ACOM: Any-Source Capacity-Constrained Overlay Multicast in Non-DHT P2P Networks

Shiping Chen, Baile Shi, Shigang Chen, and Ye Xia

Abstract—Application-level multicast is a promising alternative to IP multicast due to its independence from the IP routing infrastructure and its flexibility in constructing the delivery trees. The existing overlay multicast systems either support a single data source or have high maintenance overhead when multiple sources are allowed. They are inefficient for applications that require any-source multicast with varied host capacities and dynamic membership. This paper proposes ACOM, an any-source capacity-constrained overlay multicast system, consisting of three distributed multicast algorithms on top of a non-DHT overlay network with simple structures (random overlay with a non-DHT ring) that are easy to manage as nodes join and depart. The nodes have different capacities, and they can support different numbers of direct children during a multicast session. No explicit multicast trees are maintained on top of the overlay. The distributed execution of the algorithms naturally defines an implicit, roughly balanced, capacity-constrained multicast tree for each source node. We prove that the system can deliver a multicast message from any source to all nodes in expected $O(\log_c n)$ hops, which is asymptotically optimal, where c is the average node capacity and n is the number of members in a multicast group.

Index Terms—Any-source overlay multicast, peer-to-peer networks, distributed multicast algorithms.

1 Introduction

The deployment of router-based IP multicast has been slow due to the requirement for global deployment of multicast-capable routers, the lack of ISP support, and the state scalability problem caused by a large number of multicast groups. Application-level multicast becomes a promising alternative that can be quickly deployed without the dependency on the router infrastructure [1], [2], [3], [4]. Recent papers [5], [6], [7], [8], [9], [10] studied overlay multicast from different aspects. However, they are insufficient in supporting applications that require any-source multicast with varied host capacities and dynamic membership.

Many works focus on designing an optimized overlay multicast tree for a single data source [11], [12], [13], [14]. They are suitable for video or software distribution but not for distributed games, teleconferencing, virtual classrooms (discussion sessions), or multimedia chat groups, where many or all nodes can potentially be the data sources at any time. A multicast tree that is optimal for one source may be bad for other sources. On the other hand, one tree per member will be too costly.

To add to the problems, member hosts may vary widely in their capacities in terms of upload bandwidth, memory, and CPU power. Some are able to support a large number of direct children, but others support few. Furthermore, members may join and leave at any time, which makes the maintenance of the overlay network and the multicast trees a critical issue. Finally, the system should be fully distributed and scalable to a large Internet group.

The goal of this paper is to study low-maintenance overlay systems that support distributed applications requiring any-source capacity-constrained multicast with dynamic membership. None of the existing systems to be surveyed in the next section meets all these requirements.

In order to handle any-source multicast in dynamic groups, novel proposals were made to implement multicast on top of structured P2P networks [15], [16], [17]. However, the overhead of maintaining DHT-based multicast overlays is a major design concern [18]. Moreover, these systems assume each node has the same number of children. Host heterogeneity is not addressed. Even though overlay multicast can be implemented on top of overlay unicast, they have very different requirements. In overlay unicast, low-capacity nodes affect only traffic passing through them and, therefore, they create bottlenecks of limited impact. In overlay multicast, all traffic will pass all nodes in the group and the multicast throughput is decided by the node with the smallest throughput, particularly in the case of reliable delivery. The strategy of assigning an equal number of children to each intermediate node is far from optimal. If the number of children is set too big, the low-capacity nodes will be overloaded, which slows down the entire session. If the number of children is set too small, the high-capacity nodes will be underutilized. CAM-Chord [19], a multicast extension of Chord, allows nodes to have a different number of neighbors. It achieves roughly balanced multicast trees, which helps reduce both delay and delay jitter, but its maintenance overhead is large for highly dynamic multicast groups.

This paper proposes ACOM, an any-source capacity-constrained overlay multicast service, consisting of three distributed multicast algorithms on top of a non-DHT overlay network with simple structures (random overlay with a non-DHT ring) that are easy to manage as nodes join and depart. At runtime, the distributed execution of a proposed algorithm naturally defines an implicit, roughly balanced, capacity-constrained multicast tree for delivering a message

- *S. Chen is with the Network Center, University of Shanghai for Science and Technology, Shanghai, 200093, China. E-mail: chensp@usst.edu.cn.*
- *B. Shi is with the Department of Computer and Information Technology, FuDan University, Shanghai, 200433, China. E-mail: bshi@fudan.edu.cn.*
- *S. Chen and Y. Xia are with the Department of Computer and Information Science and Engineering, University of Florida, Gainesville, FL 32611. E-mail: {sgchen, yx1}@cise.ufl.edu.*

Manuscript received 2 Nov. 2005; revised 7 June 2006; accepted 10 July 2006; published online 9 Jan. 2007.
Recommended for acceptance by Y. Pan.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-0459-1105.
Digital Object Identifier no. 10.1109/TPDS.2007.1037.

from an arbitrary source node. No explicit multicast trees are built and maintained. In ACOM, the nodes may have different capacities and the numbers of direct children that they can support in a multicast session may vary. Each node keeps a number of random neighbors equal to its capacity. We prove that the system can deliver a multicast message from any source to all nodes in an expected $O(\log_c n)$ hops, which is asymptotically optimal, where c is the average node capacity and n is the number of members in a multicast group. In comparison, CAM-Chord requires $O(\log_c n)$ times more neighbors per node. Our observation is that the DHT-based overlay structures designed for file lookup are not optimal for multicast; other, simpler structures should be explored. We perform extensive simulations to evaluate the performance of the proposed system, and we provide detailed discussions on a variety of implementation issues, including how to maintain the overlay, how to measure the system parameters, and how to handle low-capacity nodes, proximity, as well as dynamic capacities.

The rest of the paper is organized as follows: Section 2 discusses the related work. Section 3 defines the problem and the network model. Section 4 motivates our approach, gives a basic version of the distributed multicast algorithm, and provides a thorough analysis. Section 5 discusses the implementation issues. Section 6 describes two improved multicast algorithms. Section 7 presents the simulation results. Section 8 draws the conclusion.

2 Related Work

Shi et al. proved that constructing a minimum-diameter degree-limited spanning tree is NP-hard [20]. Note that the terms "degree" and "capacity" are interchangeable in the context of this paper. Centralized heuristic algorithms were proposed to balance multicast traffic among multicast service nodes (MSNs) and to maintain low end-to-end latency [20], [21]. The algorithms do not address the dynamic membership problem, i.e., MSN join/departure.

A number of overlay multicast systems have been proposed to optimize a single-source multicast tree. Bullet [11] is designed to improve the throughput of data dissemination from one source to a group of receivers. An overlay tree rooted at the source is first established. Disjoint data objects are delivered from the source via the tree to different receivers. The receivers then communicate among themselves to retrieve the missing objects. The direct communication between receivers adds more bandwidth to the tree dissemination. Overlay Multicast Network Infrastructure (OMNI) [12] dynamically adapts its degree-constrained multicast tree to minimize the latencies to the entire client set. Riabov and Zhen Liu proposed a centralized constant-factor approximation algorithm for the problem of constructing a single-source degree-constrained minimum-delay multicast tree [13]. Yamaguchi et al. described a distributed algorithm that maintains a degree-constrained delay-sensitive multicast tree for a dynamic group [14]. The above algorithms are designed for a single source. They are not suitable for multisource scenarios (e.g., distributed games and multimedia chat groups), which are the target applications of this paper. Building one tree for each possible source is too costly. Using a single tree for all sources is also problematic. First, a minimum-delay tree for one source may not be a minimum-delay tree for other sources. Second, the single-tree approach concentrates the traffic on the links of the tree. Third, a single tree may be partitioned beyond repair for a dynamic group.

Bayeux [15] and Borg [16] were proposed to support application-level multicast based on Tapestry [22] and Pastry [23], respectively, and CAN-based Multicast [17] was implemented on top of CAN [24]. El-Ansary et al. studied efficient broadcast in a Chord network, and their approach can be adapted for the purpose of multicast [25]. Castro et al. compares the performance of tree-based and flooding-based multicast in CAN-style versus Pastry-style overlay networks [18]. One of the conclusions is that the overhead of maintaining DHT-based multicast overlays is a major design concern. With each node having the same number of children, the above systems do not consider the heterogeneity in host capacities.

3 Model

Consider a multicast group G of n nodes. Each node $x \in G$ has a capacity c_x, specifying the number of direct child nodes to which x is willing to forward the received multicast messages. The value of c_x should be made roughly proportional to the upload bandwidth of node x. The upload bandwidth is likely to be the bottleneck because the common cable-modem or DSL links are highly asymmetric and biased against upload. Moreover, a node x may have to forward c_x copies for each received message.[1] In a heterogeneous environment, the capacities of different nodes may vary in a wide range. The capacity of the same node may also change over time (Section 5.6). We want to construct a multicast service, which meets the *capacity constraints* of all nodes, allows *frequent membership changes*, and delivers multicast messages from *any source* to the group members via a *dynamic, balanced* multicast tree.

To achieve this goal, an overlay network is established for each multicast group, which transforms the multicast problem to a broadcast one within the scope of the overlay. A single delivery tree for all data sources is not a good idea because it concentrates traffic on a small number of overlay links. On the other hand, one tree per source is too costly to maintain for highly dynamic groups. An ideal approach is to use "implicit" per-source multicast trees that are not explicitly built and thus incur no maintenance overhead. These implicit trees change naturally at zero cost as the overlay topology changes. Before presenting how this can be achieved, we shall describe the non-DHT (Distributed Hash Table) overlay network that we use.

