Improving Query Response Delivery Quality in Peer-to-Peer Systems

Xiaomei Liu, *Student Member, IEEE*, Yunhao Liu, *Member, IEEE*, and Li Xiao, *Member, IEEE*

Abstract—Unstructured peer-to-peer (P2P) system is the prevalent model in today's P2P systems. In such systems, a response is sent along the same path that carried the incoming query message. To guarantee the anonymity of the requestor, no requestor information is included in the response message, and each node in the query's incoming path only knows its direct neighbors who sent the query request to it. This mechanism introduces response loss when any *one* node or connection in the path fails, which is a common occurrence in the P2P system due to its dynamic feature. In this paper, we address the response loss problem and show that peers' oscillation can cause up to a 35 percent response loss in an unstructured P2P system. We also present three techniques to alleviate this problem: the redundant response delivery (RRD) scheme as a proactive approach, the adaptive response delivery (ARD) scheme as a reactive approach, and the extended adaptive response delivery scheme to render ARD to function in an unstructured P2P system with limited or no flooding-based search mechanism. We have evaluated our techniques in a large-scale network simulation. With limited traffic overhead, all three techniques reduce response loss rate by more than 65 percent and are fully distributed. We have designed our techniques to be simple to develop and implement in existing P2P systems.

Index Terms—Peer-to-peer system, flooding search, query response, response path, response loss problem.

1 INTRODUCTION

AIMED at efficient utilization of Internet edge resources, peer-to-peer file sharing systems have received much attention since the emergence of Napster [3]. Today, peer-to-peer traffic has overwhelmed Web traffic as a leading consumer of Internet bandwidth [25].

P2P systems can be divided into three different categories: centralized, decentralized structured, and decentralized unstructured [17]. In centralized P2P systems, there is a central index server that maintains the file indices of each peer in the system. The centralized P2P system is the earliest model in P2P systems. Nevertheless, the central index server is a potential single point of failure and can be the bottleneck of the traffic. Decentralized structured P2P systems are generally based on the distributed hash table (DHT). The file placement is tightly controlled with network topology in such a way that makes subsequent searches easily satisfied. In unstructured P2P systems, files are randomly distributed among peers and, consequently, there is no correlation between file placement and network topology. Therefore, file locating is generally concluded based on the flooding search mechanism: Each peer makes duplicate copies of a query it receives and broadcasts to all its directly connected neighbors except the one that delivered the incoming message in each forwarding step. The duplication process is terminated only when the TTL value of the query is reduced

to zero, or a satisfying result has been found. This flooding mechanism is widely adopted in unstructured P2P systems due to its simplicity and robustness against node failure. Although a structured P2P system greatly reduces the overhead of file locating, its weakness in partial keyword search as well as the additional DHT maintenance cost impede its application in the real world. Accordingly, our research focuses on the unstructured P2P systems that are implemented and utilized more widely in reality.

Another benefit of the flooding search mechanism is that it provides anonymity for the query requestor: no information of query requestor is included in the query request message. The peers who received the request only know the query message ID and the neighbors who sent this message to it. To reduce the search traffic in the system, the response delivery process in the unstructured P2P system does not adopt a flooding mechanism. If a peer receives a query message and can satisfy this query, it will send a response along the same path the query came from: Each peer on the path sends a response to the first neighbor who sent the incoming query to it.

The unstructured P2P system is a highly oscillating system. Peer membership is ad hoc and dynamic due to the lack of a dedicated and centralized authority in the system. Although a flooding-based search mechanism is robust to node failures, the response message will be lost if any node on the response path fails. One may argue that loss of some response messages is not a major issue since multiple responses may be found for one query. Nevertheless, our simulations show that the loss is not negligible, as we found 35 percent of the responses are lost in a P2P system. In existing P2P systems, the most popular search method is the keyword search, which suggests the search results will not be exactly what a user expects most of the time. Increasing the number of search results for a query results in

- X. Liu and L. Xiao are with the Department of Computer Science and Engineering, 3115 Engineering Building, Michigan State University, East Lansing, MI 48824. E-mail: {liuxiaom, lxiao}@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 3 Dec. 2004; revised 19 Oct. 2005; accepted 26 Dec. 2005; published online 26 Sept. 2006.

Recommended for acceptance by K. Hwang.

For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-0295-1204.

increasing the likelihood that a user will find what he/she really needs. Also, it is obvious that all network resources consumed to find the response are wasted when a response is lost.

To reduce the response loss rate and utilize network resources more efficiently, we present three techniques: redundant response delivery (RRD), adaptive response delivery (ARD), and extended adaptive response delivery (e-ARD). All three techniques handle the response loss problem regardless of node failure or node departure. In RRD, the responder proactively sends back duplicate copies of the same response via different paths to avoid response loss. In ARD, peers in a response path automatically choose alternative paths in case of node failure or departure. The e-ARD mechanism extends the ARD mechanism with the introduction of backup response delivery agents (bRDA) to avoid response loss in P2P systems with limited or no broadcasting searching mechanisms.

The main contributions of this paper are as follows:

First, we identify the response loss problem, explore how serious this problem can be in an unstructured P2P system, and show why the problem can not be ignored in P2P system.

Second, we propose three techniques to remedy this problem, which cover the problem in the systems that do and do not adopt flooding search mechanism. To the best of our knowledge, we are the first to propose solutions for the response loss problem led by the P2P system itself.

Third, we deployed large-scale network simulations to investigate how serious the response loss problem is and also evaluate the effectiveness of the three techniques we proposed to remedy this problem.

The remainder of the paper is organized as follows: In the next section, we review background topics and related work. Section 3 presents proposed response returning techniques. Simulation methodology and performance evaluation are described in Sections 4 and 5, respectively. Finally, we summarize our work in Section 6.

