Backx, T.W., Bart Dhoedt, Piet Demeester. A comparison of peer-to-peer architectures. In *Proceeding of Eurescom Summit*. 2002. Heidelberg, Germany.
 [11] Ripeanu, M., A. Iamnitchi, and I. Foster, Mapping the Gnutella Network. *IEEE Internet Computing*, 2002.
 [12] Saroiu, S., P. Gummadi, and S. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proceeding of Multimedia Computing and Networking (MMCN)*. 2002.
 [13] Sen, S. and J. Wang. Analyzing peer-to-peer traffic across large networks. In *Proceeding of ACM SIGCOMM Internet Measurement Workshop*. 2002.
 [14] Shen, H., C.-Z. Xu, and G. Chen. Cycloid: A Constant-Degree and Lookup-Efficient P2P Overlay Network. In *Proceeding of the International Parallel and Distributed Processing Symposium (IPDPS)*. 2004. Santa Fe, New Mexico.
 [15] Singh, S., et al. The Case for Service Provider Deployment of Super-Peers in Peer-to-Peer Networks. In *Proceeding of the Workshop on Economics of Peer-to-Peer Systems*. 2003. Berkeley.
 [16] Stoica, I., et al. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proceeding of ACM SIGCOMM*. 2001.
 [17] Yang, B. and H. Garcia-Molina. Designing a super-peer network. In *Proceeding of the 19th International Conference on Data Engineering (ICDE)*. March 2003. Bangalore, India.
 [18] Yatin Chawathe, et al. Making Gnutella-like P2P Systems Scalable. In *Proceeding of ACM SIGCOMM*. 2003.
 [19] Zhuang, Z., et al. Hybrid Periodical Flooding in Unstructured Peer-to-Peer Networks. In *Proceeding of Proceedings of International Conference on Parallel Processing ICPP'03*. 2003.