The overlay network consists of two components. One is an "unrestricted" ring that connects all nodes. The other is a random graph among the nodes. The unrestricted ring is fundamentally different from the DHT-based ring found in many structured P2P networks. In the DHT-based ring, each node has a specific location, which makes the maintenance of the ring difficult when a new node joins the network. In the unrestricted ring, a node can be anywhere in the ring. The ring maintenance is trivial. Each node keeps the next node on the ring as one of its neighbors, called the *successor*. As long as a new node x knows another node z that is already in the ring, it informs z to set x as the successor, and then it sets its own successor to be z's previous successor. In addition to the successor, a node x has c_x or more other neighbors randomly selected from the set of nodes. Details about topology maintenance and proximity will be discussed later in Section 5. We assume that x is willing to support both the successor and the

1. These copies follow different Internet paths to their respective destinations, which makes the upload link of x the only shared bottleneck.

c_x random neighbors as its children in a multicast session. A TCP connection is established between two neighbors for data and control communication. When there is no data communication, TCP transmits keep-alive segments. The departure of a neighbor is detected when the TCP connection times out due to the lack of keep-alive segments.

To prevent the ring from being broken after an abrupt node departure, each node should also know a number of nodes after the successor down the ring. It also knows the previous node on the ring, called the *predecessor*. However, these nodes will not be used as neighbors for data communication.

In the existing literature, the structured P2P networks all use DHT, and the unstructured P2P networks do not use DHT. In this sense, our overlay network belongs to the unstructured category. On the other hand, it does have (non-DHT) structures, a ring, and a random network. We may reclassify P2P networks in three categories, DHT-based structured, non-DHT structured, and unstructured, with the second category including all P2P networks that do not use DHT but have topological structures, and the third category including all P2P networks that assume arbitrary topologies.

4 Any-Source Capacity-Constrained Overlay Multicast

4.1 Motivation

Random Walk, *Limited Flooding*, and *Probabilistic Flooding*. An overlay network for multicast is fundamentally different from an overlay network for file sharing such as Gnutella or Kazaa. In file-sharing applications, the existing approaches of random walk, limited flooding, and probabilistic flooding work well when the searched object has many copies stored in dispersed locations of the network. Searching only a small portion of the nodes will turn up the object with a good probability. Missing an existing object with a small probability is not considered to be a serious problem. For multicast, the requirement is different. An overlay network is constructed among the nodes of a multicast group. Multicast is implemented as broadcast. Every node of the group is supposed to receive a copy of every message, which neither random walk nor limited flooding can guarantee. By forwarding a message (received for the first time) to neighbors with a certain probability, the approach of probabilistic flooding also cannot guarantee reaching every node, even when the overlay topology itself is connected. Efficient broadcasting in a non-DHT overlay has been a challenging problem.

Hop Complexity and Communication Complexity. We introduce two performance metrics for overlay multicasting with nodes having different capacities. Let c be the average node capacity. Other performance considerations will be discussed later. The hop complexity is defined as the expected number of overlay hops it takes for a multicast message to reach any node. The communication complexity is the expected number of copies of a multicast message that are transmitted in order to ensure that the message is delivered to all nodes. Whenever a node forwards a message to a neighbor, that counts as one copy. A smaller hop complexity means a more balanced multicast tree and a smaller average delivery latency. A smaller communication complexity means less network bandwidth consumed. In the following, we use two multicast schemes to demonstrate a trade-off between these two complexities. One complexity can be optimized at the expense of the other.

TABLE 1
Comparison of Complexities

multicast scheme	hop complexity	comm. complexity
fully flooding	$O(\log_c n)$	cn
ring traversal	$n/2$	n
ACOM	$O(\log_c n)$	$n + O(\frac{n}{\log_c n})$

Full Flooding. In this scheme, each node forwards a message to its random neighbors when the message is received for the first time. It can be shown that 1) the hop complexity is $O(\log_c n)$, which is optimal because flooding follows the shortest paths, and 2) the communication complexity is cn, with each node x forwarding c_x copies.

Ring Traversal. On the other end of the spectrum, we can achieve the best communication complexity of n by forwarding a message along the ring that serially connects all nodes. However, the hop complexity is $n/2$, which is the worst among all schemes.

ACOM. We demonstrate that there are other alternatives between these two extremes. In particular, we design a multicast scheme, called ACOM, that realizes a favorable trade-off between the hop complexity and the communication complexity. It achieves asymptotically optimal hop complexity of $O(\log_c n)$ at suboptimal communication complexity of $n + O(\frac{n}{\log_c n})$.

A quick reference to the complexity comparison is given Table 1.

4.2 Idea

Given the overlay topology defined in Section 3, we want to design a multicast routine such that

1. it can deliver a message from any source to every node,
2. its hop complexity is $O(\log_c n)$, which is asymptotically optimal,
3. its communication complexity is close to n, which is optimal, and
4. it does so without building and maintaining explicit multicast trees on top of the overlay.

The idea is to perform two-phase multicast. Phase one is partial random distribution, and phase two is segmented ring traversal. Combined, they define implicit per-source trees that are embedded in the structure of the existing overlay without additional cost. These implicit trees are roughly balanced and capacity-constrained. They are different for each source and, consequently, spread the workload to as many overlay links as possible. Because we do not establish any explicit multicast tree, we avoid the problem (and the overhead) of tree maintenance on top of the underlying overlay network.

Phase one. It forwards the message via random neighbors to nodes within K hops from the source, where K is a system parameter that will be determined later. This phase delivers the message to a subset of nodes that are randomly located across the network. They are called *phase-one nodes*. An illustration is given by the first two plots of Fig. 1. Suppose the capacity of the source node s is 3 and the capacities of nodes i, j, and l are 2, 2, and 3, respectively. In

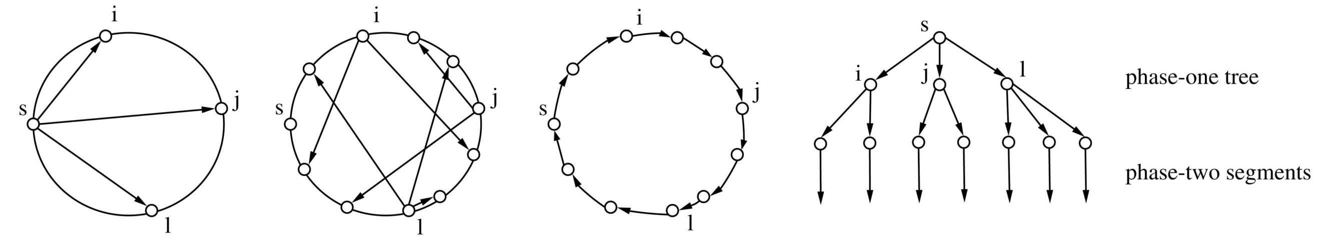


Fig. 1. Two-phase multicast.

phase one, the message is delivered to 10 nodes in a tree structure, where the number of children of a node is constrained by its capacity.

Phase two. Each phase-one node will deliver the message to its "neighborhood." If we had used the traditional concept of a neighborhood consisting of nodes within a certain number of hops away, phase two would become distributed limited flooding performed by all phase-one nodes. In order to cover the entire network, the phase-one nodes would have to flood the messages deep enough in their neighborhoods, causing overlaps where some nodes would receive multiple copies of the message. To avoid this problem, we use the unrestricted ring instead. The phase-one nodes partition the ring into segments. As illustrated in the third plot of Fig. 1, each phase-one node is responsible for one adjacent segment. It forwards the message to its successor, which further forwards the message to the successor's successor, ..., until the message reaches a node that has already received the message.

The two phases are completely decentralized with no global coordination. Together, they deliver a message in a tree structure, as illustrated in the fourth plot of Fig. 1. The tree follows the fabric of the underlying overlay network and does not need maintenance. Phase one restricts flooding in K hops. The problem with full flooding is at its final stage when most nodes forward the message to neighbors that have already received the message. The segmented ring traversal in phase two avoids such collisions while still ensuring that every node receives a copy of the message.

If the value of K is *small* enough, the partial random distribution in phase one is likely to deliver the message to distinctive random nodes, and the probability of a node receiving more than one copy will be small, which helps to reduce the communication complexity. If the value of K is *large* enough, there will be enough phase-one nodes to partition the ring into small segments, which reduces the hop complexity in phase two. These two conflicting requirements on K lead to an interesting question. Can we choose a right value for K such that the hop complexity is asymptotically optimal, $O(\log_c n)$, and the communication complexity is close to n? Nodes are allowed to have different numbers of neighbors, which makes the question more interesting and more realistic, but also much harder. Below, we first give a basic version of the algorithm. Its simple structure serves the purposes of bringing out the idea and alleviating analysis from optimization details. We then analyze the properties of the algorithm, address the implementation issues, and, finally, discuss two improved versions of the algorithm.

4.3 Basic Algorithm (ACOM-1)

A multicast message is denoted as $M(k, id)$, where M is the message, k is the TTL field whose initial value is K, and id is a globally unique identifier, consisting of the source ID (e.g., IP address) and a sequence number. To deliver a multicast message, the source node s begins by forwarding $M(K, id)$ to c_s random neighbors. When a node x receives $M(k, id)$, if it has not received M with id before and $k > 0$, it decreases k by one and forwards the message to c_x random neighbors. If $k = 0$, it only forwards the message to its successor. The pseudocode of the algorithm is given in Fig. 2. The key question is how to determine the value of K. For this purpose, we introduce some notations below that will also be used in the analysis.

In phase one, when a node receives a multicast message for the first time, it forwards the message to its random neighbors. Collectively, the message is delivered in a tree structure, called the *phase-one tree*, formed by random neighbors. The depth of the tree is K. The source node s is at level 0. Let q_i be the number of nodes at level i. $q_0 = 1$, $q_1 = c_s$, and q_i, $i \in [2..K]$, is a random number because the random neighbors of nodes at level $(i-1)$ may overlap at level i. The number of nodes at levels zero through i in the phase-one tree is $Q_i = \sum_{j=0}^{i} q_j$. The total number of nodes in the phase-one tree is Q_K.

