Dynamic Layer Management in Superpeer Architectures

Li Xiao, *Member*, *IEEE*, Zhenyun Zhuang, and Yunhao Liu, *Member*, *IEEE*

Abstract—Superpeer unstructured P2P systems have been found to be very effective by dividing the peers into two layers, superlayer and leaf-layer, in which message flooding is only conducted among superlayer and all leaf-peers are represented by corresponding superpeers. However, current superpeer systems do not employ any effective layer management schemes, so the transient and low-capacity peers are allowed to act as superpeers. Moreover, the lack of an appropriate size ratio maintenance mechanism on superlayer to leaf-layer makes the system's search performance far from being optimal. We present one workload model aimed at reducing the weighted overhead of a network. Using our proposed workload model, a network can determine an optimal layer size ratio between leaf-layer and superlayer. We then propose a Dynamic Layer Management algorithm, DLM, which can maintain an optimal layer size ratio and adaptively elect and adjust peers between superlayer and leaf-layer. DLM is completely distributed in the sense that each peer decides to be a superpeer or a leaf-peer independently without global knowledge. DLM could effectively help a superpeer P2P system maintain the optimal layer size ratio and designate peers with relatively long lifetime and large capacities as superpeers, and the peers with short lifetime and low capacities as leaf-peers under highly dynamic network situations. We demonstrate that the quality of a superpeer system is significantly improved under the DLM scheme by comprehensive simulations.

Index Terms—Unstructured peer-to-peer, superpeer architecture, layer management, workload analysis, adaptive algorithms.

1 Introduction

An early generation of unstructured P2P systems is *purely* unstructured, such as Gnutella [8], where all the peers are involved in the query flooding process without exception. However, peers can be very different from each other in bandwidth, CPU power, duration times, shared files, and interests [6], [9]. In pure P2P systems, all peers, regardless of their capacities, act equal roles and take the same responsibilities for all the operations. As the network size increases, the weak peers will seriously limit the scalability of P2P systems. To address this problem, superpeer architectures were proposed, which have been attracting more and more users in unstructured P2P communities. For example, KaZaA and early Morpheus [15], based on FastTrack structure, have dominated the top downloads lists for most of 2001 and are continuing to increase in popularity in 2002 [9], [3]. Schemes are also proposed to introduce Ultra-peers into Gnutella protocol [22].

The advantages of superpeer systems are well discussed in [26]. However, current superpeer systems do not exploit any effective layer management schemes; thereby, they leave the following three problems unsolved: First, how many superpeers are preferred to exist in the network? In other words, what is the optimal size ratio of leaf-layer to superlayer? Second, assuming an optimal value of layer size ratio exists for a network, how can this layer size ratio be maintained? Current superpeer approaches lack an appropriate size ratio maintenance mechanism and make the system's search performance far from optimal. Third, what types of peers should be elected to superlayer? This problem has not been effectively solved yet. As far as we know, no efficient mechanism is given to ensure that the superpeers have large-capacity and long-lifetime. Since no global knowledge is available 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 lifetime.

In this paper, we present one workload model to solve the first question mentioned above and present one dynamic layer management algorithm (DLM) to solve the other two questions. The main contributions of this paper are as follows:

- First, we discuss the importance for a superpeer P2P system to have an appropriate size ratio on superlayer to leaf-layer and obtain the optimal size ratio by modeling superpeer systems based on existing studies and our observations.
- Second, we propose a fully distributed dynamic layer management algorithm, DLM, which can adaptively elect peers and adjust them between superlayer and leaf-layer. DLM is completely distributed in the sense that each peer is determined to be a superpeer or a leaf-peer independently without global knowledge. DLM could effectively help a superpeer-P2P system maintain an appropriate layer size ratio and designate peers with relatively long lifetime and large capacities as superpeers and the

- *L. Xiao and Z. Zhuang are with the Department of Computer Science and Engineering, 3115 Engineering Building, Michigan State University, East Lansing, MI 48824. E-mail: {lxiao, zhuangz1}@cse.msu.edu.*
- *Y. Liu is with the Department of Computer Science, Hong Kong University of Science and Technology, Kowloon, Hong Kong. E-mail: liu@cs.ust.hk.*

Manuscript received 31 July 2004; revised 17 Dec. 2004; accepted 21 Dec. 2004; published online 21 Sept. 2005.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-0190-0704.

peers with short lifetime and low capacities as leaf-peers under highly dynamic network situations.

- Third, we demonstrate that the quality of a superpeer system is significantly improved under the DLM scheme by comprehensive simulations.

The rest of this paper is organized as follows: In Section 2, we discuss the related work. We present our workload model to illustrate the trade-off between two workloads as a function of layer size ratio in Section 3. Using the workload model, we obtain an optimal layer size ratio value that can minimize the weighted workload. In Section 4, we describe the design rationale of DLM, along with different variations of it. We then evaluate the effectiveness of the workload model and DLM through simulations in Section 5 and discuss the side effects of DLM in Section 6. We conclude this paper in Section 7.

2 Related Work

To accurately understand P2P systems, many researchers have tried to model the behaviors of the peers and the network. Ge et al. [7] developed a simple mathematical model to illustrate the performance issues of P2P systems and apply the model to three different peer-peer architectures: centralized indexing, distributed indexing with flooded queries, and distributed indexing with hash-directed queries.

For unstructured P2P systems, Menasce and Kanchanapalli [14] proposed a P2P protocol and developed an analytical model to analyze the performance of the presented protocol. The studied metrics include the success rates, the fraction of peers involved in a search, and the average number of hops required to find a directory entry. Cooper and Garcia-Molina [5] proposed a search/index links (SIL) model for P2P search networks and their simulation results suggest that the parallel search clusters are superior to existing superpeer networks under some metrics.

Numerous structured P2P protocols have also been presented, along with mathematical analysis [18], [23], [27], [17]. To capture the common design techniques of the various structured P2P mechanisms, a tree-based model was proposed in [10] and the routing adaptively feature was analyzed. Xu et al. [24] studied the fundamental trade-offs between the routing table size and the network diameter by modeling the P2P systems.

Applications based on superpeer architectures are consuming a large portion of Internet traffic. In June 2002, for example, KaZaA consumed approximately 37 percent of all TCP traffic, which was more than twice the Web traffic on the University of Washington campus network [9]. Sandvine also estimates that 76 percent of P2P file sharing traffic belongs to KaZaA/FastTrack traffic and only 8 percent comes from Gnutella in the US [2]. Thus, superpeer architectures are attracting more and more attention in P2P research communities.

To well understand the behavior of superpeer systems, Yang and Garcia-Molina [26] examined the performance trade-offs in superpeer systems by considering superpeer redundancy and topology variations. They also studied the potential drawbacks of superpeer networks and reliability issues. To make the Gnutella network more scalable, Singla and Rohrs [22] described how ultrapeers work in an ideal network with a static topology and a handshaking mechanism based on the Gnutella v0.6 protocol, in which some requirements for superpeers were proposed, such as not firewalled, suitable operating system, sufficient bandwidth, and sufficient uptime. Singh et al. [21] presented some incentives to deploy superpeers and proposed a topic-based search scheme to increase the effectiveness of superpeers.

However, because KaZaA is proprietary and uses encryption, little has been known about the protocols, architectures, and behaviors of KaZaA, the most popular application based on superpeer architectures. With over three million users at any time, KaZaA has neither been documented nor analyzed. Since superpeers and leaf-peers take different responsibilities, peers need 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. In this paper, we propose a dynamic layer management algorithm and evaluate its effectiveness.

3 Workload Model

We first discuss the importance of maintaining an appropriate layer size ratio in a superpeer network. We then propose a workload model to analyze the performance of superpeer architectures in this section. Our model focuses on the amounts of workloads, both on superpeers and the overall network, and we obtain an optimal layer size ratio by minimizing the weighted workload.

3.1 Importance of Appropriate Layer Size Ratio

In a superpeer network, the search is mainly performed by superpeers, which actually form the "backbone" of the P2P network. Superpeer systems take advantage of peers' heterogeneity by dividing peers into two layers: *superlayer* and *leaf-layer*, thereby scaling better by reducing the number of query paths [26]. The peers in the superlayer are called *superpeers*, which are responsible for processing and relaying the queries that come from the leaf-peers and other superpeer neighbors. Each superpeer behaves like a proxy or an agent of its leaf-peers and keeps an index of its leaf-peers' shared data. The peers in the leaf-layer are called *leaf-peers*, which only keep a number of connections to superpeers for the purpose of reliability. In superpeer systems, both superpeers and leaf-peers can submit queries, but only superpeers can relay queries and responses. After receiving a query, a superpeer first checks to see if the queried data is stored locally or in its leaf-peers (by checking the index of its leaf-peers' objects). If some results are found in a peer, it sends a *QueryHit* message back to the query source along the inverse query path.

