Capacity-Aware Multicast Algorithms on Heterogeneous Overlay Networks

Zhan Zhang, Shigang Chen, Yibei Ling, *Member*, *IEEE*, and Randy Chow, *Member*, *IEEE*

Abstract—The global deployment of IP multicast has been slow due to the difficulties related to heterogeneity, scalability, manageability, and lack of a robust interdomain multicast routing protocol. Application-level multicast becomes a promising alternative. Many overlay multicast systems have been proposed in recent years. However, they are insufficient in supporting applications that require any-source multicast with varied host capacities and dynamic membership. In this paper, we propose two capacity-aware multicast systems that focus on host heterogeneity, any source multicast, dynamic membership, and scalability. We extend Chord and Koorde to be capacity-aware. We then embed implicit degree-varying multicast trees on top of the overlay network and develop multicast routines that automatically follow the trees to disseminate multicast messages. The implicit trees are well balanced with the workload evenly spread across the network. We rigorously analyze the expected performance of multisource capacity-aware multicasting, which was not thoroughly addressed in any previous work. We also perform extensive simulations to evaluate the proposed multicast systems.

Index Terms—Multicast, network communication.

1 Introduction

Multicast is an important network function for group communication among a distributed, dynamic set of heterogeneous nodes. Many research papers (e.g., [1], [2], [3], [4]) pointed out the disadvantages of implementing multicast at the IP level [5], [6], and argued for an application-level overlay multicast service. More recent work (e.g., [7], [8], [9], [10], [11], [12], [13]) studied overlay multicast from different aspects. In this paper, we consider four design issues in an overlay multicast system.

- *Capacity awareness:* Member hosts may vary widely in their capacities in terms of network bandwidth, memory, and CPU power. Some are able to support a large number of direct children, but others support few.
- *Any source multicast:* The system should allow any member to send data to other members. A multicast tree that is optimal for one source may be bad for another source. On the other hand, one tree per member is too costly.
- *Dynamic membership:* Members may join and leave at any time. The system must be able to efficiently maintain the multicast trees for a dynamic group.
- *Scalability:* The system must be able to scale to a large Internet group. It should be fully distributed without a single point of failure.

- *Z. Zhang is with the University of Florida, E438 CSE Building, Gainesville, FL 32611-6120. E-mail: zzhan@cise.ufl.edu.*
- *S. Chen is with the University of Florida, E460 CSE Building, PO Box 116120, Gainesville, FL 32611-6120. E-mail: sgchen@cise.ufl.edu.*
- *Y. Ling is with the Applied Research Lab, Telcordia Technologies, NJ 08854-4157. E-mail: lingy@research.telcordia.com.*
- *R. Chow is with the University of Florida, E348 CSE Building, Gainesville, FL 32611-6120. E-mail: chow@cise.ufl.edu.*

Manuscript received 15 Jan. 2005; revised 28 May 2005; accepted 14 July 2005; published online 28 Dec. 2005.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDSSI-0029-0105.

Our goal is to study capacity-aware overlay systems that support distributed applications requiring multiple-source multicast with dynamic membership. To be surveyed below, none of the existing systems meet all these requirements.

To handle dynamic groups and ensure scalability, novel proposals were made to implement multicast on top of P2P overlay networks. For example, Bayeux [14] and Borg [15] were implemented on top of Tapestry [16] and Pastry [17], respectively, and CAN-based Multicast [18] was implemented based on CAN [19]. El-Ansary et al. studied efficient broadcast in a Chord network, and their approach can be adapted for the purpose of multicast [20]. Castro et al. compares the performance of tree-based and flooding-based multicast in CAN-style versus Pastry-style overlay networks [21].

However, the above 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 only affect 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. To support efficient multicast, we should allow nodes in a P2P network to have different numbers of neighbors.

Shi et al. proved that constructing a minimum-diameter degree-limited spanning tree is NP-hard [22]. Note that the

1045-9219/06/$20.00 © 2006 IEEE Published by the IEEE Computer Society

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 [22], [23]. The algorithms do not address the dynamic membership problem, i.e., MSN join/departure.

There has been a flourish of capacity-aware multicast systems which excel in optimizing single-source multicast trees but are not suitable for multisource applications such as distributed games, teleconferencing, and virtual classrooms, which are the target applications of this paper. Bullet [24] is designed to improve the throughput of data dissemination from one source to a group of receivers. An overlay tree rooted at the source must be established. Disjoint data objects are disseminated from the source via the tree to different receivers. The receivers then communicate amongst themselves to retrieve the missing objects; these dynamic communication links, together with the tree, form a mesh, which offers better bandwidth than the tree alone. Overlay Multicast Network Infrastructure (OMNI) [25] dynamically adapts its degree-constrained multicast tree to minimize the latencies to the entire client set. Riabov et al. proposed a centralized constant-factor approximation algorithm for the problem of constructing a single-source degree-constrained minimum-delay multicast tree [26]. Yamaguchi et al. described a distributed algorithm that maintains a degree-constrained delay-sensitive multicast tree for a dynamic group [27]. The above algorithms are designed for a single source and are therefore not suitable when there are many potential sources (such as in distributed games). 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 and leaves the capacities of the majority nodes (leaves) unused, which affects the overall throughput in multisource multicasting. Third, a single tree may be partitioned beyond repair for a dynamic group.

This paper proposes two overlay multicast systems that support any-source multicast with varied host capacities and dynamic membership. We model the capacity as the maximum number of direct children to which a node is willing to forward multicast messages. We extend Chord [28] and Koorde [29] to be capacity-aware, and they are called CAM-Chord and CAM-Koorde, respectively.[1] A dedicated CAM-Chord or CAM-Koorde overlay network is established for each multicast group. We then embed *implicit* degree-varying multicast trees on top of CAM-Chord or CAM-Koorde and develop multicast routines that automatically follow the implicit multicast trees to disseminate multicast messages. Dynamic membership management and scalability are inherited features from Chord or Koorde. Capacity-aware multiple-source multicast are added features. Our analysis on CAM multicasting sheds light on the expected performance bounds with respect to the statistical distribution of host heterogeneity.

The rest of the paper is organized as follows: Section 2 gives an overview of our proposed systems. Section 3 and Section 4 describe CAM-Chord and CAM-Koorde in details respectively. Section 5 discusses some related issues. Section 6 presents the simulation results. Section 7 draws the conclusion.

1. CAM stands for Capacity-Aware Multicast.

2 Overview

Consider a multicast group G of n nodes. Each node $x \in G$ has a capacity c_x, specifying the maximum 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. Intuitively, x is able to support more direct children in a multicast tree when it has more upload bandwidth. In a heterogeneous environment, the capacities of different nodes may vary in a wide range. Our goal is to construct a resilient capacity-aware 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.

Our basic idea is to build the multicast service on top of a *capacity-aware* structured P2P network. We focus on extending Chord [28] and Koorde [29] for this purpose. The resulting systems are called CAM-Chord and CAM-Koorde, respectively. The principles and techniques developed in this paper should be easily applied to other P2P networks as well.

A CAM-Chord or CAM-Koorde overlay network is established for each multicast group. All member nodes (i.e., hosts of the multicast group) are randomly mapped by a hash function (such as SHA-1) onto an identifier ring $[0, N-1]$, where the next identifier after $N-1$ is zero. $N(=2^b)$ should be large enough such that the probability of mapping two nodes to the same identifier is negligible. Given an identifier $x \in [0, N-1]$, we define successor(x) as the node clockwise after x on the ring, and predecessor(x) as the node clockwise before x on the ring. $\widehat{x}$ refers to the node whose identifier is x; if there is not such a node, then it refers to successor(x). Node $\widehat{x}$ is said to *be responsible for* identifier x. With a little abuse of notation, x, $\widehat{x}$, successor(x), and predecessor(x) may represent a node or the identifier that the node is mapped to, depending on the appropriate context where the notations appear. Given two arbitrary identifiers x and y, $(x, y]$ is an identifier segment that starts from $(x+1)$, moves clockwise, and ends at y. The size of $(x, y]$ is denoted as $(y-x)$. Note that $(y-x)$ is always positive. It is the number of identifiers in the segment of $(x, y]$. The distance between x and y is $|x-y| = |y-x| = \min\{(y-x), (x-y)\}$, where $(y-x)$ is the size of segment $(y, x]$ and $(x-y)$ is the size of segment $(x, y]$. $(y, x]$ and $(x, y]$ form the entire identifier ring.