All of the leaves of the phase-one tree are at level K. The number of internal nodes is Q_{K-1}. The number of children of an internal node is bounded by its capacity. Let $c_{i,1}$, $c_{i,2}$, ..., and c_{i,q_i} be the capacities of the nodes at level i. The summation of the capacities of all internal nodes, plus one for the source, i.e., $1 + \sum_{i=0}^{K-1} \sum_{j=0}^{q_i} c_{i,j}$, gives an upper bound for the tree size Q_K. To avoid excessive duplicate receipts of the same message, our design philosophy is for phase one to reach only a small fraction of nodes in the

```
Upon receipt of message $M(k, id)$ by node $x$
        /* Phase One */
(1)  **if** $k > 0$ and $x$ hasn't forwarded $M$ with $id$ to random neighbors **then**
(2)      select $c_x$ random neighbors
(3)      **for** each selected neighbor $y$ **do**
(4)          forward $M(k-1, id)$ to $y$
(5)      **if** $x$ hasn't forwarded $M$ with $id$ to $successor(x)$ **then**
(6)          forward $M(0, id)$ to $successor(x)$
        /* Phase Two */
(7)  **else if** $k = 0$ and $x$ hasn't forwarded $M$ with $id$ to $successor(x)$ **then**
(8)      forward $M(0, id)$ to $successor(x)$
(9)  **else**
(10)     discard the message
```

Fig. 2. Basic algorithm (ACOM-1).

multicast group. From those nodes, the message is delivered in a circular fashion to reach all other nodes in phase two. Therefore, we want to choose K conservatively such that the tree size Q_K is far smaller than n. Define a system parameter λ as follows:

$$\sum_{i=0}^{K} c^i = \lambda n. \tag{1}$$

λ characterizes the "intended" percentage of nodes in N that are covered by the phase-one tree. We emphasize that λ *is a system parameter that we pick*. After we pick the value of λ, the value of K can be calculated from (1). Hence, the problem of determining an appropriate value for K becomes the problem of determining an appropriate value for λ. In the following, we show that, if we choose $\lambda = \Theta(\frac{1}{\log_c n})$, the hop complexity will be $O(\log_c n)$ and the communication complexity will be $n + O(\frac{n}{\log_c n})$. How to measure n and c will be addressed in Section 5.

4.4 Analysis

The main analytical results are given below. The node capacities are not fixed but, instead, are treated as random variables, which makes the analysis considerably tougher. The proofs can be found in the Appendix.

Theorem 1 (Hop Complexity). *The expected number of overlay hops for a multicast message to reach any node is bounded by*

$$\log_c(\lambda n) + \frac{1}{2\big(1 - \frac{1}{c}\big)\big(1 - \frac{\lambda c}{c-1}\big)\lambda}.$$

Corollary 1 (Hop Complexity). *If $\lambda = \Theta(\frac{1}{\log_c n})$ and $c \geq 2$, then the expected number of overlay hops for a multicast message to reach any node is $O(\log_c n)$.*

Theorem 2 (Communication Complexity). *The expected number of copies of a multicast message that are transmitted is bounded by $(1 + \lambda)n$.*

Corollary 2 (Communication Complexity). *If $\lambda = \Theta(\frac{1}{\log_c n})$, then the expected number of copies of a multicast message that are transmitted is bounded by $n + O(\frac{n}{\log_c n})$.*

5 Implementation Issues

5.1 Maintaining the Ring

Unlike Chord [26], in our overlay network, there is no restriction on where a node should be placed on the ring. When a new node x joins the multicast group, it must know a node z that is already in the group.[2] x joins as z's successor, and z's previous successor becomes x's successor. In comparison, Chord or most other DHT networks must perform a lookup in order to determine the specific location where x can join in the network.

As we discussed in Section 3, each node knows a number of nodes after the successor down the ring to prevent the ring from being broken after the successor's abrupt departure. This method was used by Chord to keep its successor ring from being broken, and it was inherited by other systems [27], [19] that used Chord's network maintenance protocols.

In the rare case that all of those nodes depart abruptly together, a broadcast is performed to identify the nodes that do not have predecessors or successors so that they can be connected to form a ring.

5.2 Establishing Random Neighbors

After a new node x joins the ring, it can ask a multicast source s to help find its c_x random neighbors. When s multicasts a message, it attaches c_x tokens, containing x's IP address. Each token, piggybacked in the message, independently chooses a random path in the phase-one tree, and then traverses a ring segment in phase two, during which it will pick a random node from the segment. More specifically, the token tentatively picks the first node of the segment, sets $L = 1$, and begins the segment traversal. At each hop, L is increased by one and the token has a probability of $1/L$ of replacing the previously picked node with the current node. At the end of the ring segment, the node that is picked last is reported to x as a random neighbor. Because a multicast message is received by every node in the group, every node will have a chance to be x's neighbor. As a low-probability special case in phase one, when a message carrying a token arrives at a node that has already received the message, the node discards the message and reports itself to x.

Without the knowledge of a multicast source, when x joins as z's successor, it can ask z's help to find random neighbors. z sends out c_x tokens. Each token independently performs a random walk of $O(\log_c n)$ hops and then reports the reached node to x. In total, the tokens travel $O(c_x \log_c n)$ hops to find c_x random neighbors.

When a node loses a random neighbor (that departs), it adds a new neighbor by random walk.

The neighbor links are directed, which means, even if y is a random neighbor of x, x may not be a random neighbor of y. In a random network, each node is expected to serve as a neighbor for roughly the same number of other nodes. Based on this observation, we design a simple optimization method to improve the randomness in neighbor selection. Let I_x be the set of nodes for which x is a random neighbor. x has the knowledge of I_x. Periodically x exchanges $|I_x|$ with its neighbors. If $|I_x|$ is greater than the smallest $|I_y|$ value that x receives from neighbors y, x will inform a node in I_x to use y (with the smallest $|I_y|$ value) instead of x as a random neighbor. This will gradually equalize $|I_x|$ and $|I_y|$. In addition, x selects an arbitrary neighbor z and swaps a node in I_x with a node in I_z. All of the above communication can be piggybacked in the keep-alive messages between neighbors.

5.3 Estimating n and c

Because a multicast message will reach every node, the measurement of n can be carried out during the process of multicast. Periodically, a multicast source piggybacks a flag in a multicast message. As the message with the flag is distributed, each node temporarily remembers who the parent is. The node count information is propagated backward from the end nodes of the Phase-two ring segments back to the source node, where the value of n is

2. When a node joins a P2P network, it must know an existing node in the overlay. The node cannot join if it has zero clue about where the overlay network is. This requirement is true for Gnutella, Kazaa, or any other P2P network. In the IP multicast, a new member may query a directory server for the root of the multicast tree. In ACOM, a new member may query the server for an existing node in the overlay. The owner (creator) of a multicast group is a permanent member that is responsible for updating the server with a number of existing nodes.

```
Upon receipt of message $M(k, p, id)$ by node $x$
      /* Phase One */
(1)   if $k > 0$ and $x$ hasn't forwarded $M$ with $id$ to random neighbors then
(2)       select $c_x$ random neighbors
(3)       for each selected neighbor $y$ do
(4)           forward $M(k-1, p, id)$ to $y$
(5)       if $x$ hasn't forwarded $M$ with $id$ to $successor(x)$ then
(6)           forward $M(0, 0, id)$ to $successor(x)$
(7)   else if $k = 0$ and $p > 0$ and $x$ hasn't forwarded $M$ with $id$ to random neighbors then
(8)       select $c_x$ random neighbors
(9)       for each selected neighbor $y$ do
(10)          forward $M(0, 0, id)$ to $y$ with probability $p$
(11)      if $x$ hasn't forwarded $M$ with $id$ to $successor(x)$ then
(12)          forward $M(0, 0, id)$ to $successor(x)$
      /* Phase Two */
(13)  else if $k = 0$ and $x$ hasn't forwarded $M$ with $id$ to $successor(x)$ then
(14)      forward $M(0, 0, id)$ to $successor(x)$
(15)  else
(16)      discard the message
```

Fig. 3. Improved algorithm (ACOM-2).

determined. The source node then distributes n to all other nodes, piggybacked in the next multicast message.

To guard against some nodes leaving the network during the process, when a node x finds the connection to a child node is broken or it hasn't received the node count from the child after a timeout period, it will artificially assign that child the average node count reported by other children minus one (due to the departure of that child). x will then report the total node count from all children plus one to its parent. To further improve the accuracy, the source may perform a number of measurements of n in a period of time. It may also receive the measurements from other sources. It will take the average measured value of n during that period as the new estimation of n.

The value of c is measured along with n. Together with the node count, the information of the average node capacity will also be propagated backward from the nodes to the source.

5.4 Group Members with Very Small Upload Bandwidth

A node x with very small upload bandwidth should only be a leaf in the implicit multicast trees unless it is the data source. In order to make sure that we do not select x as an intermediate node in any phase-one tree, x should not be a regular member of the overlay network. Instead, it joins as an external member. x asks a node y known to be in the overlay to perform a random walk of $O(\log_c n)$ hops to identify a random node z. x then attempts to join z as an external member. If z cannot support x, z forwards x to $successor(z)$. If z admits x as an external member, z will forward the received multicast messages to x and x will multicast its messages via z. If z leaves the group, x must rejoin via another node in the overlay.

5.5 Proximity

The purpose of phase one is to spread a message across the entire network such that each region is likely to receive a copy. Therefore, random neighbors are desirable. In phase two, the message travels along ring segments. There is no restriction on where a node should be located on the ring. Hence, proximity measures can come into play. We want to reduce the latency of the overlay links on the ring. One approach uses virtual coordinates [28]. When a new node x joins the overlay, it pings a set of landmark machines. From the delays to the landmarks, a set of virtual coordinates are calculated. When c_x random walks are performed to identify x's random neighbors (Section 5.2), x's virtual coordinates are carried. As a random-walk control message travels around, it finds the node on its path that is closest to x based on the distance calculated from the virtual coordinates. This node is reported to x at the end of the random walk. Among all reports, x picks the best one to join as its successor.