2 BACKGROUND AND RELATED WORK

Most researchers consider that flooding search mechanisms provide fairly good capability in covering the node failure in unstructured P2P networks, and thus hardly consider the node failure issue during the query process in such systems, even though it is well known that the P2P system is highly transient. This robustness against node failure is of course true if a flooding mechanism is used in the entire search process. However, as we have discussed in the previous section, the dynamic nature of the system will lead to response loss since the flooding mechanism is not adopted in the response process. To the best of our knowledge, Chawathe et al. [8] are the only other researchers who also noticed the response loss problem in unstructured P2P systems and expected that their proposed optimization technique, which targeted to optimize the network topology by periodically adjusting the neighbor connections of each peer, may make the case worse. They proposed two methods to remedy this problem: the first is to stop forwarding the query earlier than to stop forwarding the response, and the second is to set a time threshold and reissue the query when it cannot be satisfied within the time limit. While the first method may help to make up the response loss

caused by their proposed technique of adjusting overlay topology, it cannot help the response loss problem caused by inherent P2P system oscillation because the first technique needs the cooperation of each peer. The second technique may reduce the unnecessary query reissuing traffic in the case of slow response, but it provides no benefit in reducing traffic and potential extra response delay caused by the lost responses. We also noticed the problem during our work in [16] and we realize that, with more and more optimization techniques being used to limit the query traffic, this problem will become more serious. This also motivates us to further investigate the problem.

Research has been done to provide fault tolerance in structured P2P systems [4], [22], [28], [30], [31], [32]. However, their target is different from ours in that fault tolerance needs to be provided in the whole search process in a structured P2P system, while, in our case, we only need to provide fault tolerance to the response message. We expect no modifications to the query request process and thus our technique can be seamlessly combined with existing query mechanisms. In addition, in structured P2P systems, each node is tightly controlled and node information can be used to construct fault tolerance strategies. However, in unstructured P2P systems, the only information a peer has is on itself and its neighbors. For example, each node in a Chord system [28] maintains a "successor-list" of its r nearest successors to provide fault tolerance during node failure. To do this, all the nodes need to be assigned a hashed key and orderly organized. This is impossible in unstructured P2P systems where nodes are completely autonomous and connections between nodes are formed randomly.

Message delivery in an ad hoc network may also fail due to node movement. However, routing mechanisms used in an ad hoc network [13], [19], [20] cannot be used to solve the problem in unstructured P2P systems. Unlike an ad hoc network, where the source node and destination node know each other, there is the requirement of anonymity during the query process in an unstructured P2P system. In addition, the delivery algorithm in an ad hoc network generally adopts broadcasting in a wireless network, which should be avoided in the response delivery process. To the best of our knowledge, no effective solution has been proposed to deal with the response loss problem incurred by the system itself in the unstructured P2P system.

Mechanisms have been proposed to provide fault tolerance for unstructured P2P systems against DoS/DDoS attacks [9], [10], [15]. Our research is different in that we focus on solving the response loss problem created by the system itself instead of outside attacks. Performance loss incurred by the third party attacks is beyond the scope of this paper. Yet, one can see from the rest of the paper, our mechanism can also combine with these approaches to reduce response loss in these cases.

3 RESPONSE DELIVERY OPTIMIZATION

In this section, we first discuss the response loss problem in detail and then present our techniques to solve the problem. Before the discussion of the response loss problem, we would like to define *forwarding neighbor* and *primary forwarding neighbor*.

Fig. 1. Response loss problem in P2P system.

Definition 1. During the query process, if a neighbor of peer p forwards a copy of this query message to p , then this neighbor is p 's forwarding neighbor.

Definition 2. Among all the forwarding neighbors of p , the first one that forwards a copy of this query to p is called primary forwarding neighbor.

In existing P2P systems, p always sends responses back to its primary forwarding neighbor. All of our three techniques are based on the observation that the amount of forwarding neighbors of a peer in a flooding-based query mechanism is more than one in most cases. However, the third proposed mechanism, e-ARD, also takes care of the case that only one forwarding neighbor is available during the query process.

3.1 Response Loss Problem

The P2P system is a highly oscillating system in that peers come and go frequently, and even when they are in the system, they may adjust their connections with a high frequency. Previous studies show that a peer's lifetime varies from less than 10 minutes in FastTrack [24] to 60 minutes in Gnutella and Napster [26]. Uptime of logical connections is obviously even shorter than individual peers, from 1 minute to less than 24 minutes [23]. In addition, many new techniques trying to improve the performance of P2P systems also require peers to adjust their connections to find better neighbors [14] or achieve optimized overlay topologies [8], [14], [16]. This further increases the dynamic nature of P2P systems.

To reduce the response traffic, responses will be sent back to the requestor along the query incoming path instead of by flooding. In addition, to keep requestor anonymity, no requestor information is stored in the requestor/response message. Peers on the response path only depend on the local knowledge of their primary forwarding neighbor to route the response. Therefore, the response message will be thrown away if the primary forwarding neighbor fails. For example, in Fig. 1, peer S issues a query and peer T has a response for the query. The response is sent back along the incoming query path $A \rightarrow B \rightarrow C \rightarrow D$. If any one of A, B, C, or D leaves after it forwards the query or any connection in any two of these nodes changes, a response loss occurs.

In this case, network resources for both response return and original query propagation are wasted.

3.2 Redundant Response Delivery (RRD)

Making a backup copy of vulnerable/critical components of the system is a common technique to improve the system's fault tolerance. The redundant response delivery (RRD) scheme also tries to alleviate the response loss problem via backup paths. The success of using backup paths here is based on whether the same query message can be forwarded to the same node from more than one path. This is intuitively true for a flooding query delivery system in a highly connected network. It is also attested to by the result of our simulation: Without specifying how the query is transferred except flooding and network topology settings being kept consistent with those of the real world, RRD does significantly reduce response loss rate compared with the original response delivery mechanism. What needs to be noted here is that RRD is totally different from a flooding/broadcast delivery system in that in the RRD scheme, only the responder issues responses to multiple selected neighbors, while other nodes in the path will only forward the response to its primary forwarding neighbor. In this sense, RRD is more like the k-walker mechanism discussed in [11], [17]: The requestor distributes several walkers in the system; each will detect the node that can satisfy the query via the DFS mechanism. However, nodes in RRD forward responses according to the recorded path instead of a randomly chosen neighbor.

We illustrate RRD here with an example. Fig. 1 supposes that peer S floods a query, and peer T has a response for the query. According to the Gnutella 0.6 protocol [1], if peer A is T's primary forwarding neighbor, T will send back the response via A and discard the same query message from other forwarding neighbors, such as G, M, and F. The path for T to send back a response to S is $A \rightarrow B \rightarrow C \rightarrow D$. If any of the peers on the path fails or leaves, the response will be lost. Instead of returning a response through one path, the RRD scheme adds one or more redundant response paths. For the same example, in addition to sending a response back immediately to A, T also selects one of the other forwarding neighbors (F, G, and M) as back up neighbors with redundancy probability γ , and sends a response back

through each of the selected neighbors. All other peers except the responder will drop response messages with the same message ID previously received.