Before we discuss the CAMs, we briefly review Chord and Koorde. Each node x in Chord has $O(\log_2 n)$ neighbors, which are $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$, ..., of the ring away from x, respectively. When receiving a lookup request for identifier k, a node forwards the request to the neighbor closest to k. This greedy algorithm takes $O(\log_2 n)$ hops with high probability to find $\widehat{k}$, the node responsible for k. Each node in Koorde has m neighbors. A node's identifier x is represented as a base-m number. Its neighbors are derived by shifting one digit (with value 0..m-1) into x from the right side and discard x's leftmost bit to maintain the same number of digits. When x receives a lookup request for k, the routing

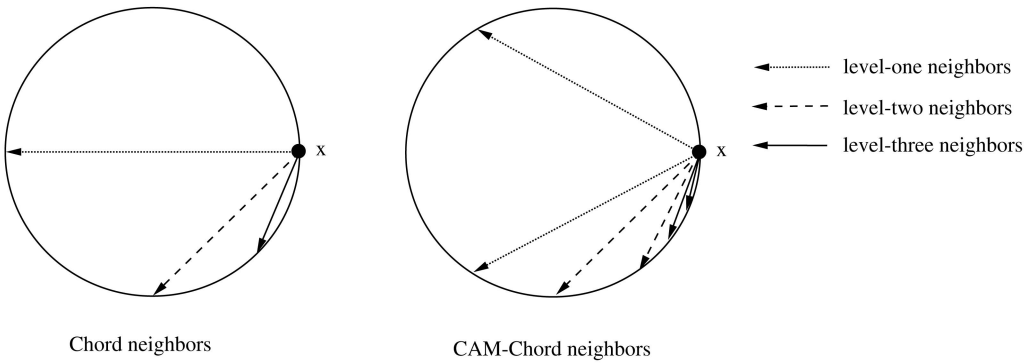

Fig. 1. Illustration of Chord versus CAM-Chord neighbors ($c_x = 3$).

path of the request represents a transformation from x to k by shifting one digit of k at a time into x from the right until the request reaches the node responsible for k. Because k has $O(\log_m n)$ digits, it takes $O(\log_m n)$ hops with high probability to resolve a lookup request. Readers are referred to the original papers for more details.

Our first system is CAM-Chord, which is essentially a base-c_x (instead of base-2) Chord with c_x variable for different nodes. The number of neighbors of a node x is $O(c_x \frac{\log n}{\log c_x})$, which is related to the node's capacity. Hence, different nodes may have different numbers of neighbors. The distances between x and its neighbors on the identifier ring are $\frac{1}{c_x}, \frac{2}{c_x}, \ldots, \frac{c_x-1}{c_x}, \frac{1}{c_x^2}, \frac{2}{c_x^2}, \ldots, \frac{c_x-1}{c_x^2}, \ldots$ of the ring, respectively. Apparently, CAM-Chord becomes Chord if $c_x = 2$, for all node x. We will design a greedy lookup routine for CAM-Chord and a multicast routine that essentially embeds implicit, capacity-constrained, balanced multicast trees in CAM-Chord. The multicast messages are disseminated via these implicit trees. It is a challenging problem to analyze the performance of CAM-Chord. The original analysis of Chord cannot be applied here because c_x is variable. We will provide a new set of analysis on the expected performance of the lookup routine and the multicast routine.

The second system is CAM-Koorde, which differs from Koorde in both variable number of neighbors and how the neighbors are calculated. This difference is critical in constructing balanced multicast trees. Each node x has c_x neighbors. The neighbor identifiers are derived by shifting x to the *right* for a variable number l of bits and then replacing the *left-most* l bits of x with a certain value. In comparison, Koorde shifts x one digit (base m) to the *left* and replaces the *right-most* digit. This subtle difference makes sure that CAM-Koorde spreads neighbors of a node evenly on the identifier ring while neighbors in Koorde tend to cluster together. We will design a lookup routine and a multicast routine that essentially performs broadcast. Remarkably, we show that this broadcast-based routine achieves roughly balanced multicast trees with the expected number of hops to a receiver being $O(\log n / E(\log c_x))$.

CAM-Chord maintains a larger number of neighbors than CAM-Koorde (by a factor of $O(\frac{\log n}{\log c_x})$), which means larger maintenance overhead. On the other hand, CAM-Chord is more robust and flexible because it offers backup paths in its topology [30]. The two systems achieve their best performances under different conditions. Our simulations show that CAM-Chord is a better choice when the node capacities are small and CAM-Koorde is better when the node capacities are large.

3 CAM-Chord Approach

CAM-Chord is an extension of Chord. It takes the capacity of each individual node into consideration. We first describe CAM-Chord as a regular P2P network that supports a lookup routine, which is to find $\widehat{k}$ for a given identifier k. We then present our multicast algorithm on top of CAM-Chord.

When a node joins or leaves the overlay topology, the lookup routine is needed to maintain the topology as it is defined. CAM-Chord is not designed for data sharing among peers as most other P2P networks (e.g., Chord [28]) do. There are NO data items associated with the identifier space. Each multicast group forms its own CAM-Chord network whose sole purpose is to provide an infrastructure for *dynamic capacity-aware* multicasting.

3.1 Neighbors

For a node x in Chord, its neighbor identifiers are $(x + 2^i) \bmod N$, $\forall i \in [1..\log_2 N]$, which are $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \ldots$, of the ring away from x. CAM-Chord is a base-c_x Chord with variable c_x for different nodes. Let c_x^i mean $(c_x)^i$. The neighbor identifiers are $(x + j \times c_x^i) \bmod N$, denoted as $x_{i,j}$, $\forall j \in [1..c_x - 1]$, $\forall i \in [0..\frac{\log N}{\log c_x} - 1]$. i and j are called the *level* and the *sequence number* of $x_{i,j}$. Let $x_{0,0} = x$. The actual neighbors are $\widehat{x_{i,j}}$, which are the nodes responsible for $x_{i,j}$. Note that $\widehat{x_{0,1}}$ is always successor(x).

See an illustration in Fig. 1 with $c_x = 3$. The level-one neighbors in CAM-Chord divide the whole ring into c_x segments of similar size. The level-two neighbors further divides a close segment $(x, x + \frac{1}{c_x} N]$ into c_x subsegments of similar size. And, so on.

Consider an arbitrary identifier k. Let

$$i = \left\lfloor \frac{\log (k - x)}{\log c_x} \right\rfloor, \tag{1}$$

$$j = \left\lfloor \frac{k - x}{c_x^i} \right\rfloor. \tag{2}$$

It can be easily verified that $x_{i,j}$ is the neighbor identifier of x that is counter-clockwise closest to k, which means $\widehat{x_{i,j}}$ is the neighbor node of x that is counter-clockwise closest to node $\widehat{k}$.[2] We call i the *level* and j the *sequence number* of k with respect to x.

3.2 Lookup Routine

CAM-Chord requires a lookup routine to assist member join/departure during a dynamic multicast session. This routine returns the address of node $\widehat{k}$ responsible for a given identifier k. $x.foo()$ denotes a procedure call to be executed at x. It is a local (or remote) procedure call if x is the local (or a remote) node. The set of identifiers that x is responsible for is $(\text{predecessor}(x),\ x]$. The set of identifiers that successor(x) is responsible for is $(x,\ \text{successor}(x)]$.

x.LOOKUP(k)
1. **if** $k \in (x,\ successor(x)]$ **then**
2. **return** the address of $successor(x)$
3. **else**
4. $i \leftarrow \left\lfloor \frac{\log (k-x)}{\log c_x} \right\rfloor$
5. $j \leftarrow \left\lfloor \frac{k-x}{c_x^i} \right\rfloor$
6. **if** $k \in (x,\ \widehat{x_{i,j}}]$ **then**
7. **return** the address of $\widehat{x_{i,j}}$
8. **else**
/* forward request to $x_{i,j}$ */
9. **return** $\widehat{x_{i,j}}$.LOOKUP(k)

First, the LOOKUP routine checks if k is between x and successor(x). If so, LOOKUP returns the address of successor(x). Otherwise, it calculates the level i and the sequence number j of k. If k falls between x and $\widehat{x_{i,j}}$, which means $\widehat{x_{i,j}}$ is responsible for the identifier k, LOOKUP returns the address of $\widehat{x_{i,j}}$. On the other hand, if $\widehat{x_{i,j}}$ precedes k, then x forwards the lookup request to $\widehat{x_{i,j}}$. Because $\widehat{x_{i,j}}$ is x's closest neighbor preceding k, CAM-Chord makes a greedy progress to move the request closer to k.

3.3 Topology Maintenance

Because CAM-Chord is an extension of Chord, we use the same Chord protocols to handle member join/departure and to maintain the correct set of neighbors at each node. The difference is that our LOOKUP routine replaces the Chord LOOKUP routine. The details of the protocols can be found in [28].

The join operation of Chord can be optimized because two consecutive nodes on the ring are likely to have similar neighbors. When a new node joins, it first performs a lookup to find its successor and retrieves its successor's neighbors (called fingers in Chord). It then checks those neighbors to make corrections if necessary. In a base-c Chord, the join complexity without the optimization is $O(c\frac{\log^2 n}{\log^2 c})$ for a *constant* c. The optimization reduces the complexity to $O(c\frac{\log n}{\log c})$.

CAM-Chord can be regarded as a base-c Chord where c is a *random variable* following the node-capacity distribution. It cannot always perform the above optimization because consecutive nodes may have different capacities, which make their neighbor sets different. When this happens, a new node x has to perform $O(c_x \frac{\log n}{\log c_x})$ lookups to find all its neighbors. The lookup complexity is $O(-\frac{\log n}{\log E(\frac{\log c}{c})})$ by Theorem 1 (to be proved). The join complexity is therefore $O(c_x \frac{\log^2 n}{-\log c_x \log E(\frac{\log c}{c})})$, which would be reduced to $O(c\frac{\log^2 n}{\log^2 c})$ if the capacities of all nodes had the same value, i.e., c was a constant. This overhead is too high for a traditional P2P file-sharing application such as FastTrack, because the observations in [31] showed that over 20 percent of the connections last 1 minute or less and 60 percent of the IP addresses keep active for no more than 10 minutes each time after they join the system. But, CAM-Chord is not designed for file-sharing applications. One appropriate CAM-Chord application is teleconferencing, which has far less participants than FastTrack and less dynamic member changes. We do not expect the majority of participants to keep joining and departing during a conference call. Another application is distributed games, where a user is more likely to play for an hour than for one minute.