5.6 Dynamic Capacities

The capacity of a node does not have to be fixed. It may change as the node's upload bandwidth changes. According to the algorithms in Figs. 2, 3, and 4, a node x forwards a multicast

```
Upon receipt of message $M(t, id)$ by node $x$
    /* Phase One */
(1)  if $t > 0$ and $x$ hasn't forwarded $M$ with $id$ to random neighbors then
(2)      $t := t - 1$
(3)      $m := t - \lfloor \frac{t}{c_x} \rfloor c_x$
(4)      for each of $m$ selected random neighbors $y$ do
(5)          forward $M(\lceil \frac{t}{c_x} \rceil, id)$ to $y$
(6)      if $\lfloor \frac{t}{c_x} \rfloor > 0$ then
(7)          for each of $(c_x - m)$ selected random neighbors $y$ do
(8)              forward $M(\lfloor \frac{t}{c_x} \rfloor, id)$ to $y$
(9)      if $x$ hasn't forwarded $M$ with $id$ to $successor(x)$ then
(10)         forward $M(0, id)$ to $successor(x)$
    /* Phase Two */
(11) else if $t = 0$ and $x$ hasn't forwarded $M$ with $id$ to $successor(x)$ then
(12)     forward $M(0, id)$ to $successor(x)$
(13) else
(14)     discard the message
```

Fig. 4. Ticket algorithm (ACOM-3).

message to c_x selected random neighbors based on the current value of c_x, which may be different for future messages. If c_x becomes greater than the current number m of random neighbors, then x will only forward the message to the m neighbors. If c_x is consistently greater than m for a period of time, x will perform random walks to add more neighbors.

5.7 Terminating Duplicate Copies

Consider a node x forwarding a message to a neighbor y. Once y receives the message header, it knows whether the message has already been received based on the message identifier id. If this is a duplicate copy and the message is very long, y immediately sends a control message to x and asks x to stop forwarding, which significantly reduces the overhead.

6 Improved Algorithms (ACOM-2 and ACOM-3)

By the design of the algorithm ACOM-1, a source node first chooses a value for λ in the order of $\frac{1}{\log_c n}$. For an estimated phase-one tree size in the order of λn, we calculate K from (1) as follows:

$$K = \log_c(\lambda n(c-1)+1) - 1. \quad (2)$$

Because K may not be an integer, the source has to send either $M(\lfloor K \rfloor, id)$ or $M(\lceil K \rceil, id)$ to initiate the execution of the distributed algorithm ACOM-1 in Fig. 2. The rounding introduces inaccuracy. If $M(\lfloor K \rfloor, id)$ is used, the estimated size of the phase-one tree is no longer λn. Instead, it is $\sum_{i=0}^{\lfloor K \rfloor} c^i$, which can be as small as $\frac{1}{c}\lambda n$. On the other hand, if $M(\lceil K \rceil, id)$ is used, the estimated size of the phase-one tree is $\sum_{i=0}^{\lceil K \rceil} c^i$, which can be as large as $c\lambda n$. To keep the tree size around λn, the phase-one tree should have levels 0 through $\lfloor K \rfloor + 1$, but the $(\lfloor K \rfloor + 1)$th level should have been only partially populated. Not all nodes at level $(\lfloor K \rfloor + 1)$ will be included in the tree. Instead, each node at this level only has a probability p of being included. p is calculated below.

$$\left(\sum_{i=0}^{\lfloor K \rfloor} c^i\right) + c^{\lfloor K \rfloor + 1} p = \lambda n,$$

$$p = \frac{\lambda n - \frac{c^{\lfloor K \rfloor + 1} - 1}{c-1}}{c^{\lfloor K \rfloor + 1}}.$$

The source s sends out $M(\lfloor K \rfloor, p, id)$ with an additional field p. When a node x receives a message $M(0, p, id)$, it forwards the message to each of its c_x random neighbors with a probability p. This algorithm, called ACOM-2, is given in Fig. 3.

Alternatively we can use a different concept, called *tickets*, to control the phase-one tree size. Instead of using a k field to carry the depth of the tree, we use a t field to carry the size of the tree. The message format is $M(t, id)$, where t is the number of tickets, each representing the permission of including one node in the subtree rooted at the receiver of the message. The source node sends out $M(\lambda n - 1, id)$. When a node x receives $M(t, id)$, it consumes one ticket by subtracting one from t. It then evenly distributes the remaining tickets among its c_x children at the next level of the tree. The algorithm is called ACOM-3 and is given in Fig. 4. Whereas λn gives an upper bound on the expected size of the phase-one tree in ACOM-1 and ACOM-2 according to Lemma 5, it sets the upper bound on the actual tree size in ACOM-3.

7 Simulation

7.1 Setup

An overlay network is constructed among 10,000 nodes. We assume the nodes come from all over the Internet, rather than being clustered in a few places. In [29], Mukherjee found that the end-to-end packet delay on the Internet can be modeled by a shifted Gamma distribution, which is a long-tail distribution. The shape parameter varies from approximately 1.0 during low loads to 6.0 during high loads on the backbone. In this paper, we set the shape parameter to be 4.0 and the average packet delay to be 50 ms.[3] Because every node can be a potential data source, we do not maintain one explicit multicast tree per node. Instead, the multicast is performed (as broadcast) directly on top of the overlay network, consisting of a random graph and an optional augmented ring, as described in Section 3. No central element is available to coordinate the overlay construction or the multicast procedure. Nodes have different capacities. When the average node capacity is set to be c, the node capacities are taken from $[2, 2(c-1)]$ with uniform probabilities. If not specified otherwise, $c = 6$ by default. We will vary the value of c in a wide range in some simulations.

We implement seven multicast algorithms and their variants. The algorithms are *flooding*, *probabilistic flooding*, *limited-degree (LD) flooding*, *ACOM-1 with* $\lfloor K \rfloor$, *ACOM-2*,

3. Changing the average packet delay in the simulation will proportionally scale the curves in the latency comparison, but will not change the shape of the latency curves.

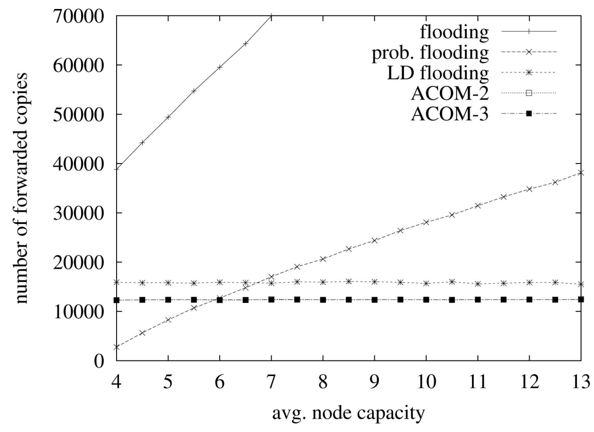


Fig. 5. Comparison of different algorithms of traffic volume for delivering one multicast message.

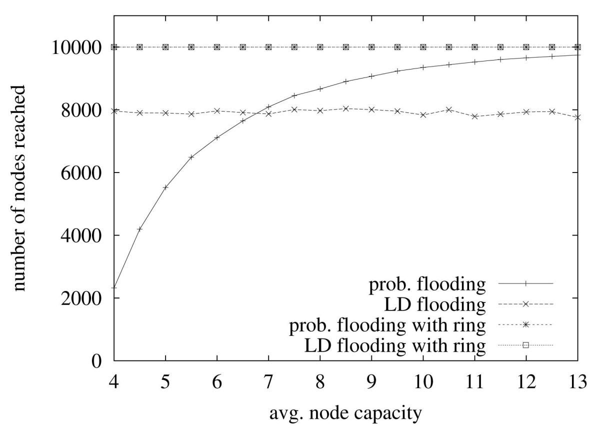


Fig. 6. Number of nodes reached by a multicast message. Not all algorithms deliver the message to all nodes.

ACOM-3, and *CAM-Chord* [19]. Flooding is one of the most popular broadcast algorithms on the IP networks; reverse path forwarding is one example. In our case, each node forwards a copy of a message to all random neighbors when it receives the message for the first time. In order to control the overhead, probabilistic flooding forwards a copy of the message to a random neighbor with a certain probability, which means not all neighbors of a node will receive the message forwarded by the node. On the other hand, LD flooding only forwards a copy of the message to a fixed number of random neighbors. ACOM-1 with $\lfloor K \rfloor$ is the algorithm in Fig. 2 with K calculated from (2). Because ACOM-2 and ACOM-3 are better algorithms (Section 6), we only present their results in most cases. Proximity on the ring is implemented by default. The above six algorithms are all designed based on a random network. CAM-Chord is designed based on a more complex DHT network. It has $O(\log_c n)$ times more neighbors per node, which means multiple times higher overhead for maintaining the network, but it is more efficient in delivering a multicast message. ACOM and CAM-Chord represent a trade-off between cheaper network maintenance and more efficient multicasting (Section 7.6).

The performance metrics that we study include

1. *the traffic volume for delivering one message*, which is measured by the number of copies that are forwarded by all nodes,
2. *the average length of the delivery paths*, which is measured by the average number of overlay hops it takes a message to reach a node,
3. *the average delivery latency*, which is measured by the average time it takes a message to reach a node,
4. *hop distribution*, which is the distribution of the delivery-path lengths for all nodes,
5. *delay distribution*, which is the distribution of the latency values for all nodes, and
6. *the maintenance overhead*, which is the average number of control messages sent for each node join/departure.