Compared with pure P2P systems, superpeer systems have higher search efficiency because, instead of all the peers, only superpeers 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 superpeers, is of great importance. One proposed layer management mechanism [25] uses preconfigured values as the thresholds to

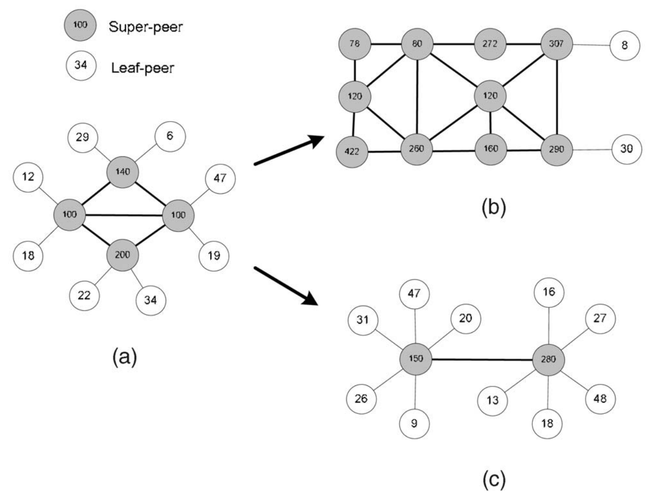


Fig. 1. Inappropriate layer size ratio.

select superpeers which cannot maintain an appropriate size ratio and are explained in Fig. 1.

Suppose that a system sets the bandwidth threshold to 50KB/s (the Ultra-peer Proposal in Gnutella 0.6 [1] recommends at least 15KB/s downstream and 10KB/s upstream bandwidth). Peers with bandwidths larger than 50KB/s can be selected as superpeers. A network with good size ratio is shown in Fig. 1a, where the number inside a peer denotes its bandwidth value. However, after a certain time period, if most joining peers have high bandwidths, the system will soon have too many superpeers, as shown in Fig. 1b. The system in this case will be more like a pure P2P system since most peers take part in the search process. If most joining peers are weak ones with low bandwidths, the system will behave like a centralized P2P system with too few superpeers, as shown in Fig. 1c. A major drawback of a centralized P2P system is the single point of failure.

3.2 Workload Model

To better analyze the impact of layer size ratio on the system performance, we model a P2P superpeer system as follows: Consider a P2P network with n participating peers, in which n_s peers are superpeers and n_l peers are leaf-peers. Assume that each leaf-peer connects to m superpeers and each superpeer connects to k_s other superpeers and k_l leaf-peers, on the average. We use η to denote the layer size ratio of a superpeer network, which is given by: $\eta = \frac{n_l}{n_s}$.

Theorem 1. *On the average, each superpeer connects to m leaf-peers and the number of superpeers is given by $n_s = \frac{n}{1+\eta}$.*

Proof. Consider all the connections between the superlayer and the leaf-layer. The out-degree from the superlayer to the leaf-layer is $n_s k_1$ and the out-degree from the leaf-layer to the superlayer is $n_l m$. Since these two out-degrees are equal, we have $k_l = \frac{n_l m}{n_s} = m\eta$. Also, since $n_s + n_l = n$ and $\eta = \frac{n_l}{n_s}$, we have $n_s = \frac{n}{1+\eta}$. □

We define two different types of workloads: the *Workload on the Overall Network* and the *Workload on a Superpeer*. The *Workload on the Overall Network*, denoted as W_{on}, is defined as the total traffic overhead required to perform a search task in a P2P network. The *Workload on a Superpeer*, denoted as W_{sp}, is defined as the traffic overhead required on a superpeer to perform a search task. The performance of a network depends on the workloads on each peer and the overall network. The network administrators (or ISPs) are concerned more about W_{on}, while the users are concerned more about W_{sp}. We use the number of messages needing to be processed to measure the amount of the workloads. Hence, each connection creation, query initiation, and query relay can be done using one message, respectively.

Now, we analyze W_{on} and W_{sp} as a function of η. A superpeer mainly performs the following tasks: 1) maintaining the connections to the neighboring peers, 2) processing the queries initiated from its leaf-peers and itself, and 3) relaying the queries come from its superpeer neighbors. Therefore, the workloads on a superpeer and the overall network can be divided into three parts: *Connection Workload*, *Query Workload*, and *Relay Workload*. We now define and analyze each of them.

3.2.1 Connection Workload

There are two kinds of connections in superpeer networks—the connections between superpeers and the connections between leaf-peers and superpeers. The *Connection Workload* of a peer is defined as the traffic overhead incurred to maintain the connections to the neighboring peers. The amount of connection workload on any peer is directly related to the size and stability of the neighboring peer set and is proportional to the number of neighbors and inversely proportional to the average up time of them. We use W_{sp_cw} and W_{on_cw} to denote the portions of connection workload in W_{sp} and W_{on}, respectively. Thus,

$$W_{sp\_cw} = \frac{k_l}{t_l} + \frac{k_s}{t_s} = \frac{m\eta}{t_l} + \frac{k_s}{t_s}$$

and

$$W_{on\_cw} = \frac{n_l m}{t_l} + \frac{n_s}{t_s}(k_l + k_s) = \frac{mn\eta}{(1+\eta)t_l} + \frac{n(m\eta + k_s)}{(1+\eta)t_s},$$

where k_l is the average number of neighboring leaf-peers of a superpeer, k_s is the average number of neighboring superpeers of a superpeer, and t_l and t_s are the average lifetimes of neighboring leaf-peers and superpeers, respectively.

3.2.2 Query Workload

In a typical query process, when a query is issued by a leaf-peer, the leaf-peer sends the query to its neighboring superpeers. (In this paper, we assume that a leaf-peer sends a query to all of its super neighbors.) This process contributes to W_{on} and W_{sp} by the communication overhead between leaf-peers and superpeers.

Query Workload is defined as the traffic overhead incurred for a peer to process the queries generated by its leaf neighbors and itself. The amount of the query workload is proportional to the number of leaf neighbors and the query frequency of each peer. We use W_{sp_qw} and W_{on_qw} to denote the portions of query workload in W_{sp} and W_{on}, respectively. Thus, we have,

$$W_{sp\_qw} = k_l f = m\eta f \text{ and } W_{on\_qw} = \frac{nm\eta f}{1+\eta},$$

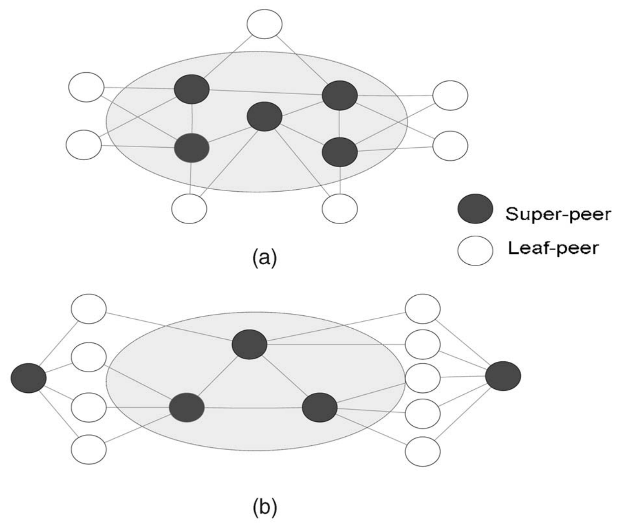


Fig. 2. Number of covered superpeers.

where f is the query frequency of a peer. Since, in reality, the query frequency of a peer does not have a correlation with its identity, we assume that all the peers have the same f values regardless of superpeers or leaf-peers.

3.2.3 Relay Workload

After receiving a query, superpeers relay it to other superpeers. In this process, the relaying communications are performed among superpeers only. *Relay Workload* is defined as the traffic overhead incurred to process queries relayed from the superpeer neighbors. Since, in unstructured P2P systems, each peer independently decides what data to share, the search success rate heavily depends on how many peers are searched. To quantify the relay workload, we assume that each query should reach at least a certain number of peers to ensure a given query search scope. We use p to denote the number of peers that are queried (covered) in a search task. The following theorem specifies two bounds of the number of covered superpeers as a function of p:

Theorem 2. *In a superpeer network with each leaf-peer connecting m superpeers and each superpeer connecting k_1 leaf-peers, to cover p peers, the number of superpeers that should be queried has a lower bound of $\frac{p}{(1+k_l)}$ and an upper bound of $\frac{mp}{(m+k_l)}$ on average.*

Proof. Since the queries are relayed and processed among superpeers and each superpeer stores the data indices of k_l leaf-peers, a superpeer can be viewed to represent k_l+1 peers (k_l leaf-peers and itself). In Fig. 2, suppose p_s superpeers are queried and they are all located in the gray ellipse zone, then the number of their connected leaf-peers is $p_l = p - p_s$.