The probability γ is a performance control factor that will affect the performance gain of RRD. Assume a node in the system has c neighbors. There are h_j nodes in each path starting from neighbor j . The failure probability of each node is p_{ji} . In addition, assume there is no overlap among these paths. The probability that a response returned by this node is lost is

$$f_{loss} = \prod_{j=1}^c \left(1 - \gamma \times (1 - P_{ji}) \sum_{i=1}^{h_j} \right).$$

It is obvious that the increase of γ results in the decrease of f_{loss} . The user can reduce the probability of response loss by setting a larger value of γ . However, f_{loss} does not reduce linearly due to the existence of path overlap and the variance of p_{ji} . In order to further investigate the influence of γ in RRD's performance, we deploy a series of experiments on it and show the results in Section 5.3.1. RRD creates very limited computation overhead for local nodes in choosing multiple neighbors to send back the responses. Nevertheless, it creates extra network traffic overhead, which is increased with the increase of redundant paths, if we ignore the path length, path overlap, and traffic variations among paths. However, the response traffic along one path is very limited, in order of $O(1)$ and restricted by the maximum TTL, which is suggested as 7 in [1]. The extra response traffic along redundant paths is also limited and still in order of $O(1)$, restricted by maximum TTL and γ .

3.3 Adaptive Response Delivery (ARD)

In the adaptive response delivery scheme, peers deliver the response to a different forwarding neighbor when the primary forwarding neighbor fails.

To adaptively deliver response upon node failure, each peer keeps a forwarding neighbor list for each query message within a certain period of time. The format of each item in the forwarding neighbor list is $\langle \text{Message ID}, \text{Forwarding Neighbor} \rangle$. The primary forwarding neighbor of this peer will not be recorded in its forwarding neighbor list.

The basic idea of ARD is as follows: If a peer in a response path finds that the next hop cannot be reached, he will check his forwarding neighbor list to see if other neighbors also forwarded the same query message to him. If so, he will select in the list the first that arrived and reroute the response to this forwarding neighbor. Otherwise, he will send a failure message with the query message ID and the information about the unreachable neighbors to his previous hop who will try to reroute the response. The Gnutella 0.6 protocol suggests that the TTL of a response should be set to at least the hop value of the responding query plus 2 [1]. ARD needs to set a larger TTL in order to route the response back to the initiator. The response message is terminated only when it returns to the originator or its TTL value is reduced to zero. Take Fig. 2 as an example for ARD. Peer A issues a query that is flooded to many peers. Peer P receives his first copy of the query from F and finds that he has a response for the query. Assume the query path for the

Fig. 2. Adaptive response routing.

query that first reaches P is $A \rightarrow B \rightarrow F \rightarrow P$. We consider the case where B fails or leaves after he forwarded the query to F. When F receives a response from P and tries to send the response to B, F will find that he cannot reach B. With ARD, F will first check his forwarding neighbor list for this query and select from the list the first to arrive. In this example, F chooses G as his new forwarding neighbor. When G receives the response, he again finds that B is nonreachable and picks E to return the response. Peer E will eventually send the response back to A via C or D depending on which one first forwards the query message to E. Another case is that both B and G leave or fail. Peer F will inform his previous hop, P, that he cannot deliver the response through all his forwarding neighbors, i.e., B and G. Peer P will pick H to reroute the response through. Although G is also in P's forwarding list for this query, P will not select G since P has been informed by F that G is not reachable.

The update frequency of the forwarding neighbor list should also be considered. Update frequency is $1/T_{wait}$ if we define T_{wait} as the average lifetime of a forwarding neighbor record in the forwarding list. A forwarding neighbor record will be removed from the list if T_{wait} expires. Therefore, it is important to select the proper value of T_{wait} . If T_{wait} is set too low, a forwarding neighbor record may be removed before the response is sent back to this peer and ARD will not be very effective. If T_{wait} is set too high, the overhead for each peer to keep the forwarding neighbor lists will be very high. The value of T_{wait} is related to the average response time that is defined as the time difference from when a responder sends a response to when the initiator receives the response.

We will show that ARD, a simple resilient forwarding technique, is able to increase response success rate with low traffic cost, especially when many peers fail in a system. Compared with RRD, the additional traffic overhead in ARD is small because ARD only reroutes a response if necessary. The simulation shows that the traffic caused by the adaptive delivery algorithm is close to that of the original response delivery mechanism.

One may also be concerned that ARD may perform poorly in the case that only a few or none of the peers have multiple forwarding neighbors. However, like we mentioned before in the case of RRD, this is rare in the unstructured P2P system where the flooding mechanism is being used. Even when this case happens, a peer can return the response to the node that delivered it, which can

TABLE 1
Type II Request Message

	Minimum Speed	bRDA Addr	Search criteria	Extension blocks
Byte Offset	0	1 2	5 6- end with 0x00	Optional

TABLE 2
Type II Responder Message

	Number of Hits	Port	Responder Addr	Speed	bRDA Addr	Result Set
Byte Offset	0	1 2	3 6	7 10	11 14	15-

provide a second chance to the response message by rerouting it to some other peers.

3.4 Extended Adaptive Response Delivery (e-ARD)

The success of RRD and ARD relies on how many duplicate copies of a query message can be sent to one node via its different neighbors. Neither RRD nor ARD will function effectively in the case where only one forwarding neighbor is available for a peer. Although this situation rarely occurs in most of the existing unstructured P2P systems, it may happen more frequently with the adoption of new techniques targeted to cut query traffic overhead by removing unnecessary node connections and limiting query message duplication.

To address this limitation, we have extended our ARD technique and made it effective even in the case that there is only one forwarding neighbor available for a peer in the system. We call this new technique *extended Adaptive Response Delivery mechanism* (e-ARD). In the e-ARD mechanism, an IP address used for adaptive response delivery is appended to each query message. When the next hop neighbor in the response transfer path fails, the peer can, in addition to forwarding the response to an alternative neighbor, forward the response to the node of this IP address. One important issue that cannot be ignored here is that the requestor anonymity must be maintained during the process, that is, with this additional information of a backup IP address, a third party cannot tell the identity of the query requestor. Therefore, we cannot simply append the address of requestor to the query message. In e-ARD, we achieve the requestor anonymity by introducing the backup response delivery agent (bRDA).