7.2 Performance Evaluation

The first set of simulations compares the average performance of various algorithms in delivering messages. Fig. 5 shows the traffic volumes caused by these algorithms with respect to the average node capacity. In probabilistic flooding, the probabilistic for a node to forward a message to a neighbor is 30 percent. In LD flooding, each node forwards a message to two neighbors. In ACOM-2 and ACOM-3, $\lambda = 0.25$. According to the figure, flooding is the clear loser because of too much traffic. The traffic volume of probabilistic flooding is small when the average node capacity is small. The traffic volume of LD flooding is modest and insensitive to the node capacity. The problem is that neither of them ensures that a message will reach all nodes, as demonstrated in Fig. 6. One way to solve this problem is to incorporate the segmented ring traversal proposed in this paper. A ring is augmented to the overlay network. When a node receives a message for the first time, it not only forwards the message to random neighbors but also starts traversing the adjacent ring segment until reaching a node that has also received the message. But, the segmented ring traversal comes with a cost; it forwards $n(= 10{,}000)$ copies of the message, which brings up the traffic volume as shown in Fig. 7. In comparison, the traffic volumes caused by ACOM-2 and ACOM-3 are both much smaller.

Fig. 8 compares the average lengths of the delivery paths. It takes the least number of hops for flooding to reach the nodes. ACOM-2 performs slightly better than ACOM-3. Both ACOM-2 and ACOM-3 perform better than LD flooding, but worse than flooding. Probabilistic flooding performs well

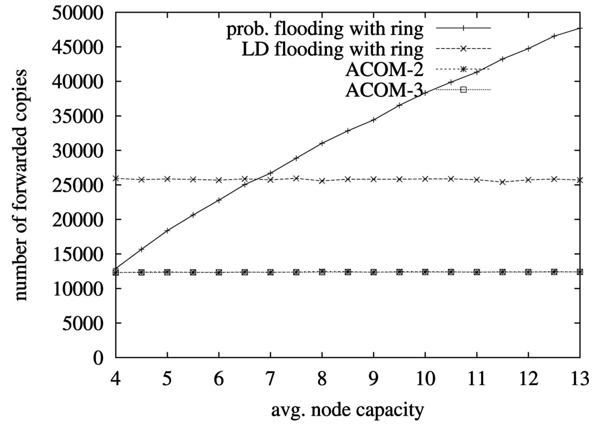


Fig. 7. Comparison on traffic volume for delivering one message. All algorithms perform segmented ring traversal.

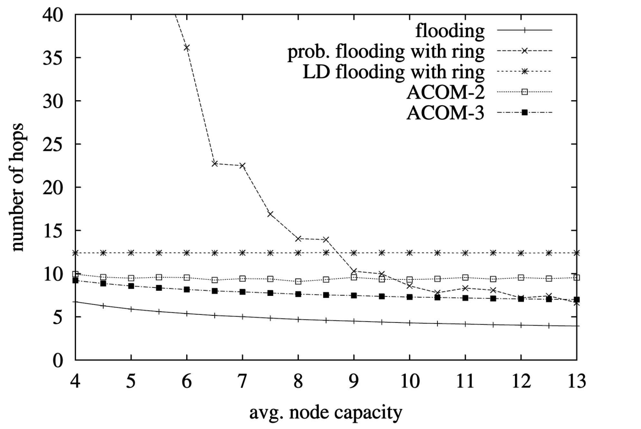


Fig. 8. Comparison of the algorithms on average length of delivery paths.

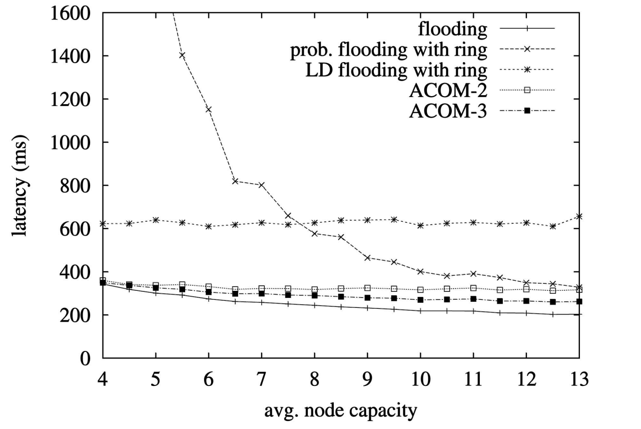


Fig. 9. Comparison of the algorithms on average latency of delivering a message.

when the average node capacity is large, but that comes with a large overhead, shown in Fig. 7.

Due to the proximity measure that is implemented on the ring (Section 5.5), ACOM-2 and ACOM-3 perform closer to flooding in terms of latency than in terms of hops. As shown in Fig. 9, proximity has less impact on LD flooding because it traverses on shorter ring segments than ACOM-2 and ACOM-3. The average latency values of ACOM-2 and ACOM-3 are slightly worse than that of flooding but much better than that of LD flooding.

We also perform scalability evaluation with the number of nodes in the network ranging from 2,000 to 20,000. Fig. 10 shows that the traffic volume increases linearly with the network size for all four algorithms under comparison. Fig. 11 shows that, except for probabilistic flooding, the average latency values of the algorithms are not very sensitive to the network size.

7.3 Performance Trade-off in ACOM

The second set of simulations studies the performance of ACOM with respect to the value of λ. Figs. 12, 13, and 14 demonstrate a trade-off in ACOM between communication complexity (traffic volume) and hop complexity (number of hops and latency). By increasing λ, ACOM can reduce the latency at the cost of higher traffic volume, and vice versa.

In Fig. 12, a larger λ value means a larger phase-one tree, which increases the traffic volume that constructs the

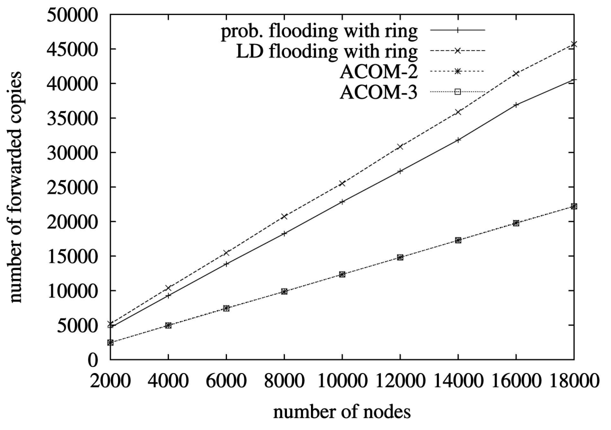


Fig. 10. Scalability comparison on traffic volume for delivering one message with respect to network size.

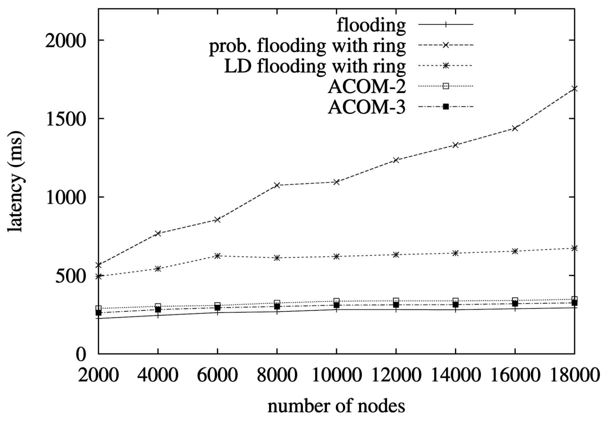


Fig. 11. Comparison of the algorithms on average length of delivery paths with respect to network size.

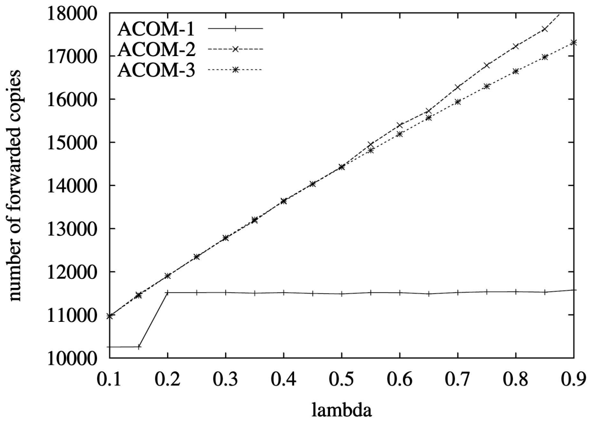


Fig. 12. Comparison of ACOM algorithms on traffic volume with respect to λ.

tree. Note that the traffic volume of phase two is always $n(=10,000)$ forwarded copies. On the other hand, a larger phase-one tree means smaller ring segments, which reduces the number of hops and, thus, latency in phase two, also causing overall hop/latency decrease (Figs. 13 and 14). The performance curve of ACOM-1 with $\lfloor K \rfloor$ is a staircase with respect to λ. That is because changing λ does not continuously change the value of $\lfloor K \rfloor$, according to (2).

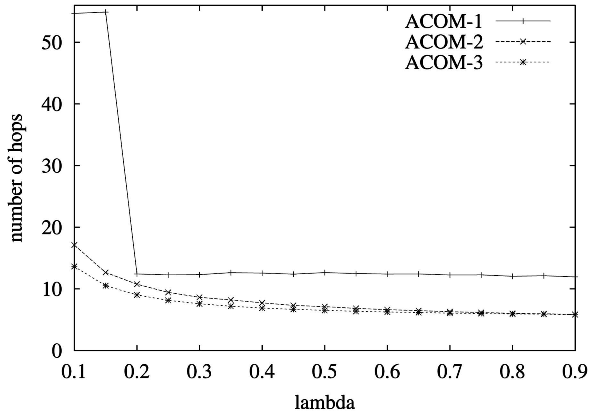


Fig. 13. Comparison of ACOM algorithms on average length of delivery paths with respect to λ.