CAM-Chord makes a tradeoff between capacity awareness and maintenance overhead, which makes it unsuitable for highly dynamic multicast groups. For them, CAM-Koorde is a better choice because a node only has c_x neighbors. Our future research will attempt to develop new techniques to overcome this limitation of CAM-Chord.

3.4 Multicast Routine

On top of the CAM-Chord overlay, we want to implicitly embed a dynamic, roughly balanced multicast tree for each source node. Each intermediate node in the tree should not have more children than its capacity. It should be emphasized that no explicit tree is built. Given a multicast message, a node x executes a MULTICAST routine, sending the message to c_x selected neighbors, which, in turn, execute the MULTICAST routine to further propagate the message. The execution of the MULTICAST routine at different nodes makes sure that the message follows a capacity-aware multicast tree to reach every member.

Let msg be a multicast message, k be an identifier, and x be a node executing the MULTICAST routine. The goal of x.MULTICAST(msg, k) is to deliver msg to all nodes whose identifiers belong to $(x, k]$. The routine is implemented as follows: x chooses c_x neighbors that split $(x, k]$ into c_x subsegments as even as possible. Each subsegment begins from a chosen neighbor and ends at the next clockwise chosen neighbor. x forwards the multicast message to each chosen neighbor, together with the subsegment assigned to this neighbor. When a neighbor receives the message and its subsegment, it forwards the message using the same method. The above process repeats until the size of the subsegment is reduced to one. The distributed computation of MULTICAST recursively divides $(x, k]$ into nonoverlapping subsegments and, hence, no node will receive the multicast message more than once.

x.MULTICAST(msg, k)
1. **if** $k = x$ **then**
2. **return**
3. **else**
4. $i \leftarrow \lfloor \frac{\log k-x}{\log c_x} \rfloor$

2. It is possible that $\widehat{x_{i,j}} = \widehat{k}$.

5. $j \leftarrow \lfloor \frac{k-x}{c_x^i} \rfloor$
   /*select children from level-i neighbors preceding k*/
6. $k' \leftarrow k$
7. **for** $m = j$ down to 1
8. $\widehat{x_{i,m}}$.MULTICAST(msg, k')
9. $k' \leftarrow x_{i,m} - 1$
   /* select $(c_x - j - 1)$ children from level-$(i-1)$ neighbors */
10. $l \leftarrow c_x$
11. **for** $m = c_x - j - 1$ **down to** 1
12. $l \leftarrow l - \frac{c_x}{c_x - j}$ /* for even separation */
13. $\widehat{x_{i-1,[l]}}$.MULTICAST(msg, k')
14. $k' \leftarrow x_{i-1,[l]} - 1$
    /* select x's successor */
15. $\widehat{x_{0,1}}$.MULTICAST(msg, k')

To split $(x, k]$ evenly, x first calculates the level i and the sequence number j of k with respect to x (Lines 4-5). Then, neighbors $\widehat{x_{i,m}}$ $(\forall m \in [1..j])$ at the ith level preceding k are selected as children of x in the multicast tree (Lines 6-9). We also select x's successor, which is $\widehat{x_{0,1}}$ (Line 15). Since $j+1$ may be less than c_x, in order to fully use x's capacity, $c_x - 1 - j$ neighbors at the $(i-1)$th level are chosen; Lines 10-14 ensure that the selection is evenly spread at the $(i-1)$th level. Because the algorithm selects neighbors that divide $(x, k]$ as even as possible, it constructs a multicast tree that is roughly balanced. At Line 9, we optimize the code by using $k \leftarrow x_{i,m} - 1$ instead of $k \leftarrow \widehat{x_{i,m}} - 1$. That is because there is no node in $(x_{i,m}, \widehat{x_{i,m}})$ by the definition of $\widehat{x_{i,m}}$.

3.5 Analysis

Assume $c_x \geq 2$ for every node x. We analyze the performance of the LOOKUP routine and the multicast routine of CAM-Chord. Suppose a node x receives a lookup request for identifier k and it forwards the request to a neighbor node $\widehat{x_{i,j}}$ that is closest to k. We call $\frac{k-\widehat{x_{i,j}}}{k-x}$ the *distance reduction ratio*, which measures how much closer the request is from k after one-hop routing. The following lemma establishes an upper bound on the distance reduction ratio with respect to c_x, which is a random variable of certain distribution.

Lemma 1. *Suppose a node x forwards a lookup request for identifier k to a neighbor $\widehat{x_{i,j}}$. If $\widehat{x_{i,j}} \in (x, k]$, then*

$$E\left(\frac{k-\widehat{x_{i,j}}}{k-x}\right) < E\left(\frac{\ln c_x}{c_x - 1}\right).$$

Proof. Based on the algorithm of the LOOKUP routine, i must be the height of k with respect to x. By (1) and (2), k can be written as

$$k = x + jc_x^i + l,$$

where $j \in [1..c_x - 1]$ is the sequence number of k with respect to x and $l \in [0..c_x^i)$.

$$k - x = jc_x^i + l. \tag{3}$$

By definition (Section 3.1), $x_{i,j} = x + jc_x^i$. Because x, $x_{i,j}$, $\widehat{x_{i,j}}$, and k are in clockwise order on the identifier ring, we have

$$k - \widehat{x_{i,j}} \leq k - x_{i,j} = l. \tag{4}$$

By (3) and (4), we have

$$\frac{k-\widehat{x_{i,j}}}{k-x} \leq \frac{l}{jc_x^i + l} < \frac{c_x^i}{jc_x^i + l}.$$

We now derive the expected distance reduction ratio. $E(\frac{k-\widehat{x_{i,j}}}{k-x})$ depends on three random variables, j, l, and c_x. Because the location of k is arbitrary with respect to x, we can consider j and l as independent random variables with uniform distributions on their respective value ranges.

$$\begin{aligned}
E\left(\frac{k'-\widehat{x_{i,m}}}{k-x}\right) &= E\left(\frac{1}{c_x-1}\Sigma_{j=1}^{c_x-1}\frac{1}{c_x^i}\int_0^{c_x^i}\frac{k'-\widehat{x_{i,j}}}{k-x}dl\right) \\
&= E\left(\frac{1}{c_x-1}\Sigma_{j=1}^{c_x-1}\frac{1}{c_x^i}\int_0^{c_x^i}\frac{c_i^x}{jc_x^i+l}dl\right) \\
&= E\left(\frac{1}{c_x-1}(\Sigma_{j=1}^{c_x-1}(\ln(j+1)-\ln j))\right) \\
&= E\left(\frac{\ln c_x}{c_x-1}\right).
\end{aligned}$$

□

Theorem 1. *Let c_x, for all nodes x, be independent random variables of certain distribution. The expected length of a lookup path in CAM-Chord is $O(-\frac{\ln n}{\ln E(\frac{\ln c_x}{c_x})})$.*

Proof. Suppose $(x_1, x_2, \ldots, x_m)$ is a prefix of a lookup path for identifier k, where x_1 is the node that initiates the lookup, and x_i, $i \in [1..m]$, and k are in clockwise order on the identifier ring. Because the nodes are randomly mapped to the identifier ring by a hash function, the distance reduction ratio after each hop is independent of those after other hops. Consequently, $\frac{k-x_i}{k-x_{i-1}}, i \in [2..m]$, are independent random variables.

$$\begin{aligned}
E(k-x_m) &= E\left(\frac{k-x_m}{k-x_{m-1}}\cdot\frac{k-x_{m-1}}{k-x_{m-2}}\cdot\ldots\cdot\frac{k-x_2}{k-x_1}\cdot(k-x_1)\right) \\
&= E\left(\frac{k-x_m}{k-x_{m-1}}\right)\cdot E\left(\frac{k-x_{m-1}}{k-x_{m-2}}\right)\cdot\ldots\cdot E\left(\frac{k-x_2}{k-x_1}\right) \\
&\quad \cdot E(k-x_1) \\
&< \left(E\left(\frac{\ln c_x}{c_x-1}\right)\right)^{m-1}\cdot N,
\end{aligned} \tag{5}$$

where c_x is a random variable with the same distribution as $c_{x_i}, i \in [1..m]$. Next, we derive the value of m that ensures $E(k-x_m) < \frac{N}{n}$, which is the average distance between two adjacent nodes on the identifier ring. The following is a sufficient condition to achieve $E(k-x_m) < \frac{N}{n}$.

$$\left(E\left(\frac{\ln c_x}{c_x-1}\right)\right)^{m-1}\cdot N = \frac{N}{n}$$

$$m = -\frac{\ln n}{\ln E(\frac{\ln c_x}{c_x-1})}.$$

If $E(k-x_m) < \frac{N}{n}$, the expected number of additional routing hops from x_m to k is $O(1)$. Therefore, $O(m) = O(-\frac{\ln n}{\ln E(\frac{\ln c_x}{c_x})})$ gives the expected length of the lookup path.□

It is natural that the expected length of a lookup path in CAM-Chord depends on the probability distribution of c_x, which affects the topological structure of the overlay network. For a given distribution, an upper bound of the expected path length can be derived from Theorem 1. The following theorem gives an example.