3.4.1 Backup Response Delivery Agent (bRDA)

In the e-ARD scheme, we define a type II query request/response message shown in Tables 1 and 2 by adding a new field of bRDA address based on the query request and response message specified in Gnutella 0.6 [1]. During the query process, type II request/response messages will be issued. When a requestor makes a query request, it will put its own address in the field of the bRDA address of the request message and broadcast it to all its neighbors. The peer who receives the query will decide with a wrapping probability of α whether it will replace the bRDA's IP address in the query message with its own IP address. We call the node that decided to append its own IP address to the query message the *backup response delivery agent* (bRDA). As a requestor always appends its address to the query message, a requestor is a bRDA.

When a bRDA appends its own IP address in the query message, it also stores the old IP address in the query message in its forwarding neighbor list. The forwarding neighbor list of each node in e-ARD is also extended to add the address of the previous bRDA. Each entry in the forwarding neighbor list is <message ID, pre-bRDA address, forwarding neighbor 1, forwarding neighbor 2, etc>. For the node which is not the bRDA, the pre-bRDA address of the related record is NULL.

A responder will copy the bRDA address of the received query to its response message. During the response delivery process, a node will check its neighbor list and see if it is the bRDA of the related message. If it is, it will remove the IP address from the response message and append the previous bRDA address stored in its forwarding neighbor list to the response message. Here, update frequency of the forwarding neighbor list is also an important factor for the effectiveness of e-ARD. The impact of this update frequency on the average query response time is comparable to that in ARD.

3.4.2 Wrapping Probability α

It is important to determine a proper value of wrapping probability. If it is too big, most of the nodes in the query-incoming path will become bRDA. This will consume too many local resources and reduce the response return efficiency. One extreme case would be to set α as 1. This reduces e-ARD to ARD since the backup path address appended to the query message will be just the address of the primary forwarding neighbor address of each peer. On the other hand, if α is too small, then one can always consider that the bRDA address in the query message is the address of the requestor and, thus, cannot guarantee the anonymity of the requestor. Previous studies show that the older a peer is, the longer it is expected to remain in the system [7]. With the above considerations in mind, the definition of α should satisfy the following requirements:

- There is one lower bound α_0 and one upper bound α_1 for α such that $\alpha_0 \leq \alpha \leq \alpha_1$.
- α increases as peer lifetime increases.
- The accelerated speed of α decreases as peer lifetime increases.
- Two peers with similar lifetime should also have similar values of α .

Therefore, we define α as:

$$\alpha = \alpha_0 + (\alpha_1 - \alpha_0) \times \left(1 - \frac{c}{\phi(T_{on}) + c}\right) = \alpha_1 - \frac{c(\alpha_1 - \alpha_0)}{\phi(T_{on}) + c},$$

Fig. 3. Tuning alpha with different nonlinear function of T_{on} and C value. (a) Alpha under different nonlinear functions (assume $C = 70$) and (b) scope of alpha varies upon different C values.

where α_0 is the lower bound, α_1 is the upper bound, and c is a fixed value that can be set by the user to further decide the distribution of α across $[\alpha_0, \alpha_1]$ given the scope of T_{on} .

To choose α_0 , we introduce f as the probability of the bRDA address being changed at least once by any of the nodes in the query path. If we assume that it takes k ($k > 1$) hops for a query to be satisfied, then $f = 1 - (1 - \alpha)^k$. Our goal is to find out the minimum value of α that makes f close to 1. It is obvious, however, that f monotonically increases respect to α . Therefore, we consider that f is close enough to 1 if it is greater than 90 percent. Considering that the TTL of a query message in most of the unstructured P2P systems is set as 7, the lower bound of α can be 0.32. To make the analysis simpler, we assume that the node lifetime is not less than 1 minute, and take 0.35 as the value of α_0 . To further test out the lower bound, we consider a case where all nodes in the query path take 0.35 as the value of α and that query path length is only as short as three hops. We can see f will be 0.578 which means there is still a fairly large chance for the bRDA address to be changed. It is obvious that the upper bound of α_1 should be a value of (0.5, 1). We thus recommend α_1 of 0.75 which is in the middle of (0.5, 1). Replacing α_0 and α_1 with 0.35 and 0.75, we can simplify the formula to $\alpha = 0.75 - \frac{0.4c}{\phi(T_{on}) + c}$.

Without the second requirement, $\Phi(T_{on})$ would be a linear function of T_{on} . With the last requirement, we need to add a nonlinear factor of T_{on} to tune the acceleration of α . $\Phi(T_{on})$ thus turns into a function with the format of $T_{on} \times \Delta(T_{on})$, with $\Delta(T_{on})$ being the nonlinear function of T_{on} . As research indicates that the average peer lifetime ranges from less than 10 minutes to 60 minutes [24], we check the power function, log function, and exponential function with T_{on} from 1 to 60. We show that the trend of how α increases under different nonlinear functions in Fig. 3a, for example, assume $c = 70$. It is obvious log function is a good choice in that it does not increase as fast as exponential and power function, while not as slow as linear function. From Fig. 3b, we can observe how different c values affect the scope of α . As the value of peer lifetime in our simulation is from 1 minute to 60 minutes, we take $c = 70$, which renders α fall within the scope of [0.36, 0.68].

3.4.3 Adaptive Response Delivery

When a responder forms a response message, it will insert the bRDA address it obtained from the query message in the response message. When a node receiving the response message finds that its primary forwarding neighbor for this response message fails, one of three situations occurs: 1) There are other neighbors in its forwarding neighbor list and they are still alive. The node will choose one of these available forwarding neighbors according to the ARD and forward the response back through it. 2) There are no other forwarding neighbors for this response or all the neighbors on list for this query are offline. The node will use the bRDA address appended to the response message to build a UDP connection to this agent and forward the response message to it. 3) The backup response delivery agent itself is failed, behind a firewall or any other circumstances that the UDP connection cannot be built. In e-ARD, the node informs its previous hop node and asks it to choose other node to forward the response message as in ARD. At the same time, a backup response delivery agent will always need to replace the IP address in the response with its stored IP address when it receives a response. (While a TCP connection may still be built up via a mechanism similar to the push operation in Gnutella network in this case, we do not recommend such an operation since the cost to create such kind of connection is too expensive for a one time communication.)