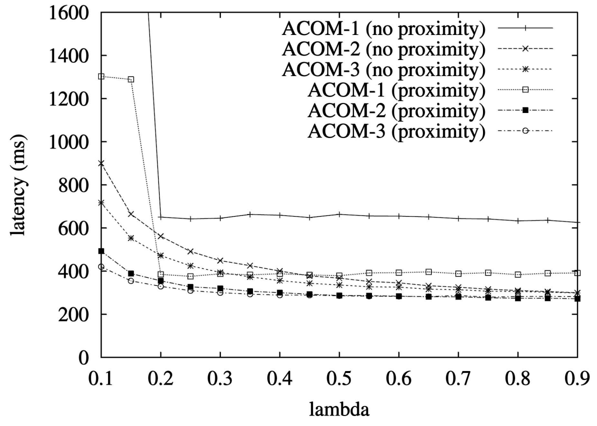


Fig. 14. Comparison of ACOM algorithms on average latency with and without implementing proximity.

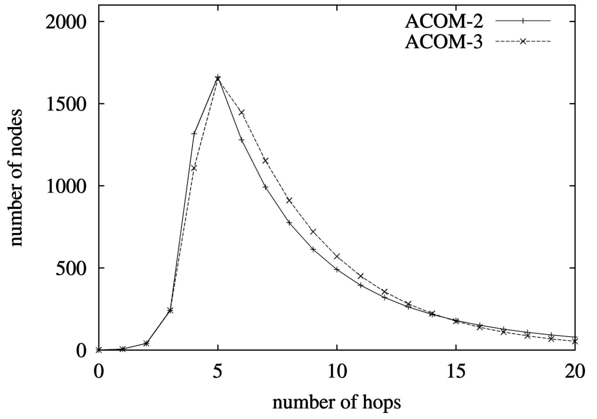


Fig. 15. Hop distribution.

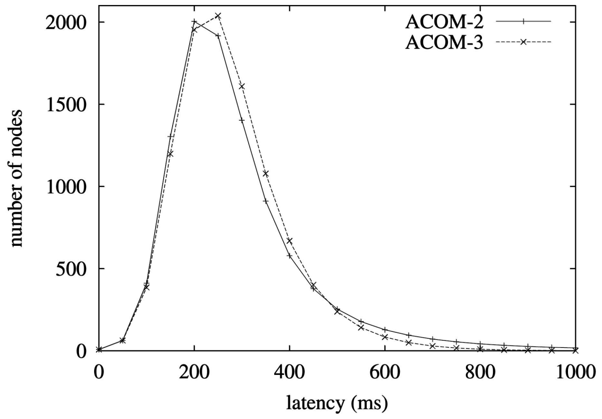


Fig. 16. Delay distribution.

7.4 Delay and Proximity

The third set of simulations studies the impact of proximity. Our ring is an "unrestricted" ring, where a node can be located anywhere. This gives us the freedom of moving nearby nodes adjacent on the ring (Section 5.5), which reduces the latency of performing the segmented ring traversal in phase two. Fig. 14 shows that the impact of proximity is greater when λ is smaller. That is because a smaller value for λ means a smaller phase-one tree and, consequently, larger ring segments in phase two, which gives proximity more impact.

7.5 Hop and Delay Distributions

The fourth set of simulations studies the hop distribution and the delay distribution, which are shown in Fig. 13 and Fig. 14, respectively. Because the average delay of an overlay link is 50 ms, 20 hops in Fig. 15 corresponds to 1,000 ms in Fig. 16. The value of λ is chosen to be 0.25. In Fig. 15, over 93.2 percent of all nodes receive the message within twice the average number of hops, which is 9.43 for ACOM-2 and 8.14 for ACOM-3. The delay distribution is more concentrated. In Fig. 16, over 96.4 percent of all nodes receive the message within twice the average latency, which is 327.17 ms for ACOM-2 and 309.62 ms for ACOM-3.

7.6 Network Maintenance Overhead and ACOM versus CAM-Chord

The algorithms evaluated so far are all based on random overlay networks. Because these algorithms use the same underlying overlay topology in the simulations, the network maintenance overhead is the same for them.

The goal of ACOM is to support any-source capacity-constrained multicast with dynamic membership. We have elaborated in the introduction and the related work that most existing systems do not meet these requirements. CAM-Chord is an exception, but it is DHT-based and requires $O(c \log_c n)$ neighbors per node, which is $O(\log_c n)$ times more than ACOM. The fifth set of simulations compares CAM-Chord and CAM in terms of multicast efficiency and network maintenance overhead.

Thanks to its more complex topological structure, CAM-Chord forwards only one copy of a multicast message to each node. Therefore, it is more efficient in terms of multicasting (Fig. 17), but less efficient in terms of network maintenance (Fig. 18). Assume a TCP connection is established between two neighbors. The maintenance overhead is the summation of 1) the number of control messages used to identify the neighbors, 2) the number of control messages for establishing new TCP connections between neighbors, and 3) the number of control messages for tearing down the TCP connections between nodes that cease to be neighbors. The maintenance overhead of ACOM increases slowly with the network size, and it is far less than that of CAM-Chord.

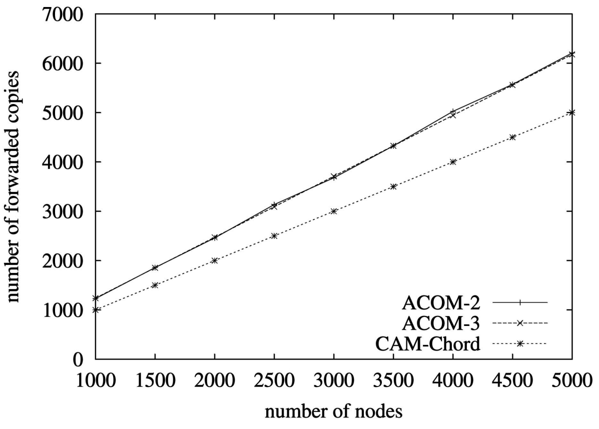


Fig. 17. CAM-Chord forwards one copy of the multicast message for each node, which is optimal. In total, ACOM forwards about 24 percent more copies than necessary due to its random structure. CAM-Chord achieves the optimality at the cost of $O(\log_c n)$ times more neighbors per node and the more complex DHT-based network structure, which is harder to repair once it is fractured.

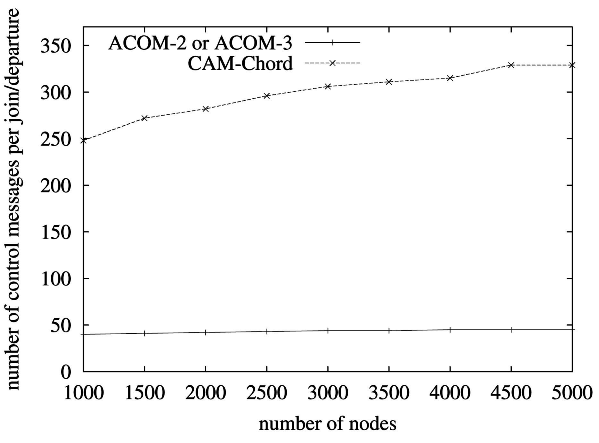


Fig. 18. CAM-Chord has many more neighbors per node than ACOM. Its overhead, in terms of number of control messages per node join/departure, is much higher than that of ACOM, which is a structurally-simpler non-DHT network. Therefore, CAM-Chord and ACOM represent a trade-off between topology maintenance overhead and multicast overhead.

CAM-Chord and ACOM represent a trade-off between topology maintenance overhead and multicast overhead. They provide different options for practitioners to choose based on the system requirements that may be in favor of more efficient multicasting or low overhead maintenance. The average latency values of CAM-Chord and ACOM are comparable and the simulation results are omitted.

8 Conclusion

This paper proposes an any-source overlay multicast service on top of a random overlay network with an augmented ring. It is able to deliver a multicast message from any source with a hop complexity of $O(\log_c n)$ and a communication complexity of $n + \frac{n}{\log_c n}$. Remarkably, this is achieved without maintaining any explicit multicast trees. We provide a rigorous analysis and perform extensive simulations to evaluate the performance of the system. We present three distributed multicast algorithms and address a variety of issues, including overlay maintenance, parameter measurement, low-capacity nodes, proximity, and dynamic capacities.

Appendix

Proofs

We first give the upper and lower bounds of Q_K and then derive the hop complexity and the communication complexity of the algorithm. The definitions for Q_K and other symbols can be found in Section 4.3 and Section 3.

Lemma 1. $E(Q_i) \leq \sum_{j=0}^{i} c^j$.

Proof. The number of nodes at level i is bounded by the summation of the capacities of all nodes at level $(i-1)$. Namely,

$$q_i \leq \sum_{j=1}^{q_{i-1}} c_{i-1,j}.$$

The expected value of q_i, under the condition that there are q_{i-1} nodes at level $(i-1)$, is calculated below. $c_{i-1,j}$, $j \in [1..q_{i-1}]$, is a random variable whose expected value is c.

$$E(q_i \mid q_{i-1}) \leq E\left(\sum_{j=1}^{q_{i-1}} c_{i-1,j}\right) = \sum_{j=1}^{q_{i-1}} E\big(c_{i-1,j}\big) = q_{i-1}c.$$

Take the expected value on both sides with respect to q_{i-1} and, because $E(E(q_i|q_{i-1})) = E(q_i)$, we have

$$E(q_i) \leq E(q_{i-1})c.$$

By recursively applying the above inequality, we have

$$E(q_i) \leq c^i E(q_0) = c^i.$$

It follows that

$$E(Q_i) = E\left(\sum_{j=0}^{i} q_j\right) \leq \sum_{j=0}^{i} c^j.$$

□

Lemma 2. *For any set S of conditions, it must be true that*

$$E(Q_i|S) \geq E\left(Q_i \mid Q_i \leq \sum_{j=0}^{i} c^j, S\right).$$

Proof. $E(Q_i \mid S)$ is the expected value of Q_i among all possible multicast instances that satisfy S.