Theorem 2. *Suppose the node capacity c_x follows a uniform distribution with $E(c_x) = c$. The expected length of a lookup path in CAM-Chord is $O(\frac{\ln n}{\ln c})$.*

Proof. Suppose the range of c_x is $[t_1..t_2]$ with $E(c_x) = c$. We perform Big-O reduction as follows:

$$\begin{aligned} E\left(\frac{\ln c_x}{c_x}\right) &= \Theta\left(\sum_{c_x=t_1}^{t_2} \frac{\ln c_x}{c_x} \times \frac{1}{t_2 - t_1 + 1}\right) \\ &= \Theta\left(\int_{t_1}^{t_2} \frac{\ln c_x}{c_x} dc_x \times \frac{1}{t_2 - t_1 + 1}\right) \\ &= \Theta\left(\left(\frac{1}{2}\ln^2 t_2 - \frac{1}{2}\ln^2 t_1\right) \times \frac{1}{c}\right) \\ &= \Theta\left(\frac{\ln^2 c}{c}\right) \text{ because } t_2 \le 2c \text{ and } t_1 \le c. \end{aligned}$$

Therefore,

$$\ln E\left(\frac{\ln c_x}{c_x}\right) = \Theta\left(\ln \frac{\ln^2 c}{c}\right) = \Theta(-\ln c).$$

By Theorem 1, $O(-\frac{\ln n}{\ln E(\frac{\ln c_x}{c_x})}) = O(\frac{\ln n}{\ln c})$. □

Other distributions of c_x may be analyzed similarly.

Next, we analyze the performance of the MULTICAST routine in CAM-Chord. Suppose x executes

$$x.\text{MULTICAST}(msg, k),$$

which is responsible for delivering msg to all nodes in the identifier segment (x, k). Specifically, x forwards msg to some neighbor nodes $\widehat{x_{i,m}}$ by remotely invoking $\widehat{x_{i,m}}.\text{MULTICAST}(msg, k')$, which is responsible for delivering msg to a smaller subsegment $(\widehat{x_{i,m}}, k')$, where $\widehat{x_{i,m}}, k' \in (x, k)$. It is a typical divide-and-conquer strategy. We call $\frac{k' - \widehat{x_{i,m}}}{k-x}$ the *segment reduction ratio*, which measures the degree of reduction in problem size after one-hop multicast routing. The following lemma establishes an upper bound on the segment reduction ratio with respect to c_x, which is a random variable of certain distribution.

Lemma 2. *Suppose a node x forwards a multicast message to a neighbor $\widehat{x_{i,m}}$, i.e., $x.\text{MULTICAST}(msg, k)$ calls $\widehat{x_{i,m}}.\text{MULTICAST}(msg, k')$. It must be true that*

$$E\left(\frac{k' - \widehat{x_{i,m}}}{k-x}\right) < E\left(\frac{\ln c_x}{c_x - 1}\right).$$

Proof. Based on the algorithm of the MULTICAST routine, the execution of $x.\text{MULTICAST}(msg, k)$ will divide its responsible segment (x, k) into c_x subsegments, and $\widehat{x_{i,m}}$ is responsible for delivering msg to all nodes in one subsegment $(\widehat{x_{i,m}}, k')$. The largest subsegment is created by Lines 6-9. When Line 7 is executed for $m \in [j-1..1]$, $k' = x_{i,m+1} - 1$. Therefore,

$$\begin{aligned} k' - \widehat{x_{i,m}} &< x_{i,m+1} - x_{i,m} \\ &= x + (m+1)c_x^i - x - mc_x^i \\ &= c_x^i. \end{aligned} \quad (6)$$

By Line 4, i is the level of k with respect to x. By (1) and (2), k can be written as

$$k = x + jc_x^i + l,$$

where $j \in [1..c_x - 1]$ is the sequence number of k with respect to x and $l \in [0..c_x^i)$.

$$k - x = jc_x^i + l. \quad (7)$$

By (6) and (7), we have

$$\frac{k' - \widehat{x_{i,m}}}{k-x} < \frac{c_i^x}{jc_x^i + l}.$$

We now derive the expected segment reduction ratio. $E(\frac{k' - \widehat{x_{i,m}}}{k-x})$ depends on three random variables, j, l, and c_x. Because the location of k is arbitrary with respect to x, we can consider j and l as independent random variables with uniform distributions on their respective value ranges.

$$\begin{aligned} E\left(\frac{k' - \widehat{x_{i,m}}}{k-x}\right) &= E\left(\frac{1}{c_x - 1}\Sigma_{j=1}^{c_x-1}\frac{1}{c_x^i}\int_0^{c_x^i}\frac{k' - \widehat{x_{i,j}}}{k-x}dl\right) \\ &= E\left(\frac{1}{c_x - 1}\Sigma_{j=1}^{c_x-1}\frac{\ln c_x}{c_x^i}\int_0^{c_x^i}\frac{c_i^x}{jc_x^i + l}dl\right) \\ &= E\left(\frac{1}{c_x - 1}(\Sigma_{j=1}^{c_x-1}(\ln(j+1) - \ln j))\right) \\ &= E\left(\frac{\ln c_x}{c_x - 1}\right). \end{aligned}$$

□

A *multicast path* is defined as the path that the MULTICAST routine takes to deliver a multicast message from a source node to a destination node. The proofs of the following two theorems are very similar to those of Theorems 1 and 2, due to the similarity between Lemma 2 and Lemma 1, on which the theorems are based. To avoid excessive repetition and to conserve space, we omit the proofs for Theorems 3 and 4.

Theorem 3. *Let c_x, for all nodes x, be independent random variables of certain distribution. The expected length of a multicast path in CAM-Chord is $O(-\frac{\ln n}{\ln E(\frac{\ln c_x}{c_x})})$.*

Theorem 4. *Suppose the node capacity c_x follows a uniform distribution and $E(c_x) = c$. The expected length of a multicast path in CAM-Chord is $O(\frac{\ln n}{\ln c})$.*

4 CAM-Koorde Approach

This section proposes CAM-Koorde. For any node x in CAM-Koorde, its number of neighbors is exactly equal to its capacity c_x. The maintenance overhead of CAM-Koorde is

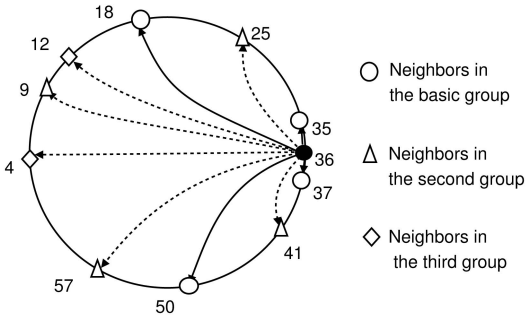

Fig. 2. CAM-Koorde topology with identifier space [0..63].

smaller than that of CAM-Chord due to a smaller number of neighbors.

Like Koorde, CAM-Koorde embeds the Bruijn graph in the identifier ring. On the other hand, it has two major differences from Koorde, which are critical to our capacity-aware multicast service.

- The first difference is about neighbor selection. The neighbor identifiers of a node x in Koorde are derived by shifting x one digit (base m) to the left and then replacing the last digit with 0 through $m-1$. The neighbor identifiers differ only at the last digit. Consequently, they are clustered and often refer to the same physical node. For the purpose of multicast, we want the neighbors to spread evenly on the identifier ring. The neighbor identifiers of a node x in CAM-Koorde are derived by shifting x one or more bits to the right and then replacing the high-order bits with 0 through certain number. The neighbor identifiers differ at the high-order bits and, therefore, they are evenly distributed on the identifier ring.
- The second difference is about the number of neighbors. Koorde requires every node to have the same number of neighbors. CAM-Koorde allows nodes to have different numbers of neighbors.

4.1 Neighbors

Let $N = 2^b$. In CAM-Koorde, x has c_x neighbors, which are categorized into three groups. All computations are assumed to be modulo N.

- The *basic group* has four neighbors. Two are x's predecessor and successor. The other two are the nodes responsible for identifiers $(x/2)$ and $2^{b-1} + (x/2)$, respectively.
- After the basic group, there are $c_x - 4$ remaining neighbors. Let $s = \lfloor \log(c_x - 4) \rfloor$. If $s > 1$, we shall shift x by s bits to the right to derive the neighbor identifiers.[3] If $s > 1$, then let $t = 2^s$; otherwise, let $t = 0$. The neighbors in the second group are the nodes responsible for identifiers $(i \times 2^{b-s} + x/2^s)$, $\forall\ i \in [0..t-1]$.
- After the basic and second groups, there are $t' = c_x - 4 - t$ remaining neighbors. Let $s' = s + 1$. The neighbors in the third group are the nodes responsible for identifiers $(i \times 2^{b-s'} + x/2^{s'}), \forall i \in [0..t'-1]$.

3. If $s = 1$, it means to shift one bit. The basic group already does that.

It is required that $c_x \geq 4$. The basic group is mandatory. The optional second and third groups pick up the remaining neighbors.

