An Utility-driven Routing Scheme for Scaling Multicast Applications

Yuehua Wang[1,2,3] Ling liu[1] Calton Pu[1] Gong Zhang[1]
[1] Georgia Institute of Technology,Atlanta,GA,USA 30332,
[2] State Key Laboratory of Virtual Reality Technology and Systems
[3] Beihang University Beijing,China,100083
Email:{yuehuaw,lingliu,calton.pu,gzhang3}@cc.gatech.edu

Abstract—**Multicast is a common platform for supporting group communication applications, such as IPTV, multimedia content delivery, and location-based advertisements. Distributed hash table (DHT) based overlay networks such as Chord and CAN presents a popular distributed computing architecture for multicast applications. However, existing research efforts have been mostly dedicated to efficient message delivery techniques to alleviate the influence of network dynamics on geo-distance based routing, such as reducing the delivery path length or optimizing routing path by utilizing network locality.**

In this paper, we argue that the geo-distance based routing protocols used in existing overlay networks are inefficient in terms of both resource use and environmental accommodation for multicast applications. We devise a utility driven routing scheme to improve the routing efficiency with three unique features. First, our utility function is defined based on a careful combination of hop counts and routing path latency. Second, we use CAN-like routing as an example and extend it by utilizing shortcuts to reduce the routing path length and by introducing a utility function to combine path latency with geo-distance based metric in determining the near-optimal route for each routing request. Third and most importantly, our utility function is designed by using a tunable influence parameter to allow nodes to adaptively making the most promising routing decision according to their specific network state and circumstances, such as overlay connectivity and next hop latency. Our experimental evaluation shows that the utility-driven routing scheme is highly scalable and efficient compared to existing geo-distance routing protocols and demonstrates that by combining shortcuts, path latency with geo-distance can effectively enhance the multicast delivery efficiency for large scale group communication applications.

I. INTRODUCTION

The rapid growth of wireless communication technology and increasing popularity of hand-held devices are continuously escalating communication applications, such as proximity based advertising, instant messaging and software distribution. At the same time, the success of BitTorrent, Emule and Skype has made decentralized overlay networks an attractive alternative computing paradigm for distributed interactive applications and information dissemination services by harnessing widely distributed, loosely coupled, and inherently unreliable computer nodes at the edge of the Internet.

A fair number of research projects has been engaged in supporting multicast applications through distributed hash table based overlay networks. Research on building scalable and fault-tolerant decentralized architecture for multicast applications has progressed along two distinct but complementary directions. The first direction of research has been dedicated to enhancing the topology and routing protocol of the DHT overlay networks, especially Chord [1], Pastry and Tapestry. The second direction of research has been focused on techniques to enhance the multicast algorithms running on top of existing decentralized overlay networks. Example multicast systems include the DHT based multicast systems, such as Scribe [2], Peercast [3], Splitstream [4], bullet [5] and NICE [6], as well as unstructured overlay multicast systems that use gossip-based routing algorithms, such as Coolstream [7], ESM and Chainsaw [8]. By carefully examining these diverse research efforts, we observe some interesting and important facts:

First, it is commonly recognized that the properties of the underlying overlay networks, such as the communication efficiency and system scalability, tend to dominate the performance of the overlay system and the multicast applications built on top of it. Concretely, the efficiency of any overlay network heavily depends on the efficiency of its routing protocol. Thus, routing efficiency is a key performance indicator for both the underlying overlay network and its multicast applications.

Second, most of the overlay network topologies and routing protocols do not match well with the packet routing structure in the underlying network. It is common that one hop distance in an overlay network may incur IP traffics across two continents and lead to a long link latency in the underlying network. Thus, routing efficiency should take into account of reducing the routing path length and path latency, as well as optimizing routing path by utilizing the network locality.

Third, the performance enhancement to the underlying DHT networks is typically independent of and complementary to the existing multicast algorithms developed for decentralized overlay networks, such as Splitstream [4], Coolstream [7], to name a few.

Surprisingly, existing research efforts have been mostly dedicated to efficient message delivery techniques, such as reducing the delivery path length (hop counts) or optimizing routing path by utilizing network locality. We argue that the geo-distance based routing protocols used in existing overlay networks are inefficient for supporting multicast applications due to two reasons. First, most of the overlay routing protocols

fail to make a careful integration of path length, path latency, and network locality into the underlying geo-distance based routing algorithm. In their system, either nodes or links might be imposed on heavy load, which results in poor system performance. Second, few overlay routing schemes to date is capable of adapting its routing decision for each message to the network dynamics. In such a case, some data messages might have a long transmission delay or be lost before reaching the destination node due to ignoring the network state. It thus is a necessity to have an efficient routing protocol with consideration of each message's specific situation.