$$E\left(Q_i \mid Q_i \leq \sum_{j=0}^{i} c^j, S\right)$$

is the expected value of Q_i among all possible multicast instances that not only satisfy S but also $Q_i \leq \sum_{j=0}^{i} c^j$. The difference between them is that $E(Q_i \;\; Q_i \leq \sum_{j=0}^{i} c^j, S)$ excludes all Q_i instances that are greater than $\sum_{j=0}^{i} c^j$. Since it excludes large values, we certainly have that

$$E(Q_i \mid S) \geq E\left(Q_i \mid Q_i \leq \sum_{j=0}^{i} c^j, S\right).$$

□

Lemma 3. *For any set S of conditions that place no artificial restriction on a node's capacity, it must be true that*

$$E\left(Q_i \mid Q_i \le \sum_{j=0}^{i} c^j, S\right) > c\left(1-\frac{\lambda}{C^{K-i}}\right)E\left(Q_{i-1} \mid Q_i \le \sum_{j=0}^{i} c^j, S\right).$$

Proof. Let T_i be the subtree that consists of levels 0 through i of a phase-one tree. The size of T_i is Q_i. During the formation of T_i, each node x at levels 0 through $(i-1)$ makes c_x attempts to include its random neighbors in the tree at the next level. Because there are less than Q_i nodes in the tree, each attempt has a probability of at least $(1-\frac{Q_i}{n})$ of finding that a random neighbor is outside the tree and thus can be brought in, which increases the tree size by one. There are, in total, $\sum_{l=0}^{i-1}\sum_{j=1}^{q_l} c_{l,j}$ attempts at levels 0 through $(i-1)$. Therefore,

$$Q_i > \sum_{l=0}^{i-1}\sum_{j=1}^{q_l} c_{l,j}\left(1-\frac{Q_i}{n}\right). \quad (4)$$

One may be puzzled by the above inequality among random variables. The correct interpretation is that the inequality will hold for any given multicast instance.

Given any set of values, q_l, $l \in [1..i-1]$, which appears in $\{q_1,\dots,q_i | Q_i \le \sum_{j=0}^{i} c^j, S\}$, we compute the conditional expected value of Q_i.

$$\begin{aligned}
&E\left(Q_i \mid q_l,\ l\in[1..i-1],\ Q_i \le \sum_{j=0}^{i} c^j,\ S\right)\\
&> E\left(\sum_{l=0}^{i-1}\sum_{j=1}^{q_l} c_{l,j}\left(\frac{1-Q_i}{n}\right) \mid q_l,\ l\in[1..i-1],\ Q_i \le \sum_{j=0}^{i} c^j,\ S\right) \quad \text{by (4)}\\
&\ge E\left(\sum_{l=0}^{i-1}\sum_{j=1}^{q_l} c_{l,j}\left(1-\frac{\sum_{j=0}^{i} c^j}{n}\right) \mid q_l,\ l\in[1..i-1],\ Q_i \le \sum_{j=0}^{i} c^j,\ S\right) \quad \text{by condition } Q_i \le \sum_{j=0}^{i} c^j\\
&= \left(1-\frac{\sum_{j=0}^{i} c^j}{n}\right)\sum_{l=0}^{i-1}\sum_{j=1}^{q_l} E\left(c_{l,j} \mid q_l,\ l\in[1..i-1],\ Q_i \le \sum_{j=0}^{i} c^j,\ S\right)\\
&= c\left(1-\frac{\sum_{j=0}^{i} c^j}{n}\right)\left(\sum_{l=0}^{i-1}\sum_{j=1}^{q_l} 1\right) \quad \text{by lemma assumption, S places no restriction on } c_{l,j}\\
&= c\left(1-\frac{\sum_{j=0}^{i} c^j}{n}\right)Q_{i-1}\\
&> c\left(1-\frac{\lambda}{c^{K-i}}\right)Q_{i-1} \quad \text{by (1).}
\end{aligned}$$

Based on the above inequality, take the expected value of Q_i over q_l, $l \in [1..i-1]$, under the condition of $Q_i \le \sum_{j=0}^{i} c^j$ and S.

$$E\left(Q_i \mid Q_i \le \sum_{j=0}^{i} c^j,\ S\right) > c(1-\frac{\lambda}{C^{K-i}})E(Q_{i-1} \mid Q_i \le \sum_{j=0}^{i} c^j,\ S).$$

□

Lemma 4. $E(Q_i) > (1-\frac{\lambda}{c^{K-i-1}(c-1)})c^i$.

Proof. By Lemma 2,

$$E(Q_i) \ge E\left(Q_i \mid Q_i \le \sum_{j=0}^{i} c^j\right).$$

Let $S_0 = \emptyset$. By Lemma 3,

$$E(Q_i) > c\left(1-\frac{\lambda}{C^{K-i}}\right)E\left(Q_{i-1} \mid Q_i \le \sum_{j=0}^{i} c^j,\ S_0\right).$$

Let $S_1 = S_0 + \{Q_i \le \sum_{j=0}^{i} c^j\}$. By Lemmas 2 and 3,

$$\begin{aligned}
&E(Q_i)\\
&> c\left(1-\frac{\lambda}{C^{K-i}}\right)E(Q_{i-1} \mid S_1)\\
&\ge c\left(1-\frac{\lambda}{C^{K-i}}\right)E\left(Q_{i-1} \mid Q_{i-1} \le \sum_{j=0}^{i-1} c^j,\ S_1\right)\\
&> c^2\left(1-\frac{\lambda}{C^{K-i}}\right)\left(1-\frac{\lambda}{C^{K-i+1}}\right)E\left(Q_{i-2} \mid Q_{i-1} \le \sum_{j=0}^{i-1} c^j,\ S_1\right).
\end{aligned}$$

Repeat this process by recursively applying Lemmas 2 and 3. We have

$$\begin{aligned}
E(Q_i) &> c^i\prod_{j=1}^{i}\left(1-\frac{\lambda}{c^{K-j}}\right)E(q_0)\\
&= c^i\prod_{j=1}^{i}\left(1-\frac{\lambda}{c^{K-j}}\right).
\end{aligned}$$

It can be shown by induction that $\prod_{j=1}^{i}(1-\frac{\lambda}{c^{K-j}}) > 1-\frac{\lambda}{c^{K-i-1}(c-1)}$. Therefore,

$$E(Q_i) > \left(1-\frac{\lambda}{c^{K-i-1}(c-1)}\right)c^i.$$

□

Lemma 5. $(1-\frac{1}{c})(1-\frac{\lambda c}{c-1})\lambda n < E(Q_K) \le \lambda n$.

Proof. By Lemma 1, $E(Q_K) \le \sum_{i=0}^{K} c^i = \lambda n$. By Lemma 4,

$$E(Q_K) > \left(1-\frac{\lambda c}{c-1}\right)c^K. \quad (5)$$

From (1), $c^K = \frac{\lambda n(c-1)+1}{c} > \frac{\lambda n(c-1)}{c}$. Applying this to (5), we have

$$E(Q_K) > \left(1 - \frac{1}{c}\right)\left(1 - \frac{\lambda c}{c-1}\right)\lambda n.$$

□

Theorem 1 (Hop Complexity). *The expected number of hops for a multicast message to reach any node is bounded by*

$$\log_c(\lambda n) + \frac{1}{2(1-\frac{1}{c})(1-\frac{\lambda c}{c-1})\lambda}.$$

Proof. By (1), $\sum_{i=0}^{K} c^i = \lambda n$. Hence, $c^K < \lambda n$. $K < \log_c(\lambda n)$, which means that phase one terminates in no more than $\log_c(\lambda n)$ hops. By Lemma 5, after phase one, the ring is partitioned into more than $(1-\frac{1}{c})(1-\frac{\lambda c}{c-1})\lambda n$ segments. Phase two delivers the message in parallel along these segments. The expected length of a segment is bounded by

$$\frac{n}{\left(1-\frac{1}{c}\right)\left(1-\frac{\lambda c}{c-1}\right)\lambda n} = \frac{1}{\left(1-\frac{1}{c}\right)\left(1-\frac{\lambda c}{c-1}\right)\lambda}.$$

In phase two, the average number of hops for the message to reach a node is half of the segment length. Combining phase one and phase two, the expected number of hops for a multicast message to reach any node is bounded by

$$\log_c(\lambda n) + \frac{1}{2\left(1-\frac{1}{c}\right)\left(1-\frac{\lambda c}{c-1}\right)\lambda}.$$

□

Corollary 1 (Hop Complexity). *If $\lambda = \Theta\left(\frac{1}{\log_c n}\right)$ and $c \geq 2$, then the expected number of hops for a multicast message to reach any node is $O(\log_c n)$.*

Proof. By Theorem 1, the upper bound on the expected number of hops is

$$\begin{aligned}
&\log_c(\lambda n) + \frac{1}{2\left(1-\frac{1}{c}\right)\left(1-\frac{\lambda c}{c-1}\right)\lambda}\\
&= (\log_c n - \Theta(\log_c \log_c n)) +\\
&\quad \frac{1}{2\left(1-\frac{1}{c}\right)\left(1-\Theta\left(\frac{1}{\log_c n}\right)\frac{c}{c-1}\right)\Theta\left(\frac{1}{\log_c n}\right)}\\
&= \Theta(\log_c n) + \frac{1}{\Theta\left(\frac{1}{\log_c n}\right)}\\
&= \Theta(\log_c n) + \Theta(\log_c n) = \Theta(\log_c n).
\end{aligned}$$

□

Theorem 2 (Communication Complexity). *The expected number of copies of a multicast message that are transmitted is bounded by $(1+\lambda)n$.*

Proof. The number X of copies that are transmitted in phase one is equal to the summation of the capacities of all nodes at levels zero through $K-1$.

$$X = \sum_{i=0}^{K-1}\sum_{j=1}^{q_i} c_{i,j}.$$

X is the summation of Q_{K-1} independent random variables, where $Q_{k-1} = \sum_{i=0}^{K-1} q_i$. Q_{K-1} is itself a random variable. Given a value of Q_{K-1}, the conditional expected value of X is

$$E(X \mid Q_{K-1}) = \sum_{i=0}^{K-1}\sum_{j=1}^{q_i} E(c_{i,j}) = \sum_{i=0}^{K-1}\sum_{j=1}^{q_i} c = cQ_{K-1}.$$

Take the expected value on both sides with respect to Q_{K-1}. By (3) and (1), we have

$$E(X) = cE(Q_{K-1}) \leq c\sum_{i=0}^{K-1} c^i < \lambda n.$$

Phase two sends a fixed number of n copies in the segmented ring traversal. Therefore, the expected number of copies that are transmitted for a multicast message is $(1+\lambda)n$. □

Corollary 2 (Communication Complexity). *If $\lambda = \Theta(\frac{1}{\log_c n})$, then the expected number of copies of a multicast message that are transmitted is bounded by $n + O(\frac{n}{\log_c n})$.*

Proof. Directly from Theorem 2. □

ACKNOWLEDGMENTS

This work is in part supported by the National Natural Science Foundation of China under grant 60573142 and by Shanghai Leading Academic Discipline Project (Project Number T0502).