An example is given in Fig. 2, showing the neighbors of node 36 (100100) whose capacity is 10. The binary representation of the node identifier is given in the parentheses. The basic group is

$$\{35\ (100011),\ 37\ (100101),\ 18\ (010010),\ 50\ (110010)\}.$$

The second group is

$$\{9\ (001001),\ 25\ (011001), 41\ (101001), 57\ (111001)\}.$$

The third group is

$$\{4\ (000100), 12\ (001100)\}.$$

4.2 Lookup Routine

Definition 1. *Given two b-bit identifiers x and k, if an l-bit prefix of x matches an l-bit suffix of k, we say x and k share l ps-common bits. $x = k$ if the two share b ps-common bits.*

Similar to CAM-Chord, a lookup routine is needed in CAM-Koorde for member join/departure. First consider an N-node network with every identifier having a corresponding node. Given an identifier k, suppose node x wants to query for the address of node k. The lookup routine forwards the lookup request along a chain of neighbors whose identifiers share progressively more ps-common bits with k. Starting from x, we identify a neighbor that has the longest prefix matching the suffix of k. More specifically, if the third group is not empty and a third-group neighbor can be derived by selecting the $(\lfloor \log(c_x - 4) \rfloor + 1)$ bits of k that precedes the current ps-common bits and shifting them from the left into x, then the lookup request is forwarded to this neighbor. Otherwise, if the second group is not empty and a second-group neighbor can be derived by selecting the $\lfloor \log(c_x - 4) \rfloor$ bits of k that precedes the current ps-common bits and shifting them from the left into x, then the lookup request is forwarded to this neighbor. Otherwise, we forward the request to a first-group neighbor that increases the number of ps-common bits by one. To determine each subsequent node on the forwarding path, a similar process repeats by shifting more bits of k into the identifier of the next hop receiver. After at most b hops, the request can reach node k.

Now, suppose $n \ll N$, which is normally the case. We still calculate the chain of neighbor identifiers in the above way, which essentially transforms identifier x to identifier k in a series of steps, each step adding one or more bits from k. Once the next neighbor identifier y on the chain is calculated, the request is forwarded to $\widehat{y}$, which, in turn, calculates its neighbor identifier that should be the next on the forwarding path and then forwards the request.

The pseudocode of the LOOKUP routine is shown below. It uses the high-order bits of the node identifier to match the low-order bits of k, which is different from Koorde's routine and is critical for our multicast routine, to be discussed shortly.

x.LOOKUP(k)
1. **if** $k \in (\text{predecessor}(x), x]$ **then**
2. **return** the address of x
3. **if** $k \in (x, \text{successor}(x)]$ **then**
4. **return** the address of $\text{successor}(x)$

5. m_1 be the number of ps-common bits shared by x and k
6. find the neighbor y that shares the largest number m_2 of ps-common bits with k
7. **if** $m_1 \leq m_2$ **then**
8. **return** y.LOOKUP(k)
9. **else**
10. **if** $|k - \text{predecessor}(x)| < |k - \text{successor}(x)|$ **then**
11. **return** $\text{predecessor}(x)$.LOOKUP(k)
12. **else**
13. **return** $\text{successor}(x)$.LOOKUP(k)

Koorde uses Chord's protocols with a new LOOKUP routine for node join/departure, so does CAM-Koorde.

4.3 Multicast Routine

When a node receives a multicast message, it forwards the message to all neighbors except those that have received or are receiving the message. Because neighbor connections are bidirectional, it is easy for a node to perform the checking through a short control packet. The overhead is negligible when the message is large, e.g., a video file. Note that a node does not have to wait for the entire message to arrive before forwarding it to neighbors. The forwarding is done on per packet basis, but the checking is performed only for the first packet of a message, which carries the message header. The pseudocode of the MULTICAST routine is shown below:

x.MULTICAST(msg)
1. **for** each neighbor y **do**
2. **if** y has not received or is not receiving msg **then**
3. y.MULTICAST(msg)

4.4 Analysis

Lemma 3. *Let c_x, for all nodes x, be independent random variables of certain distribution. The expected length of the shortest path between two nodes in CAM-Koorde is $O(\log n/E(\log c_x))$.*

Proof. Consider two arbitrary nodes x_0 and y. Let y' be an $O(\log n)$-bit prefix of y. We show there exists a path of length $O(\log n/E(\log c_x))$ that reaches a node $\widehat{x_m}$ with y' also as its prefix.

- We construct a physical path (x_0, $\widehat{x_1}$, $\widehat{x_2}$, …, $\widehat{x_{m-1}}$, $\widehat{x_m}$) as follows: Node x_0 has a basic or second-group neighbor $\widehat{x_1}$, where x_1 is derived by shifting $\max\{1, \lfloor\log(c_{x_0} - 4)\rfloor\}$ low-order bits of y' into x_0 from the left.[4] The bits of y' that have been used in shifting are called *used bits*. Similarly, $\widehat{x_1}$ has a second-group neighbor $\widehat{x_2}$, where x_2 is derived by shifting $\max\{1, \lfloor\log(c_{\widehat{x_1}} - 4)\rfloor\}$ low-order unused bits of y' into $\widehat{x_1}$ …. Finally, $\widehat{x_{m-1}}$ has a second-group neighbor $\widehat{x_m}$, where x_m is derived by shifting the remaining $\max\{1, \lfloor\log(c_{\widehat{x_{m-1}}} - 4)\rfloor\}$ low-order unused bits of y' into $\widehat{x_{m-1}}$. The length of path (x_0, $\widehat{x_1}$, $\widehat{x_2}$, …, $\widehat{x_{m-1}}$) is m. The total number of used bits of y' is $\sum_{i=0}^{m-1} \max\{1, \lfloor\log(c_{\widehat{x_i}} - 4)\rfloor\}$, which is $O(\log n)$. Let c_x be a random variable of the same distribution as $c_{\widehat{x_i}}$, $\forall i \in [0..m-1]$.

$$\sum_{i=0}^{m-1} \max\{1, \lfloor\log(c_{\widehat{x_i}} - 4)\rfloor\} = O(\log n)$$

$$\sum_{i=0}^{m-1} \log c_{\widehat{x_i}} = O(\log n)$$

$$E\left(\sum_{i=0}^{m-1} \log c_{\widehat{x_i}}\right) = O(\log n)$$

$$mE(\log c_x) = O(\log n)$$

$$m = O(\log n/E(\log c_x)).$$

- Next, we construct an identifier list (x_0, x'_1, x'_2, …, x'_{m-1}, x'_m) as follows: x'_1 is derived by shifting $\max\{1, \lfloor\log(c_{x_0} - 4)\rfloor\}$ low-order bits of y' into x_0 from the left. x'_2 is derived by shifting $\max\{1, \lfloor\log(c_{\widehat{x_1}} - 4)\rfloor\}$ low-order unused bits of y' into x'_1 …. Finally, where x'_m is derived by shifting the remaining $\max\{1, \lfloor\log(c_{\widehat{x_{m-1}}} - 4)\rfloor\}$ low-order unused bits of y' into x'_{m-1}.

y' is an $O(\log n)$-bit prefix of both x'_m and y. Therefore, the distance between x'_m and y on the identifier ring, $|x'_m - y|$, is $O(\frac{N}{n})$. Note that $\frac{N}{n}$ is the average distance between any two adjacent nodes on the ring. If we can show that $E(|\widehat{x_m} - x'_m|) < \frac{N}{n}$, then

$$E(|\widehat{x_m} - y|) \leq E(|\widehat{x_m} - x'_m| + |x'_m - y|)$$
$$= E(|\widehat{x_m} - x'_m|) + E(|x'_m - y|) = O\left(\frac{N}{n}\right),$$

which means that the expected number of hops from $\widehat{x_m}$ to y is $O(1)$.

$\forall i \in [1..m]$, define a random variable $\Delta_i = |\widehat{x_i} - x_i|$. Because x_i can be anywhere in (predecessor($\widehat{x_i}$), $\widehat{x_i}$), we have

$$E(\Delta_i) = \frac{1}{2}E(|\text{predecessor}(\widehat{x_i}), \widehat{x_i}|) = \frac{N}{2n}, \tag{9}$$

where $\frac{N}{n}$ is the expected distance between two adjacent nodes on the identifier ring.

x_i and x'_i are derived from $\widehat{x_{i-1}}$ and x'_{i-1}, respectively, by shifting the same $\max\{1, \lfloor\log(c_{\widehat{x_{i-1}}} - 4)\rfloor\}$ bits of y' in from the left. Therefore,

$$|x_i - x'_i| = \frac{|\widehat{x_{i-1}} - x'_{i-1}|}{2^{\max\{1, \lfloor\log(c_{\widehat{x_{i-1}}} - 4)\rfloor\}}}.$$

It follows that, $\forall i \in [i..m]$,

$$|\widehat{x_i} - x'_i| \leq |\widehat{x_i} - x_i| + |x_i - x'_i|$$
$$= \Delta_i + \frac{|\widehat{x_{i-1}} - x'_{i-1}|}{2^{\max\{1, \lfloor\log(c_{\widehat{x_{i-1}}} - 4)\rfloor\}}}.$$

By induction, we have

$$|\widehat{x_m} - x'_m| \leq \sum_{i=1}^{m} \Delta_i \prod_{j=i}^{m-1} \frac{1}{2^{\max\{1, \lfloor\log(c_{\widehat{x_j}} - 4)\rfloor\}}}.$$

4. If $c_x < 6$, we can pick x_1 from the basic group, which shifts one bit of y' into x_0; if $c_x \geq 6$, we can pick x_1 from the second group, which shifts $\lfloor\log(c_{x_0} - 4)\rfloor$ bits of y' into x_0.