Fig. 4 illustrates the effect of the backup delivery agent in an adaptive response routing of the e-ARD scheme. P and K, which fail during the response return process, are the primary forwarding neighbors of peer O and N, respectively. In Fig. 4a and Fig. 4b, O is on the query incoming path. Therefore, whether O is a bRDA or not, it always sends a response to a bRDA (here is A) which is also on the query incoming path when no other forwarding neighbor is available for O. In Fig. 4c and Fig. 4d, O sends a response to its forwarding neighbor N, which is not on the original query incoming path. In Fig. 4c, N is not a bRDA itself and, thus, it still sends a response back to agent A, whose address is appended in the response message. The response message has been rerouted to the original query incoming path since the bRDA address appended to the response message is the address of the agent that is in the original query incoming path. In Fig. 4d, N itself is a bRDA. It

Fig. 4. Extended adaptive response delivery, effect of backup response delivery agent.

changes the bRDA address field according to the bRDA address stored in its forwarding neighbor list and sends a response message to M accordingly. Here, the response message did not reroute to the original query incoming path although the appended bRDA address in the message is also the address of an agent who is in the original query incoming path. In all cases, the length of the response path is reduced due to the introduction of bRDA.

The e-ARD mechanism can be used in P2P systems adopting a DFS-based search mechanism or other non-flooding-based search technique such as k-walker. In addition, forwarding responses to a backup response delivery agent in e-ARD can reduce the number of hops in a response return process and, thus, further cut both response time and traffic cost in the response process. This will partly pay for the overhead of creating a UDP connection.

4 SIMULATION METHODOLOGY

We have constructed a large-scale network simulation to evaluate the performance of our techniques. Both Gnutella [1] and KaZaA [2] are popular unstructured P2P systems. As the documentation about KaZaA is not available to the public, we deploy the simulation based on a Gnutella-like context.

4.1 Performance Metrics

We evaluate the effectiveness of RRD, ARD, and e-ARD via different performance metrics. The first important performance metric we evaluate, represented by *response return rate*, is the effectiveness. Response return rate is defined as the ratio of the number of responses returned to the requester (some responses may be lost in their backward paths) to the number of all responses found for the query.

The second metric we evaluate is the efficiency of the proposed mechanisms, which is measured by *response traffic cost*. The response traffic cost is the traffic of sending a response message from a responder to a query initiator. The response traffic cost is computed as a function of consumed network bandwidth and other related expenses. Specifically, we assume all messages have the same length in this

work. When messages traverse an overlay connection during the given time period, the traffic cost (T_c) is given as: $T_c = M \times L$, where M is the number of messages that traverse the overlay connection, and L represents the number of physical links underlying the overlay connection. The target of all optimization mechanisms is to achieve high response return rate with low cost.

The third performance metric we evaluate is the quality of service: Here, we refer the quality of service (QoS) to whether the users are satisfied with the service of the system. We use *response time*, which is commonly used in P2P researches, to evaluate the QoS performance of the system. The response time is defined as the time from when a query is issued by a requestor to when the response is received by the requestor. We compute the delay of a logic link as follows: First, we take the average delay of the physical links as the basic unit in the simulation, and compute the unit-less raw “delay” of a logic link. Assume the mean delay for all the physical links is $E(T)$. If the sum of the delays of physical links under a logical link is $Sum(T_i)$, the raw delay of a logical link is $Sum(T_i)/E(T)$. Next, to map the P2P network latency into our simulation, we generated a group of overlay link delays according to the observation in paper [26]. We then sort the overlay link according to their raw “delay” and assign the generated delay to each link.

4.2 Parameter Settings

We summarize the system configuration in Table 3. During each group of experiments in the simulation, we may change the value of some parameters to test the performance of three techniques under different circumstances. Note that for the network size, we have done simulations based on network sizes of 2,000, 3,000, 5,000, and 8,000. The results from networks with different sizes are consistent which confirms that our techniques are scalable. Here, we show the simulation results from a logical network size of 8,000.

4.3 Simulation Framework

P2P networks are overlay networks. To capture the overlay feature, two types of topology, physical topology and logical topology, are generated in the simulation. The physical topology is generated to represent the characteristics of the Internet under the P2P system. The logical

TABLE 3
Parameter Settings of the Simulation

Name	Default	Description	Name	Default	Description
Logical network size	8000	Number of peers in the logical network	Redundancy probability	0.2	Probability of responder to select each neighbor as redundant response backward neighbor in RRD
Neighbors per node	6	Average number of neighbors each peer has	Wrapping const	70	Constant to adjust accelerator of wrapping probability α in e-ARD
Query rate	0.3	Average number of queries each peer issues per minute	Forwarding neighbor list uptime	Average response time	Time forwarding neighbor record expires
Peer lifetime	10	Average peer online time (minutes)			

topology represents the overlay P2P topology built atop of the physical topology. All P2P nodes are in a subset of nodes in the physical topology. The communication cost between two logical neighbors is calculated based on the physical shortest path between this pair of nodes.

Network topology and density have a large impact on how many neighbors will send duplicate query messages to a peer. The network topology in the simulation should be consistent with the real network topologies to accurately evaluate the performance of the proposed mechanisms. Previous studies show that both large-scale Internet topology and P2P overlay topologies follow the small world and power law properties, and topologies generated using the AS Model have such properties [26], [29]. BRITE is one of the topology generation tools that provide the option to generate topologies based on the AS Model [18]. Using BRITE, we generated a physical topology of 22,000 node and based on it, four logical topologies with 2000, 3000, 5000, and 8,000 nodes. To check network with different connectivities, we varied the average number of neighbors of each node (of logical topology) from 4 to 10.