One extreme case is that every leaf-peer, which is represented by the p_s superpeers, only connects to these p_s superpeers, as shown in Fig. 2a. Since the total number of outgoing links connecting the leaf-layer is $p_s k_1$, which is equal to the number of incoming links $p_l m$ from leaf-peers, we have $p_s k_l = p_l m$. This is the upper bound case.

The other extreme case is that each leaf-peer, which is represented by the p_s superpeers, connects to only one of these p_s superpeers, as shown in Fig. 2b. So, we have $p_s k_l = p_l$. This is the lower bound case.

Thus, in general, we have $p - p_s \le p_s k_l \le (p-p_s)m$, which is $\frac{p}{(1+k_l)} \le p_s \le \frac{mp}{(m+k_l)}$. □

Theorem 3. *When $p_s << n_s$, the probability of a leaf-peer connecting to more than one covered superpeer is very close to zero.*

Proof. Consider a leaf-peer that connects to at least one covered superpeer. Now, we compute the probability that it connects to more than one covered superpeer.

Suppose that the peer already has one connection to a covered superpeer and now it needs to make $m-1$ more connections to superpeers. We can divide the rest of the $n_s - 1$ superpeers into two groups: the covered group (with size $p_s - 1$) and the uncovered group (with size $n_s - p_s$). Let us use variable c to count the number of new connections to the covered group. Since the $m-1$ more superpeers are randomly selected (without replacement) from these two distinct groups, we can see that c follows the hypergeometric distribution. Thus,

$$P(c \ge 1) = 1 - P(c=0) = 1 - \frac{\binom{p_s-1}{0}\binom{n_s-p_s}{m-1}}{\binom{n_s-1}{m-1}}$$
$$= 1 - \prod_{i=0}^{m-2} \frac{(n_s - p_s - i)}{(n_s - i - 1)}.$$

This value is very small when $p_s << n_s$. For example, when $m=2$, $P(c \ge 1) = \frac{p_s-1}{n_s-1} \approx 0$. When $m=3$,

$$P(c \ge 1) = 1 - \frac{(n_s-p_s)(n_s-p_s-1)}{(n_s-1)(n_s-2)} \approx \frac{2p_s}{n_s-2} \approx 0.$$

□

From Theorems 2 and 3, we can see that, when $p_s << n_s$, the probability of a leaf-peer connecting to more than one superpeer is very small. Thus, to cover p peers, the number of covered superpeers p_s is very close to the lower bound of $\frac{p}{(1+k_l)}$.

By considering the two extremes illustrated above, we give the upper and lower bounds of the number of messages used in a query task in Theorem 4.

Theorem 4. *To cover $p_s = \frac{p}{(1+k_l)}$ superpeers, the number of query messages ranges from $\frac{p}{(1+k_l)} - 1$ to $\frac{pk_s}{(1+k_l)}$, regardless of search mechanisms.*

Proof. In the superlayer, the queries are relayed among superpeers through various search mechanisms. Let us consider a network with five peers, which is shown in Fig. 3. The number associated with each link represents the latency between the two peers of the link.

Assume that peer O is the query source. The ideal search algorithm should only query each peer once. Therefore, to cover p_s superpeers, it can only use $(p_s - 1)$ messages, as shown in Fig. 3a. While, for an inefficient search algorithm such as blind flooding, some peers may be queried more than once with redundant messages, according to Gnutella protocol, no peer will relay the same query to the same link more than once. Thus, each link relays the same query at most twice. In

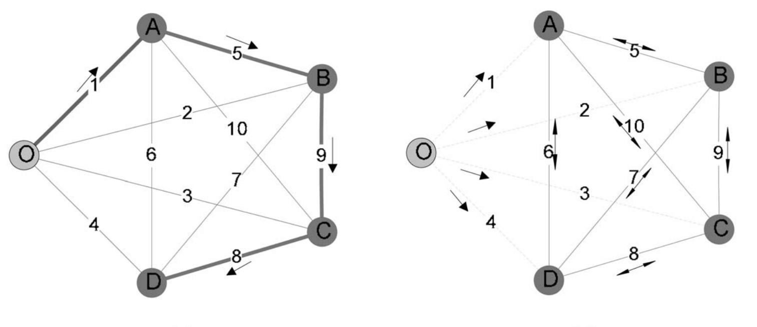


Fig. 3. Messages used to cover five peers. (a) Most efficient search. (b) Most inefficient search.

Fig. 3b, all the links among other peers except O will process the query twice.

For p_s peers with each peer connecting k_s other peers, the maximum number of links among them is $\frac{p_s k_s}{2}$, so the maximum number of messages is $p_s k_s$. Let $p_s = \frac{p}{1+k_l}$, we can prove the theorem. □

Now, we analyze the average Relay Workload part of W_{sp} and W_{on}.

Suppose that each query sent from superpeers eventually uses g messages among superpeers, which means that it will be relayed g times in the superlayer in total. Also, since each peer in the network initiates f queries per time unit and each superpeer receives $(1+k_l)f$ queries from itself and its leaf neighbors, the query frequency of the total network is $(1+k_l)n_s f$.

From Theorem 4, we know that the number of messages used by a query ranges from $\frac{p}{(1+k_l)} - 1$ to $\frac{pk_s}{(1+k_l)}$. Thus, the lower and upper bounds of the portions of the relay workload in W_{on} are given by

$$\begin{aligned} W_{on\_rw(\min)} &= (1+k_l)n_s f\left(\frac{p}{1+k_l} - 1\right) = n_s f(p-1-k_l) \\ &= \frac{nf}{1+\eta}(p-1-m\eta) \end{aligned}$$

and

$$W_{on\_rw(\max)} = n_s f p k_s = \frac{f p k_s n}{1+\eta}.$$

Since the relay workload is distributed among all the ns superpeers, on the average, Wsp_rw is $\frac{1}{n_s}$ of W_{on_rw} and has the lower and upper bounds:

$$W_{sp\_rw(\min)} = (p-1-m\eta)f \text{ and } W_{sp\_rw(\max)} = fpk_s.$$

3.2.4 Optimal Layer Size Ratio

In a network with n peers, we want to find a highly suitable layer size ratio, that is, an optimal η value. We deal with this problem by considering both W_{sp} and W_{on}.

Since the total workload is the sum of the three workloads discussed above, we have,

$$W_{sp} = W_{sp\_cw} + W_{sp\_qw} + W_{sp\_rw}$$

and

$$W_{on} = W_{on\_cw} + W_{on\_qw} + W_{on\_rw}.$$

Thus, for the most efficient search algorithm, we have the lower bounds of W_{sp} and W_{on} as

$$\begin{aligned} W_{sp(min)} &= \frac{m\eta}{t_l} + \frac{k_s}{t_s} + m\eta f + (p-1-m\eta)f \\ &= \frac{m\eta}{t_l} + \frac{k_s}{t_s} + (p-1)f \text{ and} \\ W_{on}(min) &= \frac{mn\eta}{(1+\eta)t_l} + \frac{n(m\eta+k_s)}{(1+\eta)t_s} + \frac{nm\eta f}{1+\eta} \\ &\quad + \frac{nf}{1+\eta}(p-1-m\eta) \\ &= \frac{n}{1+\eta}\left(\frac{m\eta}{t_l} + \frac{m\eta+k_s}{t_s} + fp - f\right). \end{aligned}$$

For the most inefficient search algorithm, we have

$$\begin{aligned} W_{sp(max)} &= \frac{m\eta}{t_l} + \frac{k_s}{t_s} + m\eta f + fpk_s \\ &= \left(\frac{1}{t_l} + f\right)m\eta + \frac{k_s}{t_s} + fpk_s, \\ W_{on(max)} &= \frac{mn\eta}{(1+\eta)t_l} + \frac{n(m\eta+k_s)}{(1+\eta)t_s} + \frac{nm\eta f}{1+\eta} + \frac{fpk_s n}{1+\eta} \\ &= \frac{n}{1+\eta}\left(\frac{m\eta}{t_l} + \frac{m\eta+k_s}{t_s} + m\eta f + fpk_s\right). \end{aligned}$$

Ideally, the optimal η value should minimize both of these two workloads. But, unfortunately, it is impossible to minimize these two workloads simultaneously. In fact, there is a trade-off between these two workloads. With more superpeers in the network, W_{on} will increase, but W_{sp} will decrease and vice versa.