Because $c_x \geq 4$ for any node x in CAM-Chord, $\max\{1, \lfloor \log(c_{\widehat{x_j}} - 4) \rfloor\} \geq 1$. Hence,

$$|\widehat{x_m} - x'_m| \leq \sum_{i=1}^{m} \Delta_i \left(\frac{1}{2}\right)^{m-i}$$

$$E(|\widehat{x_m} - x'_m|) \leq \sum_{i=1}^{m} E(\Delta_i) \left(\frac{1}{2}\right)^{m-i} = \frac{N}{2n} \sum_{i=1}^{m} \left(\frac{1}{2}\right)^{m-i} = \frac{N}{2n} \sum_{i=0}^{m-1} \left(\frac{1}{2}\right)^{i} < \frac{N}{n}.$$

Consequently, the expected number of hops from $\widehat{x_m}$ to x'_m and then to y is $O(1)$. The expected length of the entire path from x_0 to y is $O(m) = O(\log n / E(\log c_x))$. □

Theorem 5. *Let c_x, for all nodes x, be independent random variables of certain distribution. The expected length of a multicast path in CAM-Koorde from a source node to a member node is $O(\log n / E(\log c_x))$.*

Proof. According to the MULTICAST routine, a multicast packet is delivered in CAM-Koorde by broadcast, which follows the shortest paths to the member nodes. Therefore, the expected length of a multicast path from a source node to a member node is $O(\log n / E(\log c_x))$ by Lemma 3. □

Theorem 6. *Suppose the node capacity c_x follows a uniform distribution and $E(c_x) = c$. The expected length of a multicast path in CAM-Koorde from a source node to a member node is $O(\log n / \log c))$.*

Proof. Suppose the range of c_x is $[t_1..t_2]$ with $E(c_x) = c$. We perform Big-O reduction as follows:

$$E(\log c_x) = \frac{1}{t_2 - t_1 + 1} \sum_{c_x = t_1}^{t_2} \log c_x = \Theta\left(\frac{1}{t_2 - t_1 + 1} \int_{t_1}^{t_2} \log c_x dc_x\right) = \Theta\left(\frac{1}{t_2 - t_1 + 1} ((t_2 \log t_2 - t_2) - (t_1 \log t_1 - t_1))\right) = \Theta(\log c) \text{ because } t_2 \leq 2c \text{ and } t_1 \leq c.$$

By Theorem 5, $O(\log n / E(\log c_x)) = O(\frac{\log n}{\log c})$. □

5 Discussions

5.1 Group Members with Very Small Upload Bandwidth

A node x with very small upload bandwidth should only be a leaf in the multicast trees unless itself is the data source. In order to make sure that the MULTICAST routine does not select x as an intermediate node in any multicast tree, x must not be in the CAM-Chord (or CAM-Koorde) overlay network. Instead, it joins as an external member. x asks a node y known to be in CAM-Chord (or CAM-Koorde) to perform LOOKUP(k) for an arbitrary identifier k, which returns a random node z in the overlay network. 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 CAM-Chord (or CAM-Koorde).

5.2 Proximity and Geography

The overlay connections between neighbors may have very different delays. Two neighbors may be separated by transcontinental links, or they may be on the same LAN. There exist some approaches to cope with geography, for example, *Proximity Neighbor Selection* and *Geographic Layout*. With Proximity Neighbor Selection, nodes are given some freedom in choosing neighbors based on other criteria (i.e., latencies), in addition to the arithmetic relations between their identifiers. With Geographic Layout, node identifiers are chosen in a geographically informed manner. The main idea is to make geographically closeby nodes form clusters in the overlay. Readers are referred to [32], [30] for details.

In extension to the existing P2P networks, CAMs can naturally inherit most of those features without much additional effort. For example, instead of choosing the ith neighbor to be $(x + 2^i)$, a proximity optimization of Chord allows the ith neighbor to be any node within the range of $[(a + 2^i),\ (a + 2^{i+1})$, which does not affect the complexities [30]. This optimization can also be applied to CAM-Chord (which is an extension of Chord) without affecting the complexities. A node x can choose any node whose identifier belongs to the segment $[x + j \times c_x^i, x + (j + 1) \times c_x^i)$ as the neighbor $x_{i,j}$. Given this freedom, some heuristics (e.g., smallest delay first) may be used to choose neighbors to promote proximity-clustering. Specifically, a node x can progressively scan the nodes in the allowed segment for neighbor $x_{i,j}$, for example, by following the successor link to probe the next node in the segment after every 100k data bits sent by x, which trivializes the probing overhead. x always use the nearest node it has found recently as its neighbor.

6 Simulation

Throughput and latency are two major performance metrics for a multicast application. We evaluate the performance of CAMs from these two aspects. We simulate multicast algorithms on top of CAM-Chord, Chord, CAM-Koorde, and Koorde, respectively. The identifier space is $[0, 2^{19})$. If not specified otherwise, the default number of nodes in an overlay network is $100,000$ and the node capacities are taken from $[4..10]$ with uniform probability. The upload bandwidths of the nodes are randomly distributed in a default range of $[400, 1,000]$ kbps.

It should be noted that the value ranges may go far beyond the default ones in specific simulations. In our simulations, $c_x = \lfloor B_x / p \rfloor$, where B_x is the node's upload bandwidth and p is a system parameter of CAMs, specifying the desired bandwidth per link in the multicast tree. By varying the value of p, we can construct CAMs with different average node capacities, which also mean different average numbers of children per nonleaf node and, consequently, different tree depths (latency). If the average

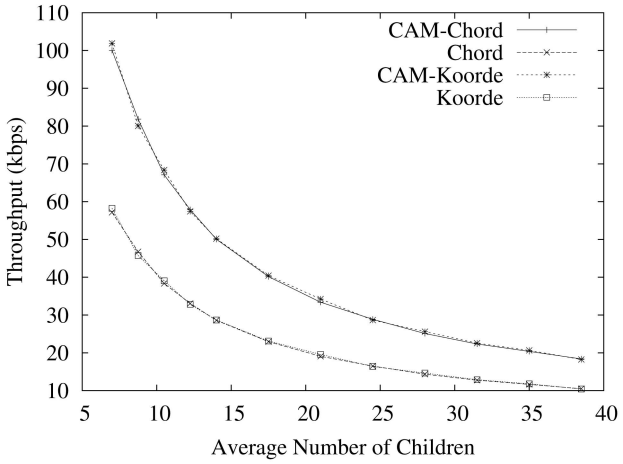

Fig. 3. Multicast throughput with respect to average number of children per nonleaf node.

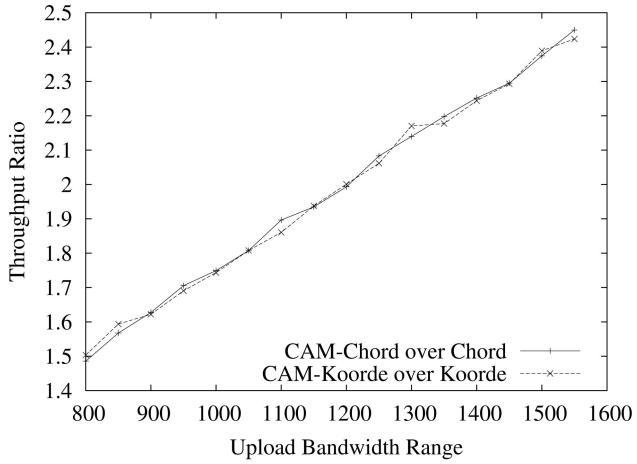

Fig. 4. Throughput improvement ratio with respect to upload bandwidth range.

node capacity c is not the default value of 7, the node capacities are taken uniformly from $[4, 2c-4]$.

6.1 Throughput

We compare the sustainable throughput of multicast systems based on CAM-Chord, Chord, CAM-Koorde, and Koorde. Throughput is defined as the rate at which data can be continuously delivered from a source node to all other nodes. Due to limited buffer space at each node, the sustainable multicast throughput is decided by the link with the least allocated bandwidth in the multicast tree. CAM-Chord and CAM-Koorde produce much larger throughput because a node's capacity c_x (which is its number of children in the multicast tree) is adjustable based on the node's upload bandwidth. The primary advantage of CAMs over the Chord/Koorde is their ability to adapt the overlay topology according to host heterogeneity.

Fig. 3 compares the throughput of CAM-Chord, Chord, CAM-Koorde, and Koorde with respect to the average number of children per nonleaf node in the multicast tree. CAMs perform much better. Their throughput improvement over Chord and Koorde is 70-80 percent when the nodes' upload bandwidths are chosen from the rather narrow default range of $[400, 1,000]$ kbps. The larger the upload-bandwidth range, the more the throughput improvement, as demonstrated by Fig. 4. In general, let $[a, b]$ be the range of upload bandwidth. The upper bound b of the range is shown on the horizontal axis of Fig. 4, while the lower bound a is fixed at 400 kbps. The figure shows that the throughput improvement by CAMs increases when the upload-bandwidth range is larger, representing a greater degree of host heterogeneity. The simulation data also indicate that the throughput ratio of CAM-Chord (CAM-Koorde) over Chord (Koorde) is roughly proportional to $\frac{a+b}{2a}$.

Fig. 5 shows the multicast throughput with respect to the size of the multicast group. According to the simulation results, the throughput is largely insensitive to the group size.

6.2 Throughput versus Latency

We measure multicast latency by the average length of multicast paths. Latency is determined by both the number of hops and the hop delay. In CAM-Chord and CAM-Koorde, the overlay links are randomly formed among the nodes due to the use of DHT. Therefore, the latency of an overlay path is statistically proportional to the number of hops. That's why we used the number of hops to characterize the latency performance. In fact, the measurement in terms of number of hops carries information beyond latency. It is also an indication of how many times a message has to be relayed, which is a resource consumption issue. However, with the proximity neighbor selection in Section 5.2, the latency is no longer proportional to the number of hops. We add a simulation in Section 6.4 to study this case, where the actual delay is measured.