The unstructured P2P system with flooding search mechanism is the prevalent model of existing P2P systems; accordingly, we evaluate our techniques mainly based on category of systems. Observations presented in [12] point out that the object popularity distribution in a P2P system does not follow a Zipf distribution like that of World Wide Web objects mentioned in [5], [6]. We simulate the flooding search process via executing a Breadth First Search (BFS)-based algorithm from a specific node. A search operation is simulated by randomly choosing a peer as a requestor and requesting keywords according to the distribution described in [12]. As for the query dispatch frequency, we set every node in the system to issue, on average, 0.3 queries per minute, which is calculated from the data collected by Sripanidkulchai [27]. For instance, 12,805 unique IP addresses issued 1,146,782 queries in 5 hours.

Based on generated P2P network topologies, we simulated the joining and leaving behavior of peers via turning on/off logical peers. The peer lifetime is generated according to the distribution observed in [26]. The lifetime is decreased by one with each second, and once a peer's lifetime reaches zero, it will leave the following second. During each second, there are a number of peers leaving the system. To keep the power law property during node leaving and joining processes, we randomly pick up (turn on) the same number of peers from

the network to join the overlay system. For each peer, a *maximum-neighbor-connection* is predefined following power law. Each peer is required to keep the number of his neighbor connections no greater than his *maximum-neighbor-connection* during the simulation.

5 PERFORMANCE EVALUATION

Simulation results about performance evaluation are presented in this section, starting with the performance evaluation in a Gnutella like context environment and followed by the performance evaluation of e-ARD in a P2P system with DFS search mechanism.

5.1 Response Return Rate

Fig. 5 compares response return rates of a Gnutella-like system, RRD scheme, ARD scheme, and e-ARD scheme with peer uptimes ranging from 10 minutes to 60 minutes. The query frequency of each node is 0.3 queries per minute. The simulations are deployed in a 60 minute period with data collected every 20 seconds. This process is repeated multiple times. We use the average response return rate based on these collected data in the evaluation. Results in Fig. 5 show that Gnutella-like systems suffer from 35 percent response loss when the average peer uptime is 10 minutes. RRD, ARD, and e-ARD can increase the response return rate by up to about 35 percent, 47 percent, and 51 percent, respectively, compared with Gnutella-like systems. The performance of the e-ARD scheme is better but close to ARD, while both of them achieve a near perfect response return rate. The reason may lie in that in a Gnutella-like

Fig. 5. Response return rate versus peer lifetime.

Fig. 6. Response return rate versus query frequency.

system, most of the nodes have multiple query forwarding neighbors and can forward a query to other forwarding neighbors instead of bRDA in the e-ARD scheme.

Given that the peer lifetime is 10 minutes, Fig. 6 compares the response return rates of different schemes versus the average query frequency, and shows that the response return rate is not sensitive to query frequency. The relationship between response return rates and the average number of response neighbors per node in the Gnutella-like system and ARD are presented in Fig. 7. We can observe that the return rate increases as the average number of neighbors increases in a Gnutella-like system: The increase of the average number of neighbors suggests a network with higher density, which eventually results in shorter response return path. If we assume there are k nodes between requestor and responder in a query response path, with the same failure probability σ , the probability that no node failure happens is $f = 1 - (1 - \sigma)^k$. Therefore, a shorter response return path will result in lower probability of peer failure during the response return process and, thus, a higher response return rate. Nevertheless, the response return rate of ARD is not sensitive to the average number of neighbors due to its near optimal performance. The effectiveness of e-ARD is similar to that of ARD since the number of neighbors of the peers has almost no impact on the routing via bRDA. The same case also applies for the impact of network topologies on response traffic cost.

5.2 Response Traffic Cost and Response Time

The average response traffic costs of four different response delivery schemes are shown in Fig. 8. For the RRD scheme, only one additional path is selected for returning a duplicated response in the simulation. Compared with a Gnutella-like system, the RRD scheme incurs a 102 percent

Fig. 8. Response traffic cost versus peer lifetime.

increase in traffic cost. This is because the RRD scheme always sends a duplicate response message through another path, which is generally longer than the original response path, whether node failure happens or not on the original response path. ARD only increases the response traffic cost by about 9 percent and e-ARD creates even less extra traffic at 6 percent, compared to the original system during the response process. The e-ARD scheme creates limited overhead due to two issues. First, when there are no forwarding neighbors alive, a peer in e-ARD does not send a response back to the peers of the last hop immediately, but can forward it to a bRDA first. Second, sending a response to bRDA generally reduces the number of hops of routing the response message back.

On the other hand, extra response traffic cost of e-ARD is close to ARD. This happens due to the next two issues: First, the extra overhead created by the construction between the peer and bRDA will cut some of the benefit in limiting the extra traffic cost. Second, in the P2P system with a flooding query mechanism, the case that no other forwarding neighbors are alive happens rarely. Accordingly, in most of the cases where the primary forwarding neighbor fails, both ARD and e-ARD only execute the first option: send a response to some forwarding neighbor alive. Fig. 9 shows that, given a fixed average peer uptime of 10 minutes, the change of query frequency does not affect the extra traffic costs induced by the three mechanisms. Fig. 10 compares response traffic cost of a Gnutella-like system and ARD with different average number of neighbor connections. We can see that the response traffic cost decreases as the average number of neighbors increases. The reason is that when a peer has more neighbors, it has a higher probability of finding a candidate in his forwarding neighbor list to reroute the response instead of going back to his previous

Fig. 7. Response return rate under different network topologies (ARD).

Fig. 9. Response traffic cost versus query frequency.

Fig. 10. Response traffic cost under different network topologies.

hop in the case that the peer cannot reach the next hop to send back a response.

To investigate the impact of the extra traffic cost created in the entire query request/response process, we show the ratio of response traffic to request traffic of all the systems in Fig. 11. We can see that request traffic is from one to two orders of magnitude greater than response traffic. Therefore, even the traffic overhead of the RRD scheme can be ignored considering the traffic cost in the whole query process.

In a system that does not adopt flooding query process, the ratio of response traffic to request traffic may increase. For example, our simulation shows the ratio of response traffic and request traffic in random walk mechanism is around 1:10. Nevertheless, in this category of systems, e-ARD will be used in most of the time, which only incurs very limited extra response return traffic due to the adaptation of bRDA.

Fig. 12 shows the average response time of the four response schemes. As we expected, the average response time of both the RRD and ARD schemes are close to that of Gnutella-like systems. The response time in RRD is only 2 percent higher than that of Gnutella-like systems, while ARD is about 4 percent higher. The average response time of e-ARD is even shorter, only 1.2 percent more than that in Gnutella-like system.