Thus, we use a weighted workload, W, to consider both W_{sp} and W_{on}; and it is given by

$$W = \alpha W_{sp} + \beta \frac{W_{on}}{n}, \tag{1}$$

where α and β are weights of W_{sp} and W_{on}, respectively, and we set $\alpha + \beta = 1$. If a network concerns more on W_{sp}, we can set $\alpha > \beta$, otherwise, we set $\beta > \alpha$. Both α and β are preconfigured.

Since both W_{sp} and W_{on} are functions of η, by differentiating (1), we can obtain the optimal η value as

$$\eta' = \sqrt{\frac{B-C}{A}} - 1, \tag{2}$$

where, for the most efficient search algorithm,

$$A = \frac{m\alpha}{t_l}, B = \left(\frac{k_s}{t_s} + fp - f\right)\beta, \text{ and } C = \left(\frac{1}{t_l} + \frac{1}{t_s}\right)m\beta,$$

while, for the most inefficient search algorithm,

$$A = \left(\frac{1}{t_l} + f\right)m\alpha, B = \left(\frac{1}{t_s} + fp\right)k_s\beta, \text{ and}$$
$$C = \left(\frac{1}{t_l} + \frac{1}{t_s} + f\right)m\beta.$$

For a specific network, we assume that the values of α, β, m, and p are all given by the protocol and the values of k_s, t_l, and t_s can be obtained from each peer's previous experiences. Actually, these values are also evaluated by many studies [11], [9]. Therefore, on each peer, an optimal value of layer size ratio can be computed independently.

In KaZaA, superpeers are found to have 100-200 leaf-neighbors [12]. Since each leaf-peer normally keeps connections to two to three superpeers, the layer size ratio is expected to be several dozens. This value is very close to the values we computed using our workload model, where the layer size ratio is also about several dozens in Section 5.1.

4 Design of DLM

We propose a Dynamic layer management algorithm (DLM) that intends to deal with the following two problems, which are not trivial to deal with since no peer has the global knowledge of the network: One problem is how to maintain an optimal layer size ratio in a highly dynamic environment. The other problem is which peers should be promoted (from leaf-layer to superlayer) or demoted (from superlayer to leaf-layer) when the system has more or less superpeers than optimal. In other words, DLM should achieve two goals: 1) maintaining the size ratio of superlayer to leaf-layer and 2) keeping the peers with larger lifetimes and capacities as superpeers and 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 superpeers are heavily demanded, but the new joining peers all have low bandwidths. In this case, DLM needs to consider both goals and adaptively promote some leaf-peers, though not "powerful" enough, to superlayer. Moreover, to perform well under both stable and highly changeable network situations, DLM needs to dynamically adjust the promotion and demotion policies adaptive to the changing environments.

Ideally, the superpeers should be more powerful and with a longer lifetime than average peers. To measure the eligibility of a peer, we define two metrics, *Capacity* and *Age*. *Capacity* is defined as the ability of a peer to process and relay queries and query responses, while *age* is defined as the length of time up to now since a peer joined the network.

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 ith metric, and $w_i(d)$ is the weight of the corresponding metric. Examples of the metrics affecting the capacity are bandwidth, CPU speed, and storage space. We assume that a peer's capacity value does not change throughout its session and can be known at the time it is connected to the network. In this paper, 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.

The lifetime of a peer is the period of time in which the peer participates in the P2P network. The *age* is less than or equal to the lifetime of a peer by the definition. DLM will automatically promote the leaf peers with larger ages to superlayer and demote the superpeers 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, the more likely it is that the peer will live in the future. So, we use the *age* of a peer in DLM to predict its lifetime and use $age(d)$ to denote peer d's *age* value.

4.1 DLM-1

We present the first approach, DLM-1. In DLM-1, each peer first collects its neighboring peers' information, which includes the *capacities*, *ages*, and number of neighboring leaf-peers (for a superpeer neighbor). After processing the collected information, the peer will determine its own status. We describe DLM-1 in four steps.

4.1.1 Step 1: Information Collection

Peers first exchange information with their neighbors using messages. We design two pairs of messages which are employed by a pair of leaf-peers and superpeers. The first pair of the messages is *neigh_num_request* and *neigh_num_response* messages; *neigh_num_request* is sent from a leaf-peer d to a superpeer s to request the leaf neighbor number of s and we use $l_{nn}(s)$ to refer to this number. Message *neigh_num_response* is the response from the superpeer. The second pair of the messages is *value_request* and *value_response*, which are sent between a superpeer and a leaf-peer to query and respond to the leaf-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 is how often the peers exchange this information. Obviously, higher frequency means higher accuracy and more traffic overhead. 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 a time interval-based policy where peers exchange information periodically.

4.1.2 Step 2: Maintaining Appropriate Layer-Size-Ratio

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

Current layer size ratio can be easily calculated if the global knowledge is known, but, in reality, no peer has this global knowledge. However, it is trivial for a peer to collect its neighbors' information. Due to the randomness of the neighbor selection mechanism in superpeer systems, to some extent, the current numbers of leaf neighbors of superpeers can reflect the current layer size ratio. That is, if the superlayer size is too small, the average number of leaf neighbors of a superpeer 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, the value of k_l can be computed by using $k_l = m\eta$. Therefore, l_{nn}, the leaf neighbor number of some superpeer, 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, denoted as G. For a superpeer 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 superpeers that it has connected within a period of time T_l.

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

For a superpeer s, since it connects to some leaf-peers, it can directly use $l_{nn}(s)$, while, for a leaf-peer l, it uses the average l_{nn} value of the superpeers in $G(l)$. We let η denote the extent of inappropriateness of current layer size ratio compared to the optimal layer size ratio η and it is computed as $\mu = log(l_{nn}/k_1)$.

We can see that a positive value of μ, where l_{nn} is larger than k_1, means that there are too few superpeers in the system since superpeers have more leaf-peer neighbors than normal., while a negative value of μ, where l_{nn} is smaller than k_1, means that there are too many superpeers. 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.

4.1.3 Step 3: Scaled Comparisons of Capacity and Age

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

One straightforward way of comparison is to directly compare 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 superpeers, but the leaf-peers all have larger metric values than the current superpeers. The results of simple comparisons would forbid any leaf-peer being promoted; thereby, the system cannot adjust the layer size ratio at all.

DLM improves the direct comparison by using a "scaled-comparison." Since the capacity and age metrics are disjoint, that is, a peer with high-capacity does not necessarily have larger age and vice versa, we analyze these two metrics individually. In the 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 pseudocodes of the scaled-comparison are listed below.

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

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 superpeer, if it finds that the system needs more superpeers, 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 superpeers 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.

4.1.4 Step 4: Promotion or Demotion

For a leaf-peer l, if Y_{capa} and Y_{age} are small enough, it means that many superpeers in $G(l)$ have metric values "smaller" than it. Thus, l may assume that it has relatively large metric values and may decide to be promoted to superlayer, while, for a superpeer 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 relatively 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 superpeer. In promotion, the leaf-peer keeps its current connections to other superpeers. The scenarios before and after peer l's promotion are illustrated in Figs. 4a and 4b.

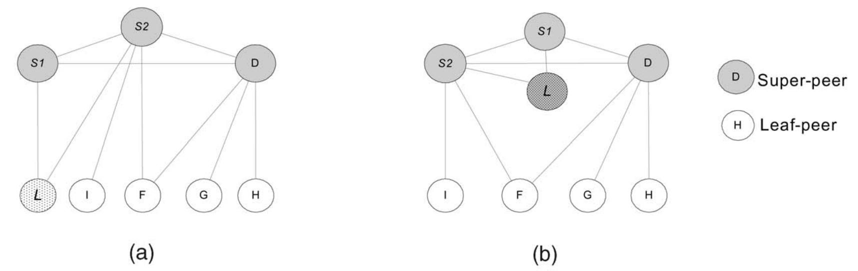


Fig. 4. Promotion of a leaf-peer. (a) Before promotion. (b) After promotion.