Both throughput and latency are functions of average node capacity. With a larger average node capacity (achieved by a smaller value of p), the throughput decreases due to more children per nonleaf node and the latency also decreases due to smaller tree depth. Therefore, there exists a tradeoff between throughput and latency, which is depicted by Fig. 6. Higher throughput can be achieved at the cost of longer routing paths. Given the same upload bandwidth distribution, the system's performance can be tuned by adjusting p. The figure also shows that, for relatively small throughput (less than 46kbps in the figure)—namely, for

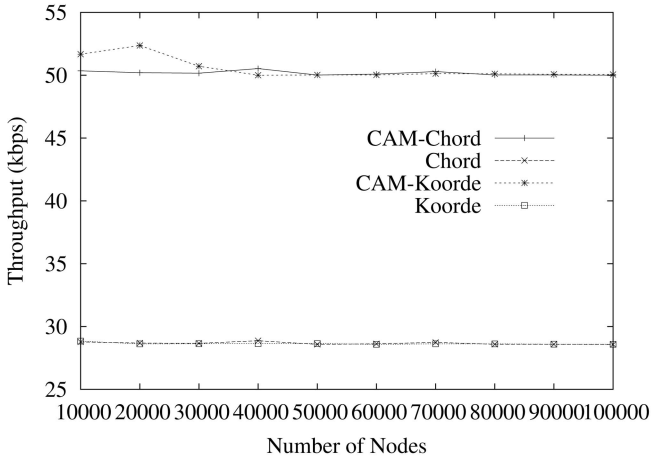

Fig. 5. Multicast throughput with respect to size of the multicast group.

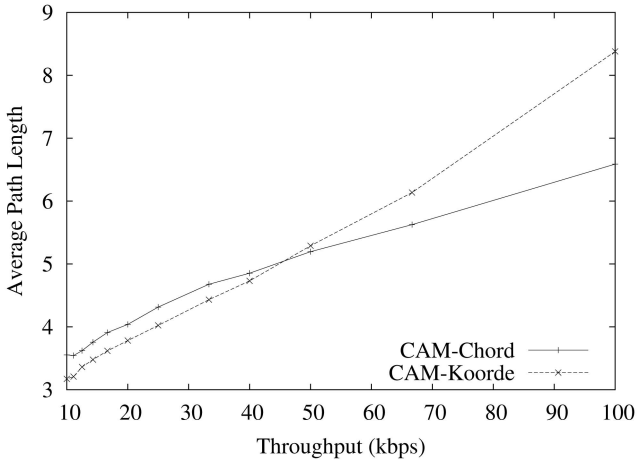

Fig. 6. Throughput versus average path length.

large node capacities—CAM-Koorde slightly outperforms CAM-Chord; for relatively large throughput (greater than 46kbps in the figure)—namely, for small node capacities—CAM-Chord outperforms CAM-Koorde, which will be further explained in Section 6.4.

6.3 Path Length Distribution

Figs. 7 and 8 present the statistical distribution of multicast path lengths, i.e., the number of nodes that can be reached by a multicast tree in certain number of hops. Each curve represents a simulation with node capacities chosen from a different range. When the capacity range increases, the distribution curve moves to the left of the plot due to shorter multicast paths. The improvement in the distribution can be measured by how much the curve is shifted to the left. At the beginning, a small increase in the capacity range causes significant improvement in the distribution. After the capacity range reaches a certain level ($[4, 10]$ in our simulations), the improvement slows down drastically.

Each curve has a single peak, and the right side of the peak quickly decreases to zero. It means that the vast majority of nodes are reached within a small range of path lengths. We did not observe any multicast path whose length was significantly larger than the average path length.

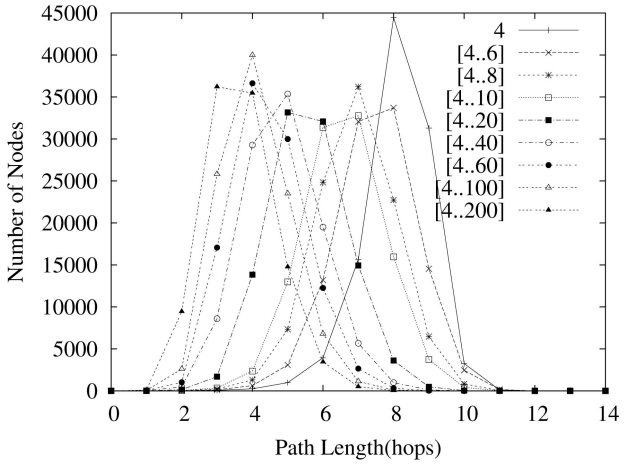

Fig. 7. Path length distribution in CAM-Chord. Legend "$[x..y]$" means the node capacities are uniformly distributed in the range of $[x..y]$.

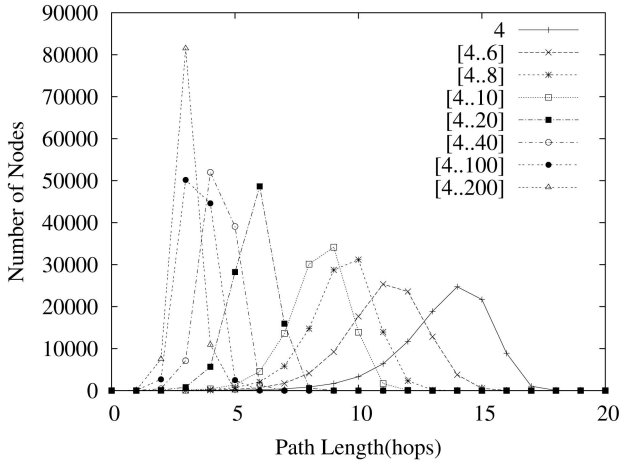

Fig. 8. Path length distribution in CAM-Koorde. Legend "$[x..y]$" means the node capacities are uniformly distributed in the range of $[x..y]$.

6.4 Average Path Length

Fig. 9 shows the average path length with respect to the average node capacity. We also plot an artificial line, $1.5\frac{\ln n}{\ln c}$ with $n = 10^5$, which upper-bounds the average path lengths of CAM-Chord and CAM-Koorde, verifying Theorem 4 and Theorem 6.

From the figure, when the average node capacity is less than 10, the average path length of CAM-Chord is smaller than that of CAM-Koorde; when the average node capacity is greater than 12, the average path length of CAM-Koorde is smaller than CAM-Chord. A smaller average path length means more balanced multicast trees. Therefore, for small node capacities, CAM-Chord multicast trees are more balanced than CAM-Koorde multicast trees, and vice versa. The reasons are explained as follows: On one hand, a nonleaf CAM-Koorde node x may have less children than c_x because some neighbors may have already received the multicast message from a different path. This tends to make the depth of a CAM-Koorde multicast tree larger than that of a CAM-Chord tree. On the other hand, a CAM-Chord node x may split $(x, k]$ into uneven subsegments (i.e., subtrees) with a ratio up to c_x (Lines 6-15 in Section 3.4). This ratio of unevenness becomes small when the node

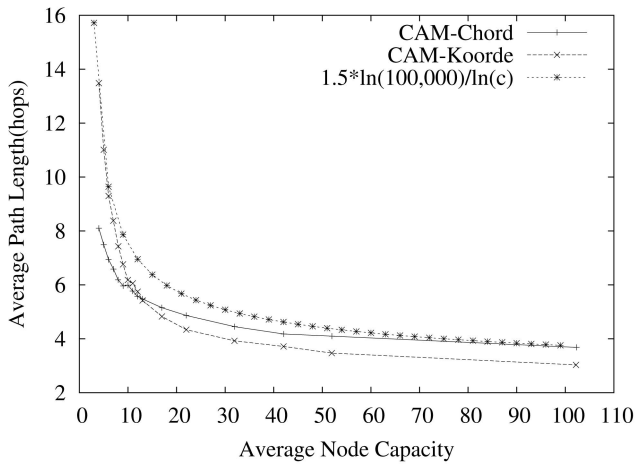

Fig. 9. Average path length with respect to average node capacity.

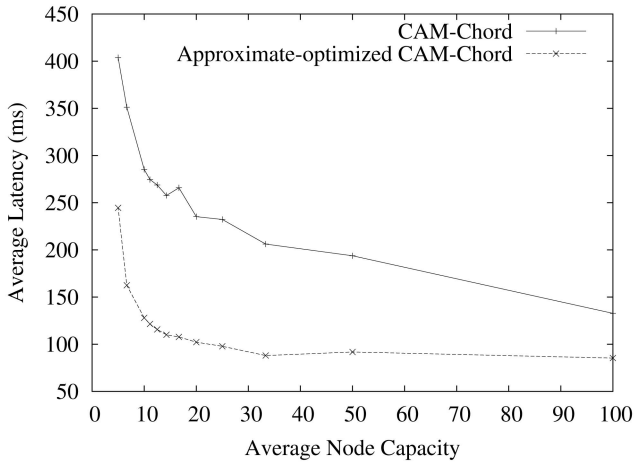

Fig. 10. Proximity optimization.