5.3 Analysis of RRD, ARD, and e-ARD

In this section, we evaluate the influence of different key parameters in the three schemes, respectively.

Fig. 11. Ratio of response traffic over request traffic.

Fig. 12. Response time versus average peer lifetime.

5.3.1 Performance Improvements upon Different γ Values in RRD

The value of γ limits the number of redundant paths in RRD and affects the performance gain that can be achieved. We investigate the response return rates upon different γ values and present the results in Fig. 13. The results show that the response return rate does increase with the increase of the γ , but the improvement tends to be very little: With an average peer lifetime of 10 minutes, the response return rate is improved by 33 percent when γ is increased from 0.2 to 0.4 and further improved when γ is increased to 0.6. Nevertheless, marginal improvement is achieved when γ is increased to more than 0.6. This may be due to path overlap in the network. According to the previous investigations we present in Section 3.1, the peer lifetime in different P2P systems ranges from less than 10 minutes to 60 minutes. The influence of γ will decrease when the average peer lifetime increases.

5.3.2 Lifetime of Forwarding Neighbor Lists in ARD

Fig. 14 investigates the impact of the lifetime of forwarding neighbor lists on ARD's response return rate. We only investigate the impact on ARD, since the influence of bRDA in e-ARD will be trivial. The curves in Fig. 14 represent the response return rates of the ARD scheme with different lifetimes of forwarding neighbor lists, i.e., $1/4$, $1/2$, 1, and 2 times of the average query response time. We can see that when the lifetime of the forwarding neighbor list reaches twice the average response time, the return rate is more than 95 percent of that for infinite lifetime. The reason is

Fig. 13. Response return rate upon different γ values (RRD).

Fig. 14. Response return rate under different lifetimes of forwarding neighbor lists.

that if the lifetime is too short, a peer failing to deliver a response is less likely to find another candidate to reroute the response because the forwarding neighbor list may not be available any more, which will result in a high response failure rate.

5.3.3 Effectiveness of e-ARD in P2P Systems with a DFS-Based Searching Mechanism

To show how e-ARD performs in a system not adopting a flooding search algorithm, we deploy DFS in our simulator, and compare the response return rate in such a system with and without e-ARD. Fig. 15 shows the impact of e-ARD in response return rate, and Fig. 16 shows the response time of the system without flooding. We can see that e-ARD can effectively improve the P2P system performance: it improves the response return rate up to 60 percent and, at the same time, needs less time to return a response back to the requestor.

We also investigate the influence of the scope of α value via varying the value of c . The curves of e-ARD ($c = n$) in Fig. 15 and Fig. 16 indicate the performance of e-ARD when c is set to be n . We can observe that the response return rate increases as the c value increases. This is because when the c increases, the scope of α decreases, which makes fewer nodes as bRDAs. The decrease of the number of bRDAs eventually results in a shorter response return path upon the node failure. For the same reason, the response time drops as the c value increases. One more thing we can observe here is that the performances of e-ARD with different scope of α are close and, therefore, we should

Fig. 15. Response return rate with and without e-ARD (DFS).

Fig. 16. Response time with and without e-ARD (DFS).

choose a c value that makes the values of α large enough to satisfy the anonymity requirement for the requestor.

5.4 Comparison of RRD, ARD, and e-ARD

We summarize RRD, ARD, and e-ARD in Table 4. Among all three mechanisms, RRD requires the least modification of the existing response return mechanism, which makes it a quick fix for the current response return mechanism. On the other hand, RRD incurs more traffic overhead than ARD and e-ARD. Although nodes in ARD need to maintain a forwarding neighbor list locally, no new packet formats and extra bRDAs need to be introduced to construct ARD. The effectiveness and efficiency of ARD outperforms that of RRD and are close to that of e-ARD: up to 47 percent (ARD) versus 51 percent (e-ARD) upon response return rate improvement and 9 percent (ARD) versus 6 percent (e-ARD) upon extra traffic cost. Although the response time of ARD is the longest among the three mechanisms, the response time of ARD is very close to that of RRD and e-ARD. Considering both performance and implementation complexity, ARD is the best choice to remedy the response loss problem in existing response return mechanism.

Both RRD and ARD, however, work effectively based on the assumption that there are more than one forwarding neighbor for each peer in the system. In a system where such an assumption does not hold, e-ARD is the best candidate to avoid the response loss incurred by the dynamic nature of the system itself.

6 CONCLUSION

In an unstructured P2P system, query responses are sent back to the requestor along the incoming query path. However, the P2P system is a highly dynamic system in that average peer lifetime is from 10 to 60 minutes, and the logical connection between peers lasts from 1 to 24 minutes in average. This leads to a response loss problem, with up to 35 percent of the responses being lost.

To remedy the response loss problem, we present three techniques here: RRD, ARD, and e-ARD. All these techniques reduce response loss rate with limited extra cost regarding to the entire query process. RRD requires the least modification of the current response return mechanism to implement. ARD functions more effective and with more efficiency than RRD, while its implementation complexity is less than that of e-ARD. e-ARD extends

TABLE 4
Comparison of RRD, ARD, and e-ARD

	RRD	ARD	e-ARD
Effectiveness (Response Return Rate)	effective (up to 35%)	more effective (up to 47%)	most effective (up to 51%)
Efficiency (Extra Traffic Cost)	double the response cost in Gnutella-like system and maybe even higher (102%)	less extra traffic overhead (less than 9%)	least extra traffic overhead (6%)
Quality of Service (Response Time)	shorter (2% more)	longest (4% more)	shortest (1% more)
Implementation Restrictions	node in the path has more than one forwarding neighbor; performance depends on how the redundant paths overlap	node in the query path has more than one forwarding neighbor; performance depends on how many forwarding neighbors a node has in the query path	none
Implementation Complexity	no extra complexity	each node maintains a forwarding list	each node maintains a forwarding list; new message formats are introduced; extra bRDAs are introduced
Application	quick fix for current system	system adopting flooding search system; dense network	system not adopting flooding search system; sparse network

ARD and can be used in unstructured P2P systems with a limited or nonflooding search mechanism.