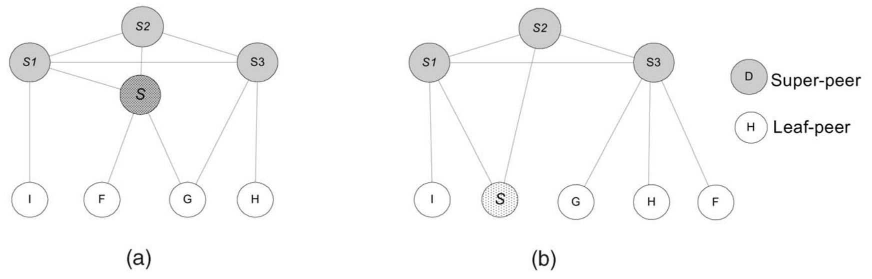


Fig. 5. Demotion of a superpeer. (a) Before demotion. (b) After demotion.

For a superpeer s, if Y_{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 superpeer only keeps m of its current connections to other superpeers and drops the connections to leaf-peers. The scenarios before and after peer s's demotion are illustrated in Figs. 5a and 5b.

The values of threshold variables, Z_{capa} and Z_{age}, are also adjusted according to the value of μ. When more superpeers are needed, superpeers 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 where there are too many superpeers, inverse measures will be taken accordingly.

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 the scaled comparison method, DLM achieves the second goal: keeping large-capacity and large-age peers on the superlayer.

4.2 DLM-2

In DLM-1, a superpeer estimates the current layer size ratio by monitoring the number of its leaf neighbors and a leaf-peer does so by monitoring the numbers of leaf neighbors of its neighboring superpeers. Because the numbers of leaf neighbors of different superpeers may vary greatly in the network, the estimation results might not be very accurate; therefore, it may not reflect the exact layer size ratio.

We propose letting neighboring superpeers exchange information in DLM-2 to more accurately estimate the system condition. In this approach, a superpeer computes the value of μ based on not only its own information, but also the information of its neighboring superpeers. A leaf-peer, in turn, can get more accurate information from its super neighbors through information exchanging. We propose three policies to exchange information among superpeers: **Pushing Exchange**, **Pulling Exchange**, and **Periodic Exchange**. In Pushing Exchange, each superpeer sends its calculated system information to superpeer neighbors whenever it runs DLM-1. In Pulling Exchange, whenever a superpeer runs DLM-1, it requests its superpeer neighbors to send back their system condition, while, in Periodic Exchange, each superpeer sends out its estimation result periodically. The Periodic Exchange method is much different from the first two. The performance of Periodic Exchange varies according to different values of T. Larger T means fewer exchanges and overhead. Smaller T means more exchanges and overhead, but more information obtained for each peer.

4.3 DLM-3

In the first two approaches of DLM, both superpeers and leaf-peers run the algorithm and can promote or demote. In DLM-3, only superpeers perform the estimation process. When one superpeer infers that the network has too many superpeers, it may decide to be demoted; this process is the same as that of DLM-1 and DLM-2. But, when it finds that more superpeers are needed in the network, it will inform the most eligible leaf neighbor and promote it to superlayer.

One issue in the promotion of leaf neighbors is the policy of selecting the most eligible leaf-peer. Similarly to DLM-1 and DLM-2, DLM-3 also uses the metrics of capacity and age in selection. We propose three different policies to select the most eligible leaf-peer: **Largest-Age**, **Largest-Capacity**, and **Weighted-Metric**. The first two policies are self-explained and a superpeer just selects the leaf-peer with the largest corresponding values. A similar lifespan-based mechanism is proposed in [4] to choose a peer's "friends," and it is very

TABLE 1
Simulation Parameters

Parameter	Value	Description
n	50,000	Number of peers in the network
n_l	48,780	Number of preferred leaf-peers
n_s	1,220	Number of preferred super-peers
η	40.0	Layer size ratio
m	2	Number of super-peer neighbors of a leaf-peer
k_l	80	Average number of leaf-peer neighbors of a super-peers
k_s	3	Average number of super-peer neighbors of a super-peers
t_l	3.5	Average duration time of leaf-peers
t_s	50	Average duration time of super-peers
f	0.3	Average number of queries of a peer per minute
p	3,000	Number of covered peers to ensure some fixed success rate

similar to our Largest Age policy, while, in the Weighted-Metric policy, we consider the two metrics, Capacity and Age, simultaneously. Since the two metrics of a peer are mostly disjoint, we propose measuring the peers' eligibility by defining a combined metric—Weighted Metric.

We first calculate the weighted metric value of a peer d by $WM_d = \gamma_1 Capacity(p) + \gamma_2 Age(p)$, where γ_1 and γ_2 are called Metric Weight Parameters, which are used to control the contribution weights of capacity and age, respectively. We also set $\gamma_1 + \gamma_2 = 1$. A superpeer then selects the leaf-peer with the largest WM value as the most eligible superpeer.

We propose one **Metric-Distribution** mechanism to set γ_i. Suppose that all the peers have the same values on one metric, this metric may not be considered to be an influential metric in the peer selection. Thus, its Metric Weight Parameter will be set to zero. On the contrary, if the values of another metric are more sparsely dispersed on these peers, this metric will be viewed as an influential one. Therefore, its Metric Weight Parameter will be assigned with larger value to reflect this diversity.

The Metric-Distribution mechanism can also be applied to problems with multiple metrics, although, in DLM, it only considers two metrics. Consider a peer d in a superpeer's Related-Set G and let us use $V_{r,d}$ to denote the value of metric r on d. The Metric-Distribution mechanism performs through the following steps.

First, we normalize the metric value to [0, 1] space; this is done by $V_{r,d} = \frac{V_{r,d} - V_{r,Min}}{(V_{r,Max} - V_{r,Min})}$, where $V_{r,Min}$ and $V_{r,Max}$ are the minimal and maximal values of metric r, respectively.

We then compute the standard variance of metric r,

$$\overline{\sigma}_r = \sqrt{\sum_d (V_{r,d} - E[V_r])^2},$$

where $E[V_r]$ is the normalized expectation of metric r. Last, letting γ_r denote the weight parameter for metric r, it is set as $\gamma_r = \frac{\overline{\sigma_r}}{\sum_i \overline{\sigma_i}}$.

4.4 Applying DLM to Multilayer Architectures

In general, the superpeer architectures are special cases of the k-layer architectures with k equals 2. The original flat architectures (for example, Gnutella v0.4) are also special cases of k-layer architectures with k equals 1. DLM can also be used on general k-layer architectures.

If a peer in layer i $(1 \leq i < k)$ has larger capacity and age values, it will be automatically promoted to the upper layer, say, $i + 1$. On the contrary, if a peer in layer i $(1 < i \leq k)$ has smaller capacity and age values compared with other lower layer peers, it will be demoted to the lower layer, say, $i - 1$. When both promotion and demotion are requested, the peer needs to compare the necessities of promotion and demotion and act correspondingly. Thereby, the algorithm can make sure that the peers with longer lifetime and larger capability are in higher layer.

5 Performance Evaluation

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 [16]. One client participates in 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 the average query frequencies of peer. The collected data are consistent with the data presented in previous studies [19], [20], [9]. We configure our simulation environments based on the collected data and these studies, which are shown in Table 1.

First, we test the workload model by measuring W_{on} and W_{sp} on varying η values. Using our proposed workload model, a network can determine an optimal layer size ratio between the leaf-layer and the superlayer. Then, we evaluate the effectiveness of the different approaches of

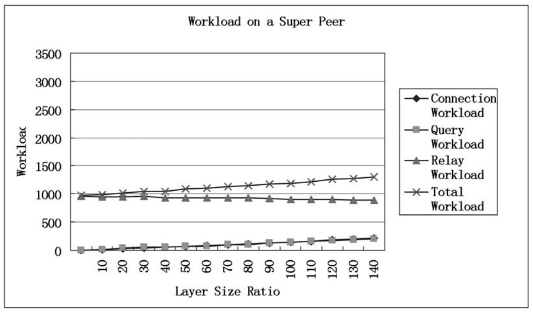


Fig. 6. W_{sp} of most efficient search.

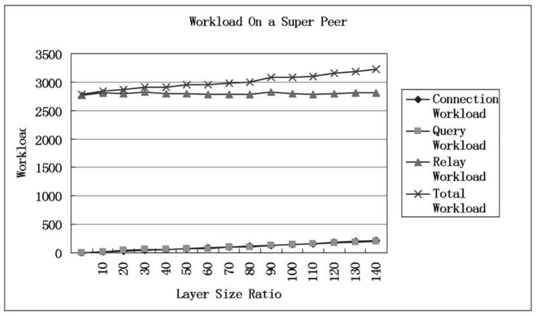


Fig. 7. W_{sp} of most inefficient search.