REFERENCES

[1] G. Banavar, M. Chandra, B. Nagarajaro, R. Strom, and C. Sturman, "An Efficient Multicast Protocol for Content-Based Publish-Subscribe System," *Proc. Int'l Conf. Distributed Computing Systems (ICDCS '98),* May 1998.

[2] Y.H. Chu, S. Rao, S. Seshan, and H. Zhang, "A Case for End System Multicast," *IEEE J. Selected Areas in Comm.,* vol. 20, no. 8, Oct. 2002.

[3] J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, and J. O'Toole, "Overcast: Reliable Multicasting with an Overlay Network," *Proc. Symp. Operating Systems Design and Implementation (OSDI '00),* Oct. 2000.

[4] C.K.S. Banerjee and B. Bhattacharjee, "Scalable Application Layer Multicast," *Proc. ACM SIGCOMM '02,* Aug. 2002.

[5] B. Zhang, S. Jamin, and L. Zhang, "Host Multicast: A Framework for Delivering Multicast to End Users," *Proc. INFOCOM '02,* June 2002.

[6] G.-I. Kwon and J.W. Byers, "ROMA: Reliable Overlay Multicast with Loosely Coupled TCP Connections," *Proc. INFOCOM '04,* Mar. 2004.

[7] P.M. Zhi Li, "Impact of Topology on Overlay Routing Service," *Proc. INFOCOM '04,* Mar. 2004.

[8] Y. Shavitt and T. Tankel, "On the Curvature of the Internet and Its Usage for Overlay Construction and Distance Estimation," *Proc. INFOCOM '04,* Mar. 2004.

[9] F. Baccelli, A. Chaintreau, Z. Liu, A. Riabov, and S. Sahu, "Scalability of Reliable Group Communication Using Overlays," *Proc. INFOCOM '04,* Mar. 2004.

[10] V. Pappas, B. Zhang, A. Terzis, and L. Zhang, "Fault-Tolerant Data Delivery for Multicast Overlay Networks," *Proc. Int'l Conf. Distributed Computing Systems (ICDCS '04),* Mar. 2004.

[11] J.A. Dejan Kosti, A. Rodriguez, and A. Vahdat, "Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh," *Proc. Symp. Operating Systems Principles (SOSP '03),* Oct. 2003.

[12] S. Banerjee, C. Kommareddy, B.B.K. Kar, and S. Khuller, "Construction of an Efficient Overlay Multicast Infrastructure for Real-Time Applications," *Proc. INFOCOM '03,* Mar. 2003.

[13] A. Riabov and L.Z. Zhen Liu, "Overlay Multicast Trees of Minimal Delay," *Proc. Int'l Conf. Distributed Computing Systems (ICDCS '04),* Mar. 2004.

[14] H. Yamaguchi, A. Hiromori, T. Higashino, and K. Taniguchi, "An Autonomous and Decentralized Protocol for Delay Sensitive Overlay Multicast Tree," *Proc. Int'l Conf. Distributed Computing Systems (ICDCS '04),* Mar. 2004.

[15] S. Zhuang, B. Zhao, A. Joseph, R. Katz, and J. Kubiatowicz, "Bayeux: An Architecture for Scalable and Fault-Tolerant Wide-Area Data Dissemination," *Proc. 11th Int'l Workshop Network and Operating System Support for Digital Audio and Video (NOSSDAV '01),* June 2001.

[16] R. Zhang and Y.C. Hu, "Borg: A Hybrid Protocol for Scalable Application-Level Multicast in Peer-to-Peer Networks," *Proc. Int'l Workshop Network and Operating System Support for Digital Audio and Video (NOSSDAV '03),* 2003.

[17] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Application-Level Multicast Using Content-Addressable Networks," *Proc. Int'l Workshop Networked Group Comm. (NGC '01),* 2001.

[18] M. Castro, M. Jones, A.-M. Kermarrec, A. Rowstron, M. Theimer, H. Wang, and A. Wolman, "An Evaluation of Scalable Application-Level Multicast Built Using Peer-to-Peer Overlays," *Proc. INFOCOM '03,* Apr. 2003.

[19] Z. Zhang, S. Chen, Y. Ling, and R. Chow, "Capacity-Aware Multicast Algorithms in Heterogeneous Overlay Networks," *IEEE Trans. Parallel and Distributed Systems,* special section on algorithm design and scheduling techniques (realistic platform models) for heterogeneous clusters, vol. 17, no. 2, pp. 135-147, Feb. 2006.

[20] S. Shi, J. Turner, and M. Waldvogel, "Dimensioning Server Access Bandwidth and Multicast Routing in Overlay Networks," *Proc. Int'l Workshop Network and Operating System Support for Digital Audio and Video (NOSSDAV '01),* June 2001.

[21] S. Shi and J. Turner, "Routing in Overlay Multicast Networks," *Proc. INFOCOM '02,* June 2002.

[22] B.Y. Zhao, J.D. Kubiatowicz, and A.D. Joseph, "Tapestry: An Infrastructure for Fault-Tolerant Wide-Area Location and Routing," Univ. of California Berkeley Technical Report UCB/CSD-01-1141, Apr. 2001.

[23] A. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems," *Proc. Middleware '01,* Nov. 2001.

[24] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content Addressable Network," *Proc. ACM SIGCOMM '01,* Aug. 2001.

[25] S. El-Ansary, L.O. Alima, P. Brand, and S. Haridi, "Efficient Broadcast in Structured P2P Networks," *Proc. Int'l Workshop Peer-to-Peer Systems (IPTPS '03),* Feb. 2003.

[26] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," *Proc. ACM SIGCOMM '01,* pp. 149-160, Aug. 2001.

[27] M. Kaashoek and D. Karger, "Koorde: A Simple Degree-Optimal Distributed Hash Table," *Proc. Second Int'l Workshop Peer-to-Peer Systems (IPTPS '03),* Feb. 2003.

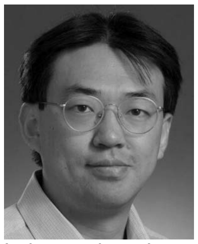
[28] T.S.E. Ng and H. Zhang, "Predicting Internet Network Distance with Coordinates-Based Approaches," *Proc. IEEE INFOCOM '02,* June 2002.

[29] A. Mukherjee, "On the Dynamics and Significance of Low Frequency Components of Internet Load," *Internetworking: Research and Experience,* vol. 5, no. 4, pp. 163-205, 1994.

Shiping Chen received the BS degree in electrical engineering from JiangXi University of China in 1984. He received the MS and PhD degrees in computer science from the Institute of Computing Technology of the Chinese Academy of Sciences and Fudan University in 1990 and 2006, respectively. He joined the University of Shanghai for Science and Technology in 1990 and is currently a full professor in the Computer Science Department. He is also the director of the network center of the university. His research interests include peer-to-peer networks, network communications, and database systems.

Baile Shi joined the Department of Computer and Information Technology of Fudan University in 1975. He was promoted to an associate and then a full professor in 1980 and 1985, respectively. He was the department chair from 1985 to 1996. His research field is database theories and applications. He has published more than 70 papers in the top Chinese journals and written more than 10 textbooks. He has won numerous awards, including one national science and technology advancement award, one Guanghua award, nine Shanghai science and technology advancement awards, and four textbook awards.

Shigang Chen received the BS degree in computer science from the University of Science and Technology of China in 1993. He received the MS and PhD degrees in computer science from the University of Illinois at Urbana-Champaign in 1996 and 1999, respectively. After graduation, he worked with Cisco Systems for three years before joining the University of Florida as an assistant professor in 2002. His research interests include network security, peer-to-peer networks, and sensor networks. He received an IEEE Communications Society Best Tutorial Paper Award in 1999. He was a guest editor of the *ACM/Baltzer Journal of Wireless Networks (WINET)* and the *IEEE Transactions on Vehicle Technologies*. He served as a technical program committee cochair for the Computer and Network Security Symposium of IEEE IWCCC 2006, a technical program committe vice chair for IEEE MASS 2005, a vice general chair for QShine 2005, a technical program committee cochair for QShine 2004, and a technical program committee member for many conferences including IEEE ICNP, IEEE INFOCOM, IEEE SANS, IEEE ISCC, IEEE Globecom, etc.



Ye Xia received the PhD degree from the University of California, Berkeley, in 2003, the MS degree in 1995 from Columbia University, and the BA degree in 1993 from Harvard University, all in electrical engineering. He became an assistant professor in the Computer and Information Science and Engineering department at the University of Florida in August 2003. Between June 1994 and August 1996, he was a member of the technical staff at Bell Laboratories, Lucent Technologies in New Jersey. His research interests are in the computer networking area, including performance evaluation of network protocols and algorithms, congestion control, resource allocation, and load balancing on peer-to-peer networks. He is also interested in probability theory, stochastic processes, and queuing theory.