We will further investigate the effect of the ARD algorithm in the case when both node failure and logical link breakage occur, and attempt to combine it with other optimization algorithms, such as P2P network topology adaptation. We also plan to implement a prototype of our approach based on the current version of the Gnutella open source code in PlanetLab [21].

ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for their critical and constructive comments on this paper. They would also like to thank Ruth Mendel for reading the paper and her suggestions. This work is supported in part by the US National Science Foundation under grants CCF-0325760, CCF 0514078, and CNS 0549006, and by Hong Kong RGC DAG 04/05.EG01. Some preliminary results of this work were presented in the *Proceedings of MASCOTS 2004*.

REFERENCES

- [1] The Gnutella protocol specification 0.6, <http://rfc-gnutella.sourceforge.net>, 2002.
- [2] KaZaA, <http://www.kazaa.com>, 2003.
- [3] Napster, <http://www.napster.com>, 2003.
- [4] K. Aberer, P. Cudre-Mauroux, A. Datta, and Z. Despotovic, "P-Grid: A Self-Organizing Structured P2P System," *Proc. Int'l Conf. Cooperative Information Systems*, 1995.
- [5] V. Almeida, A. Bestavros, M. Crovella, and A.d. Olivera, "Characterizing Reference Locality in the WWW," *Proc. IEEE Conf. Parallel and Distributed Information Systems (PDIS)*, 1996.
- [6] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web Caching and Zipf-Like Distributions: Evidence and Implications," *Proc. INFOCOM*, 1999.
- [7] F.E. Bustamante and Y. Qiao, "Friendships that Last: Peer Lifespan and Its Role in P2P Protocols," *Proc. Eighth Int'l Workshop Web Content Caching and Distribution (WCW '03)*, 2003.
- [8] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making Gnutella-Like P2P Systems Scalable," *Proc. ACM SIGCOMM*, 2003.
- [9] N. Daswani and H. Garcia-Molina, "Query-Flood DoS Attacks in Gnutella," *Proc. ACM Computer and Comm. Security Conf.*, 2002.
- [10] D. Dumitriu, E. Knightly, A. Kuzmanovic, I. Stoica, and W. Zwaenepoel, "Denial-of-Service Resilience in Peer-to-Peer File Sharing Systems," *Proc. ACM SIGMETRICS '05*, 2005.
- [11] C. Gkantsidis, M. Mihail, and A. Saberi, "Random Walks in Peer-to-Peer Networks," *Proc. INFOCOM*, 2004.
- [12] 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 (SOSP '03)*, 2003.
- [13] D.B. Johnson and D.A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networking," *Proc. Mobile Computing Conf.*, 1996.
- [14] K. Sripanidkulchai, B. Maggs, and H. Zhang, "Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems," *Proc. INFOCOM*, 2003.
- [15] P. Keyani, B. Larson, and M. Senthil, "Peer Pressure: Distributed Recovery from Attacks in Peer-to-Peer Systems," *Proc. IFIP Workshop Peer-to-Peer Computing*, 2002.
- [16] Y. Liu, X. Liu, L. Xiao, L. Ni, and X. Zhang, "Location-Aware Topology Matching in Unstructured P2P Systems," *Proc. INFOCOM*, 2004.
- [17] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and Replication in Unstructured Peer-to-Peer Networks," *Proc. ACM Int'l Conf. Supercomputing*, 2002.
- [18] E.P. Markatos, "Tracing a Large-Scale Peer to Peer System: An Hour in the Life of Gnutella," *Proc. IEEE Int'l Symp. Cluster Computing and the (CCGrid)*, 2002.
- [19] V. Park and M. Corson, "Temporally-Ordered Routing Algorithm (TORA) Version 1: Functional Specification," IETF Internet draft (draft-ietf-tora-spec-04.txt), 2001.
- [20] C.E. Perkins and E.M. Royer, "Ad-Hoc On-Demand Distance Vector Routing," *Proc. MILCOM '97*, 1997.
- [21] L. Peterson, D. Culler, T. Anderson, and T. Roscoe, "A Blueprint for Introducing Disruptive Technology into the Internet," *Proc. HotNets Conf.*, 2002.
- [22] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network," *Proc. ACM SIGCOMM*, 2001.
- [23] C. Rohrs, "Query Routing for the Gnutella Network," <http://rfc-gnutella.sourceforge.net>, 2001.

- [24] S. Sen and J. Wang, "Analyzing Peer-to-Peer Traffic across Large Networks," *Proc. ACM SIGCOMM Internet Measurement Workshop*, 2002.
- [25] S. Saroiu, K. Gummadi, R. Dunn, S. Gribble, and H. Levy, "An Analysis of Internet Content Delivery Systems," *Proc. Symp. Operating Systems Design and Implementation (OSDI)*, 2002.
- [26] S. Saroiu, P. Gummadi, S. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," *Proc. Multimedia Computing and Networking Conf. (MMCN)*, 2002.
- [27] K. Sripanidkulchai, "The Popularity of Gnutella Queries and Its Implications on Scalability," <http://www-2.cs.cmu.edu/~kunwadee/research/p2p/gnutella.html>, 2001.
- [28] I. Stoica, S. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," *Proc. ACM SIGCOMM*, 2001.
- [29] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger, "Network Topology Generators: Degree-Based vs. Structural," *Proc. ACM SIGCOMM*, 2002.
- [30] S. Wang, D. Xuan, and W. Zhao, "On Resilience of Structured Peer-to-Peer Systems," *Proc. GLOBECOM*, 2003.
- [31] B.Y. Zhao, L. Huang, J. Stribling, S.C. Rhea, A.D. Joseph, and J.D. Kubiawicz, "Tapestry: An Infrastructure for Fault-Resilient Wide-Area Location and Routing," *IEEE J. Selected Areas in Comm.*, 2001.
- [32] S.Q. Zhuang, D. Geels, I. Stoica, and R.H. Katz, "On Failure Detection Algorithms in Overlay Networks," *Proc. INFOCOM*, 2005.

Xiaomei Liu received the BS degree in electronics and information system technology from East China Normal University in 1996, and the MS degree in computer engineering from the University of Toledo in 2000. She is currently a PhD student at the Michigan State University. Her research interests include distributed operating systems and computer network. She is a student member of the IEEE and the IEEE Computer Society.

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 the Department of Computer Science at 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.

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, overlay systems and applications, sensor networks, 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.

► For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.