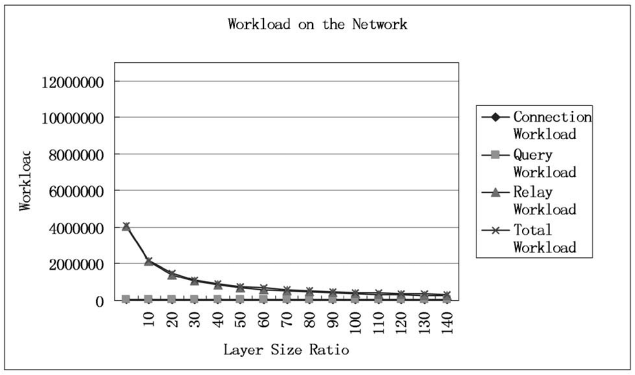


Fig. 8. W_{on} of most efficient search.

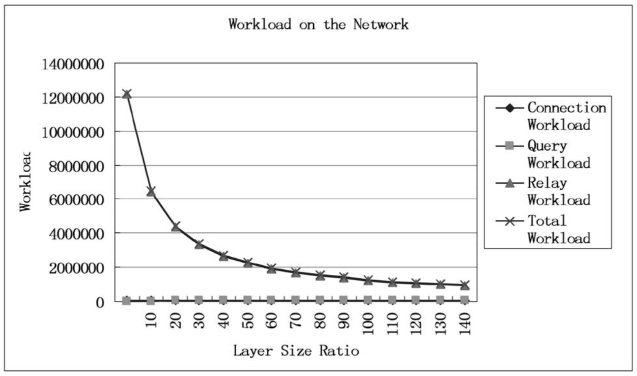


Fig. 9. W_{on} of most inefficient search.

DLM. We also compare the workloads incurred by DLM-1 and preconfigured algorithm under given success rates.

5.1 Evaluation of Workload Model

For the workload on a superpeer, Fig. 6 and Fig. 7 illustrate the connection workload, query workload, relay workload, as well as the total workload W_{sp} with varying values of η. Fig. 6 plots the minimum case of relay workload (for most efficient search) and Fig. 7 plots the maximum case (for most inefficient search). We can see that both the connection workload and query workload increase as η increases. For the relay workload, in the minimum case, the amount decreases slightly when η increases; however, in the maximum case, it almost says constant, regardless of different η values. In both cases, the total workload on one superpeer increases as η increases. Those results are not surprising since a larger η value means that each superpeer represents more leaf-peers.

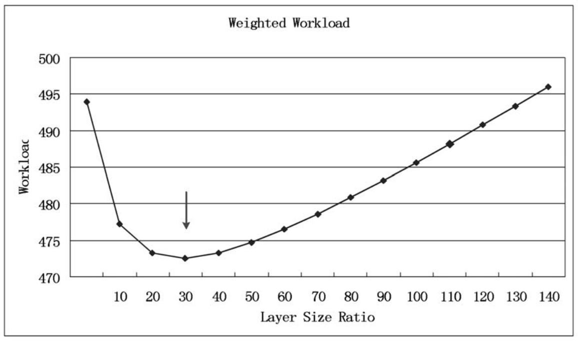


Fig. 10. Weighted workload of most efficient search $(\alpha = 0.5, \beta = 0.5)$.

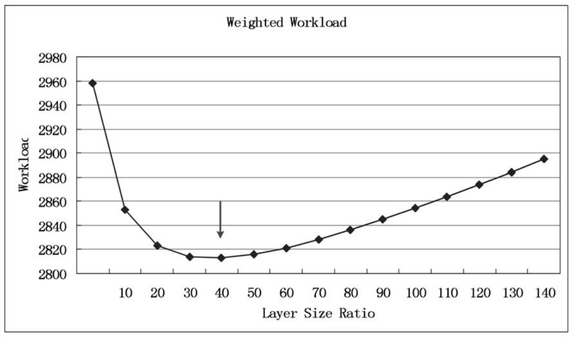


Fig. 11. Weighted workload of most inefficient search $(\alpha = 0.5, \beta = 0.5)$.

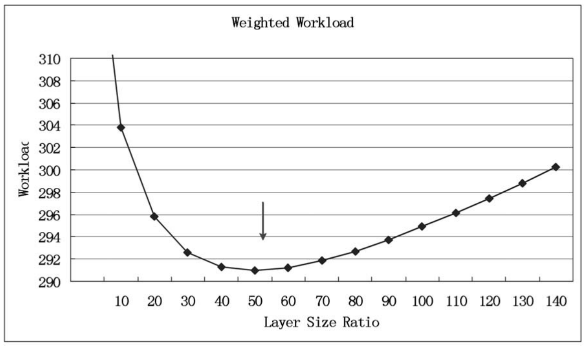


Fig. 12. Weighted workload of most efficient search $(\alpha = 0.3, \beta = 0.7)$.

For the workload on the overall network, as shown in Fig. 8 and Fig. 9, W_{on} decreases when η increases. We can see that the main part of the total workload comes from the Relay and, compared to the Relay Workload, the amounts of the Connection Workload and Query Workload are negligible.

To obtain the optimal layer size ratio, we first set $\alpha = \beta = 0.5$ and measure the weighted workload as η changes. The results are shown in Fig. 10 and Fig. 11.

With the parameters listed in Table 1, we compute the optimal value using (2) in the following way: For the most efficient search algorithm, we have $\eta_1' = \sqrt{\frac{B-C}{A}} - 1 \approx 38$, while, for the most inefficient search algorithm, we have $\eta_1' = \sqrt{\frac{B-C}{A}} - 1 \approx 51$. The optimal values we obtained from simulation are 35 and 45, respectively, and they are indicated in Fig. 10 and Fig. 11. We can see that the obtained values are very close to the theoretical ones.

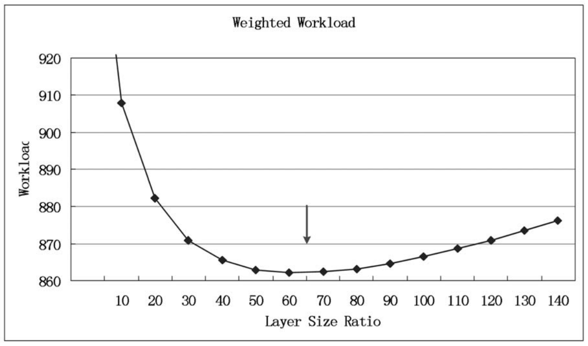
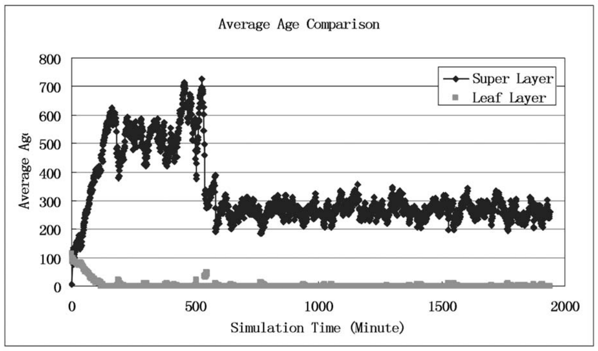


Fig. 13. Weighted workload of most inefficient search ($\alpha = 0.3, \beta = 0.7$).



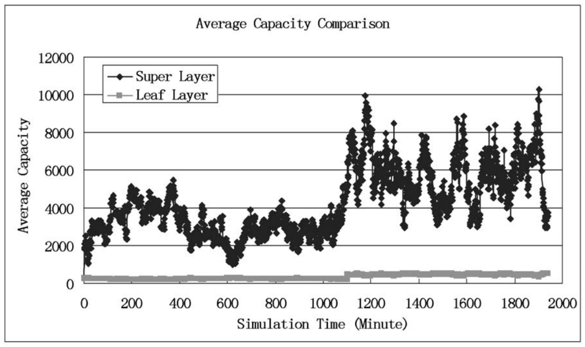


Fig. 15. Average capacity.

We also obtain the optimal layer size ratio under different values of α and β. Our simulation and computation show very close results. In Fig. 12 and Fig. 13, we present the simulation results when $\alpha = 0.3$ and $\beta = 0.7$. Compared with Fig. 10 and Fig. 11, we can see that both of the optimal values of η in Fig. 12 and Fig. 13 are slightly larger. Since larger β values indicate more weight on the network Overall Workload and larger η values mean less flooding overhead among superpeers, the optimal layer size ratio increases as α decreases.