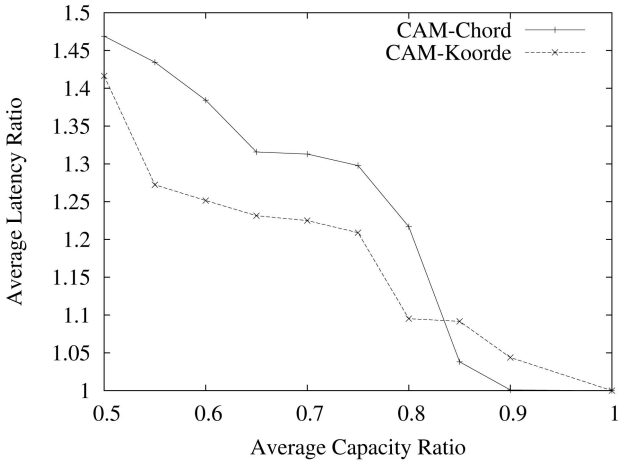

Fig. 11. Throughput versus latency.

capacities are small. Combining these two factors, CAM-Chord creates better balanced trees for small node capacities; CAM-Koorde creates better balanced trees for large node capacities.

Next, we use CAM-Chord as an example (Section 5.2) to study the impact of proximity optimization. In [33], 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 5.0 and the average packet delay to be 50 ms. Fig. 10 compares the average latency of delivering a multicast message from a source to a receiver in CAM-Chord with or without the proximity optimization. The simulation is performed for different average node capacities, and the impact of proximity optimization is significant. In most cases, it reduces the latency more than by half.

6.5 Impact of Dynamic Capacity Variation

In a real environment, the upload bandwidth of a node may fluctuate. If we always use the same implicit multicast trees, then the dynamic variation of node capacities will cause variation in average throughput but not in average latency. CAMs can also be easily modified to ensure throughput, but allow latency variation. If a node's capacity decreases, it simply forwards messages to a smaller number of neighbors, which automatically reshapes the implicit tree. If a node's capacity increases for a long time, the node can take advantage of the improved capacity by increasing the number of neighbors.

We define the *capacity ratio* as the actual capacity of a node x at the real time divided by the claimed capacity c_x that is used to build the topology of CAMs. We define the *latency ratio* as the actual delay at a given capacity ratio divided by the "benchmark" delay when the capacity ratio is 100 percent, namely, no dynamic capacity variation. Apparently, the latency ratio is a function of the capacity ratio.

Fig. 11 shows the relation between the latency ratio and the capacity ratio. When the capacity ratio is smaller, which means the nodes cannot support as many children as they have claimed, the nodes will forward the received messages to a fewer number of neighbors such that each child will still receive a sufficient amount of bandwidth. But, the height of the implicit multicast tree will be larger, which means a larger latency ratio. As show in the figure, when the average capacity becomes 50 percent of the claimed one, the average latency is increased by 47 percent and 42 percent for CAM-Chord and CAM-Koorde, respectively. It shows that the dynamic capacity variation will cause the latency variation (instead of throughput variation).

7 Conclusion

This paper proposes two overlay multicast services, called CAM-Chord and CAM-Koorde, which are capacity-aware extensions of Chord and Koorde with multicast routines that follow implicit, well-balanced trees to disseminate multicast messages. One attractive property is that the number of multicast children of a node is bounded by its capacity, which may vary widely among the nodes. It prevents the multicast throughput from degrading due to the overload of low-capacity nodes. With each source node having a separate, implicit multicast tree, the overall traffic is well balanced across the network.

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 '98,* May 1998.

[2] Y.H. Chu, S. Rao, and H. Zhang, "A Case for End System Multicast," *Proc. SIGMETRICS '00,* June 2000.

[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 '00,* Oct. 2000.

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

[5] S.E. Deering, "Multicast Routing in a Datagram Internetwork," PhD thesis, Stanford Univ., Dec. 1991.

[6] S.E. Deering, D. Estrin, D. Farinacci, V. Jacobson, C.-G. Liu, and L. Wei, "An Architecture for Wide-Area Multicast Routing," *Proc. ACM SIGCOMM '94,* pp. 126-135, Aug. 1994.

[7] Y. Chu, S.G. Rao, S. Seshan, and H. Zhang, "Enabling Conferencing Applications on the Internet Using an Overlay Multicast Architecture," *Proc. ACM SIGCOMM '01,* Aug. 2001.

[8] B. Zhang, S. Jamin, and L. Zhang, "Host Multicast: A Framework for Delivering Multicast to End Users," *Proc. INFOCOM '02,* June 2002.
[9] G.-I. Kwon and J.W. Byers, "ROMA: Reliable Overlay Multicast with Loosely Coupled TCP Connections," *Proc. INFOCOM '04,* Mar. 2004.
[10] P. Mohapatra and Z. Li, "Impact of Topology on Overlay Routing Service," *Proc. INFOCOM '04,* Mar. 2004.
[11] 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.
[12] F. Baccelli, A. Chaintreau, Z. Liu, A. Riabov, and S. Sahu, "Scalability of Reliable Group Communication Using Overlays," *Proc. INFOCOM '04,* Mar. 2004.
[13] V. Pappas, B. Zhang, A. Terzis, and L. Zhang, "Fault-Tolerant Data Delivery for Multicast Overlay Networks," *Proc. Int'l Conf. Distributed Computing Systems '04,* Mar. 2004.
[14] 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.
[15] R. Zhang and Y.C. Hu, "Borg: A Hybrid Protocol for Scalable Application-Level Multicast in Peer-to-Peer Networks," *Proc. NOSSDAV '03,* 2003.
[16] B.Y. Zhao, J.D. Kubiatowicz, and A.D. Joseph, "Tapestry: An Infrastructure for Fault-Tolerant Wide-Area Location and Routing," Technical Report UCB/CSD-01-1141, Univ. of California at Berkeley, Apr. 2001.
[17] A. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems," *Proc. Middleware '01,* Nov. 2001.
[18] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Application-Level Multicast Using Content-Addressable Networks," *Proc. Second Int'l Workshop Network Group Comm. (NGC '01),* 2001.
[19] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content Addressable Network," *Proc. ACM SIGCOMM '01,* Aug. 2001.
[20] 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 '03,* Feb. 2003.
[21] 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.
[22] S. Shi, J. Turner, and M. Waldvogel, "Dimensioning Server Access Bandwidth and Multicast Routing in Overlay Networks," *Proc. NOSSDAV '01,* June 2001.
[23] S. Shi and J. Turner, "Routing in Overlay Multicast Networks," *Proc. INFOCOM '02,* June 2002.
[24] J. Albrecht, D. Kosti, A. Rodriguez, and A. Vahdat, "Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh," *Proc. Symp. Operating Systems Principles '03,* Oct. 2003.
[25] S. Banerjee, C. Kommareddy, B. Bhattacharjee, K. Kar, and S Khuller, "Construction of an Efficient Overlay Multicast Infrastructure for Real-Time Applications," *Proc. INFOCOM '03,* Mar. 2003.
[26] A. Riabov, L. Zhang, and Z. Liu, "Overlay Multicast Trees of Minimal Delay," *Proc. Int'l Conf. Distributed Computing Systems '04,* Mar. 2004.
[27] 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 '04,* Mar. 2004.
[28] 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.
[29] M. Kaashoek and D. Karger, "Koorde: A Simple Degree-Optimal Distributed Hash Table," *Proc. Second Int'l Workshop Peer-to-Peer Systems,* Feb. 2003.
[30] K.P. Gummadi, R. Gummadi, S.D. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica, "The Impact of DHT Routing Geometry on Resilience and Proximity," *Proc. ACM SIGCOMM '03,* Aug. 2003.
[31] S. Sen and J. Wong, "Analyzing Peer-to-Peer Traffic across Large Networks," *Proc. Second Ann. ACM Internet Measurement Workshop,* Nov. 2002.
[32] S. Ratnasamy, "Routing Algorithms for DHTs: Some Open Questions," *Proc. First Int'l Workshop Peer-to-Peer Systems,* Mar. 2002.
[33] 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.

Zhan Zhang received the MS degree in computer science from the Fudan University of China in 2003. Since 2003, he has been working toward the Phd degree in computer and information science and engineering at the University of Florida. His current research fields include sensor networks and overlay networks.

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 had 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, quality of service, and sensor networks.

Yibei Ling received the BS degree in electrical engineering from Zhejiang University in 1982, the MS degree in statistics from Shanghai Medical University (now Fudan University) in 1988, and the PhD degree in computer science from Florida State University at Miami in 1995. He is a research scientist in applied research, Telcordia Technologies (formerly Bellcore). His research interests include distributed computing, query optimization in database management system, scheduling, checkpointing, system performance, fault localization and self-healing in mobile ad hoc networks, and power-aware routing in mobile ad hoc networks. He has published several papers in the *IEEE Transactions on Computers*, *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Biomedical Engineering*, SIGMOD, ICDE, PODC, and *Information System*. He is the architect, as well as developer, of the voice subsystem of Telcordia Notification System. He is a member of the IEEE.

Randy Chow received the BS degree in electronic engineering from National Chiao-Tung University, Taiwan, in 1968, and the MS and PhD degrees from the Department of Computer and Information Science at the University of Massachusetts in 1974 and 1977, respectively. He has been on the faculty in the Computer and Information Science and Engineering Department at the University of Florida since 1981, where he is currently a full professor. His research interests are distributed systems, computer networks, and security with a focus on ontology-based information access models for workflow systems. Dr. Chow's recent affiliations include serving as a program director for the Distributed Systems and Compiler Program at the US National Science Foundation from 2001 to 2003 and a visiting professor at National Chiao-Tung University form 2003 to 2004. He is a member of the IEEE, ACM, and the editorial board of the *Journal of Information Science and Engineering*.

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