5.2 Effectiveness of DLM-1

We evaluate DLM-1 under various simulation environments. We first simulate a *stable* network by assigning new peers with capacity and lifetime values based on previous studies [19]. In a *stable* network, the simulation starts "cold," i.e., without any peer. The size of the network increases, with new peers joining until it reaches the designated size. Then, with time passing, whenever a peer dies, a new peer

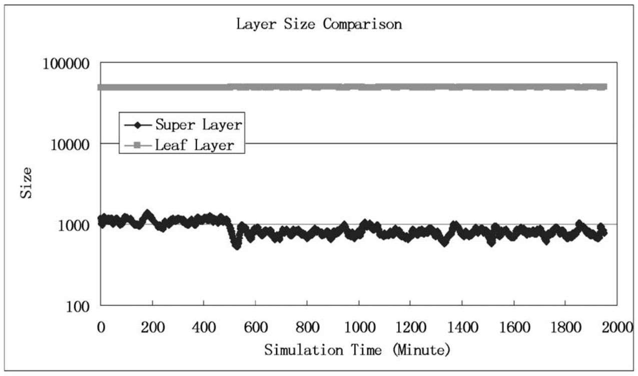


Fig. 16. Layer sizes.

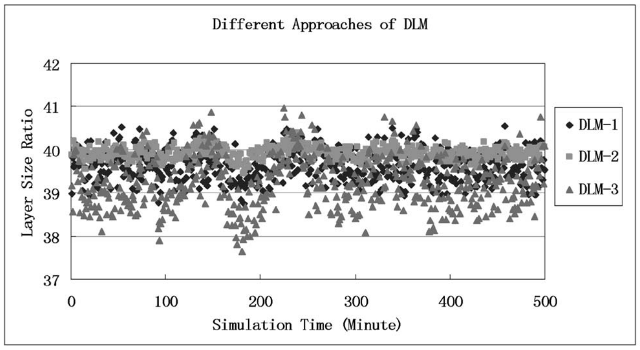


Fig. 17. Layer size ratio under different approaches of DLM.

is created and joins the network; therefore, the network size does not change. The new peer is always assigned to the leaf layer first and DLM will promote eligible leaf-peers to the superlayer. The lifetime and capacity values of a new peer are generated based on the distributions we observed and shown in previous studies. We then simulate the *dynamic* networks by varying the means of the capacity and age distributions for new joining peers. For these two cases, the network sizes are not changed. We also evaluate DLM-1 on networks with different sizes. The results are consistent, so we only present the results in dynamic environments in Fig. 14, Fig. 15, and Fig. 16.

Starting from the 300th time unit, the lifetime values of new joining peers are generated using the method as described before, but with halved mean values. Thus, from that time, the new peers have smaller lifetime values. The results in Fig. 14 show that DLM-1 adaptively promotes the peers with large capacities to superlayers and the average capacity value of a superlayer is always larger than that of a leaf-layer.

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. Fig. 15 plots the average ages of each layer. As expected, the age of the superlayer is much greater than that of the leaf-layer, regardless of the changing environments.

The layer sizes of the network are shown in Fig. 16; we can see that an almost constant ratio is maintained throughout the simulation process, even though the network environment is changing. Note that the Y-axis of Fig. 16 is logarithmic.

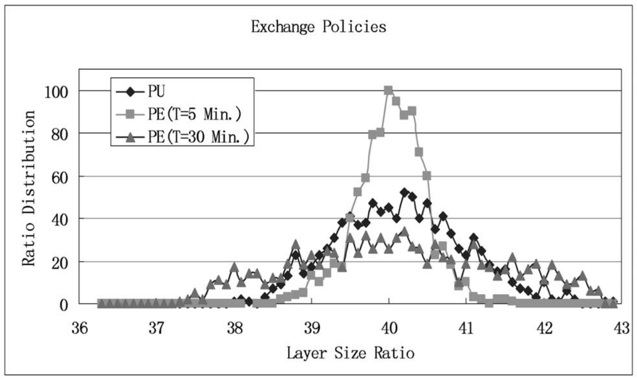


Fig. 18. Layer size ratio under different PE.

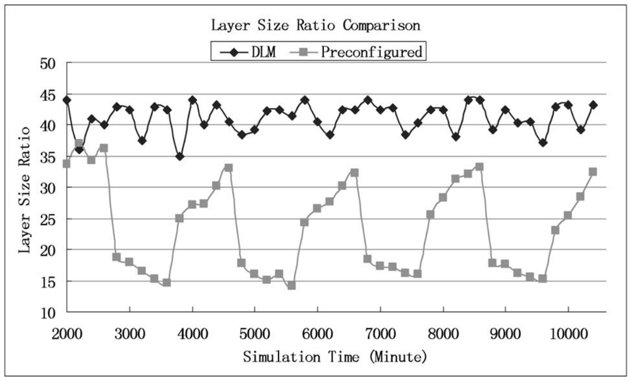


Fig. 19. Layer size ratios on the same success rate.

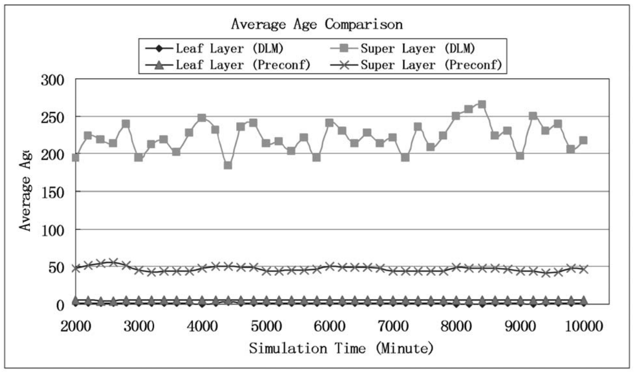


Fig. 20. Average age comparison.

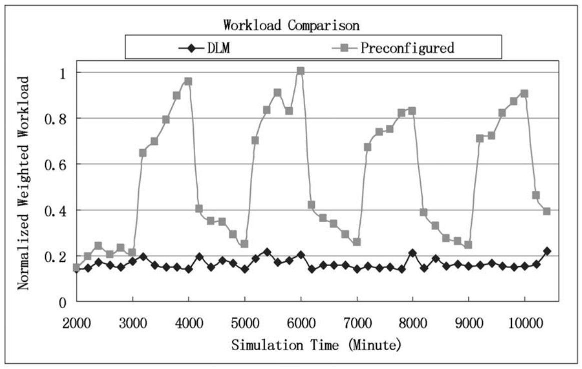


Fig. 21. Workloads on the same success rate.

5.3 Effectiveness of Different Approaches and Policies

We compare the effectiveness of different approaches of DLM and the results are shown in Fig. 17, which is the scattered graph of the layer size ratio values. As expected, DLM-2 performs the best since it employs the information exchange mechanism between neighboring superpeers. DLM-3 performs the worst as it only runs on the superpeers. Fig. 17 depicts the layer size ratio changes of the three approaches. From this figure, we can see that DLM-2 has the denser distribution, which stands for more stable values, while DLM-3 is the one with most unstable size ratio. The performances on the other two metrics, average capacity and average age, also show similar features.

In DLM-2, Periodic Exchange (PE) is an effective information exchange policy among superpeers. It ensures the system information to be exchanged between superpeers periodically. However, different T values have large effects on the performances. Fig. 18 shows the size ratio distribution curves of Periodic Exchange with different T values and Pushing Exchange (PU), the peak values corresponds to the optimal layer size ratio. We can see that, when T equals 5, the performance of Periodic Exchange is better than Pushing Exchange, while, when T equals 30, it is worse than Pushing Exchange.

5.4 Effectiveness of DLM-1 over a Preconfigured Algorithm

We compare DLM-1 with preconfigured 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-1 and a preconfigured algorithm.

The results are shown in Figs. 19, 20, and 21. We can see from Fig. 19 that the DLM-1 maintains the layer size ratio very well, while, in the preconfigured algorithm, the layer size ratio changes periodically. For the average ages of superlayer and leaf-layer, in DLM-1, they are sharply divided and the average age of the superlayer is much larger than that of the preconfigured algorithm, as shown in Fig. 20. The comparisons of capacity values show similar results.

We also compare the W_{on} of these two algorithms. We compare the two algorithms under the same success rates. The normalized results are shown in Fig. 21. The success rate is set to be 10 percent and we use an improved k-walker [13] as the search mechanism. From the results, we can see that DLM-1 has a very stable W_{on}, while the W_{on} of the preconfigured algorithm fluctuates a lot and is much larger.

6 Discussion on the Side Effects of DLM

Introducing DLM to superpeer 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 detail.

6.1 Overhead for Information Exchanging

To implement DLM, we propose adding two pairs of messages described in Section 4 to the existing superpeer P2P protocol. We argue that the additional overhead on delivering these two pairs of messages is negligible

TABLE 2
Peer Adjustment Overhead Analysis

Network size	Number of new leaf-peers	Number of demoted super-peers	Number of disconnected leaf-peers	PAO/NLCO (%) (m=2)
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%

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 a few bytes. As a result, these two pairs of messages are quite light-weight. 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 [19] 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.

6.2 Peer Adjustment Overhead

When a superpeer is demoted to be a leaf-peer, it needs to cut the connections to the leaf-peers and the connections to some superpeers because a leaf-peer only needs to keep a small number of links to superpeers, say, m superpeers. The leaf-peers disconnected by the demoted superpeer will try to connect to another superpeer instead. We call this kind of connection overhead *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 2.

We count the number of new peers, the number of superpeers 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 superpeers and incur the PAO. This process behaves like a new joining leaf-peer making connections to superpeers. The difference is that each disconnected leaf-peer only needs to create one new connection to another superpeer, while each new joining leaf-peer needs to create m new connections to superpeers. 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 *New Leaf-initiated Connection Overhead (NLCO)*. We also calculate the ratio of PAO and NLCO in Table 2.

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 superpeer connects to is more close to k_l due to the randomness of connections between peers. Therefore, the probability of misjudgments is also decreased. We believe that in the real-world P2P networks with millions of peers, the ratio of PAO to NLCO is trivial.

7 Conclusion

In this paper, we propose a workload model by analyzing the workloads on one superpeer as well as on the total network. Based on this model, we obtain an optimal layer size ratio that can minimize the weighted workload of the network. We also propose a dynamic layer management algorithm, DLM, which can adaptively elect peers and adjust them between superlayer and leaf-layer. We present three approaches and address some technical issues of DLM. DLM can also be easily extended to multilayer architectures. Our simulation results show that DLM can maintain a given size ratio of superlayer to leaf-layer and designate peers with long lifetime and large capacities as superpeers and the peers with short lifetime and low capacity as leaf-peers under highly dynamic network situations. DLM is completely distributed in the sense that each peer runs DLM independently. With the support of our proposed management mechanism, the quality of a superpeer system can be significantly improved.

Acknowledgments

This work is supported in part by the US National Science Foundation under grants CCF-0325760 and CCF-0514078 and by Hong Kong RGC DAG 04/05.EG01. Some preliminary results of this work were presented in the *Proceedings of the International Conference on Parallel Processing 2004*.

References

[1] "The Gnutella Protocol Specification 0.6," http://rfc-gnutella.sourceforge.net, 2002.
[2] "Regional Characteristics of P2P," http://www.sandvine.com, 2003.
[3] P. Backx, T. Wauters, B. Dhoedt, and P. Demeester, "A Comparison of Peer-to-Peer Architectures," *Proc. Eurescom Summit,* 2002.
[4] F.E. Bustamante and Y. Qiao, "Friendships that Last: Peer Lifespan and Its Role in P2P Protocols," *Proc. Int'l Workshop Web Content Caching and Distribution,* 2003.
[5] B.F. Cooper and H. Garcia-Molina, "SIL: Modeling and Measuring Scalable Peer-to-Peer Search Networks," *Proc. Int'l Workshop Databases, Information Systems, and Peer-to-Peer Computing,* 2003.
[6] N. Daswani, H. Garcia-Molina, and B. Yang, "Open Problems in Data-Sharing Peer-to-Peer Systems," *Proc. Ninth Int'l Conf. Database Theory,* 2003.
[7] Z. Ge, D.R. Figueiredo, S. Jaiswal, J. Kurose, and D. Towsley, "Modeling Peer-Peer File Sharing Systems," *Proc. IEEE INFOCOM,* 2003.
[8] Gnutella, http://gnutella.wego.com/, 2003.
[9] K.P. Gummadi, R.J. Dunn, S. Saroiu, S.D. Gribble, H.M. Levy, and J. Zahorjan, "Measurement, Modeling, and Analysis of a Peer-to-Peer File-Sharing Workload," *Proc. 19th ACM Symp. Operating Systems Principles,* Oct. 2003.

[10] K.-C. Hsiao and C.-T. King, "A Tree Model for Structured Peer-to-Peer Protocols," *Proc. Third Int'l Symp. Cluster Computing and the Grid,* May 2003.

[11] N. Leibowitz, A. Bergman, R. Ben-Shaul, and A. Shavit, "Are File Swapping Networks Cacheable? Characterizing P2P Traffic," *Proc. Seventh Int'l WWW Caching Workshop,* Aug. 2002.

[12] J. Liang, R. Kumar, and K.W. Ross, "Understanding KaZaA," http://cis.poly.edu/~ross/papers/UnderstandingKaZaA.pdf, 2004.

[13] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and Replication in Unstructured Peer-to-peer Networks," *Proc. 16th ACM Int'l Conf. Supercomputing,* 2002.

[14] D.A. Menasce and L. Kanchanapalli, "Probabilistic Scalable P2P Resource Location Services," *ACM SIGMETRICS Performance Evaluation Rev.*, vol. 30, pp. 48-58, 2002.

[15] Morpheus, http://www.morpheus.com/, 2004.

[16] Mutella, http://mutella.sourceforge.net/, 2003.

[17] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network," *Proc. ACM SIGCOMM,* 2001.

[18] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-Peer Systems," *Proc. Int'l Conf. Distributed Systems Platforms,* 2001.

[19] S. Saroiu, P. Gummadi, and S. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," *Proc. Multimedia Computing and Networking,* 2002.

[20] S. Sen and J. Wang, "Analyzing Peer-to-Peer Traffic across Large Networks," *Proc. ACM SIGCOMM Internet Measurement Workshop,* 2002.

[21] S. Singh, S. Ramabhadran, F. Baboescu, and A.C. Snoeren, "The Case for Service Provider Deployment of Super-Peers in Peer-to-Peer Networks," *Proc. Workshop Economics of Peer-to-Peer Systems,* 2003.

[22] A. Singla and C. Rohrs, "Ultrapeers: Another Step towards Gnutella Scalability," Version 1.0.26, http://rfc-gnutella.sourceforge.net/src/Ultrapeers_1.0.html, Nov. 2002.

[23] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," *Proc. ACM SIGCOMM,* pp. 149-160, 2001.

[24] J. Xu, A. Kumar, and X. Yu, "On the Fundamental Tradeoffs between Routing Table Size and Network Diameter in Peer-to-Peer Networks," *Proc. IEEE INFOCOM,* 2003.

[25] B. Yang and H. Garcia-Molina, "Efficient Search in Peer-to-Peer Networks," *Proc. Int'l Conf. Distributed Computing Systems,* 2002.

[26] B. Yang and H. Garcia-Molina, "Designing a Super-Peer Network," *Proc. 19th Int'l Conf. Data Eng.,* Mar. 2003.

[27] B.Y. Zhao, L. Huang, J. Stribling, S.C. Rhea, A.D. Joseph, and J. Kubiatowicz, "Tapestry: A Resilient Global-scale Overlay for Service Deployment," *IEEE J. Selected Areas in Comm.,* 2003.

Li Xiao received the BS and MS degrees in computer science from Northwestern Polytechnic University, China, and the PhD degree in computer science from the College of William and Mary in 2002. She is an assistant professor of computer science and engineering at Michigan State University. Her research interests are in the areas of distributed and Internet systems, system resource management, and design and implementation of experimental algorithms. She is a member of the ACM, the IEEE, the IEEE Computer Society, and IEEE Women in Engineering.

Zhenyun Zhuang received the BS degree in computer science from the Beijing University of Posts and Telecommunications, China, and the MS degree from Tsinghua University, China. He is a PhD student at the Michigan State University. His research interests are in the areas of distributed systems and computer network.

Yunhao Liu received the BS degree in automation from Tsinghua University, China, in 1995, the MA degree from Beijing Foreign Studies University, China, in 1997, and the PhD degree in computer science from Michigan State University in 2004. He is now an assistant professor in the Department of Computer Science at the Hong Kong University of Science and Technology. His research interests are in the areas of peer-to-peer computing, pervasive computing, distributed systems, network security, embedded systems, and high-speed networking. He is a member of the IEEE and the IEEE Computer Society.