In this paper we propose a new scheme named Utility Driven Routing (*UDR*) to improve the efficiency of geo-distance routing protocols and enhance the performance of application based on it. Our utility-driven method has three unique features. First, we define the utility function based on a careful combination of hop counts, routing path latency, and geographical locality of nodes. Second, given the nature of CAN-like DHT that is considered more reliable than Chord-like DHT due to its multi-dimensionality characteristics, we use CAN-like routing as an example and extend the CAN geo-distance based routing by utilizing shortcuts to reduce the routing path length and by introducing a utility function to combine path latency with geo-distance based metric in determining the most promising route for each routing request. Third and most importantly, we design our utility function with a tunable influence parameter to allow nodes to adaptively make the near-optimal routing decision according to their specific network state and circumstances, such as overlay network connectivity, next hop latency. Thus, our utility based routing scheme can dynamically determine the best routing path for each message in terms of hop counts and routing latency.

To compare the utility-driven routing scheme with existing CAN-like geo-distance based routing approach, we develop GeoCast, a CAN-like decentralized geographical overlay for end-to-end multicast services. We support geo-distance based routing in basic GeoCast system and implement the geographical proximity aware *UDR* routing in enhanced GeoCast system.

The rest of the paper is organized as follows. We first discuss related work in section 2 and then describe the structure of basic GeoCast system and the motivation of the design of our utility-driven routing scheme in section 3. We describe the design of our utility-driven routing scheme and analyze the settings of the tunable influence parameter in section 4. Two optimization techniques are discussed in Section 5 to further enhance the performance of our routing scheme. We evaluate the effectiveness of the proposed approach through simulation based experiments in section 6 and summarize the contributions of the paper in Section 7.

II. RELATED WORK

A fair amount of research has been carried out for improving DHT routing efficiency [9][10][11], most of which incorporate proximity into the DHT routing protocols. Generally, they can be classified into three categories: proximity neighbor selection (PNS), proximity route selection (PRS), and proximity identifier selection (PIS).

PNS selects routing state entries for each node from the closest nodes in the underlying topology that satisfy constraints required for overlay routing. When message arrives, the host node routes the message to the node that is numerically closet to the destination [10][11]. PNS is appropriate since it can achieve both low delay routes and low bandwidth usage. However, it comes at the expense of high overhead. . Unlike PNS, nodes in PRS are dedicated to minimizing the hop routing latency under the constraint that each hop should be closer to the destination. But it may lead to a long routing path in terms of hop counts and end-to-end latency of message routing could be relatively higher than other proximity based DHT routing. A detailed comparative study of PNS and PRS over a variety of overlay networks [9] shows that PNS is significantly more efficient in dealing with proximity. However, choosing the closest node from the message hosting node as the next routing hop may not necessarily be the best choice. We will use an example to show it in section III.

The PIS approaches are designed to solve the problem of mismatch between overlay network and underlying network. In PIS, the nodes are assigned with similar identifiers when they are located closely in the underlying network [3][12][13]. It potentially ensures the links between nodes locating in the vicinity are with low latency. However, this method has a main drawback in load balancing, where nodes in PIS may be imposed on heavy load when their contents are highly desired. In such a case, the messages in PIS may be delayed, which leads to a significant degradation of system performance.

It is important to note that the goal of minimizing the routing path length (hop counts) can sometimes be in conflict with the goal of minimizing the routing path latency. Surprisingly, none of the existing routing schemes has shown how to make the best routing decision by a careful tradeoff between minimizing routing path length and minimizing routing path latency. Our utility-driven routing scheme presented in this paper, to the best of our knowledge, is the first one that provides a combination of techniques to allow nodes to adaptively make the best routing decision, including utilizing shortcuts to reduce the routing path length, introducing utility function and a tunable influence parameter to integrate path latency and shortcut with geo-distance based routing.

III. BASIC GEOCAST SYSTEM

GeoCast is a geographical overlay system built on top of GeoGrid [14] for providing group communication services. It is composed of two-tier substrates: overlay network management and end system node multicast management.

A. Overlay Network Management

This is the lower tier for overlay membership management, lookup and communication. It consists of two main components: membership protocol and routing lookup protocol.
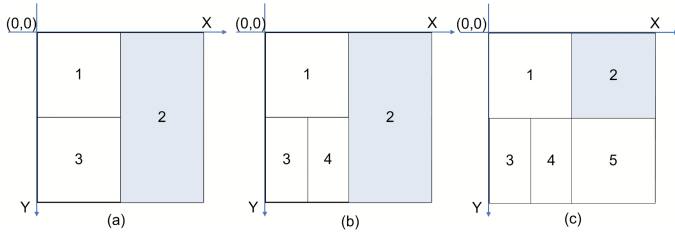
Fig. 1.   An GeoCast overlay network

1) Membership Protocol: GeoCast system uses this protocol to organize widely distributed end system nodes into an overlay network that carries the multicast service. Similar to CAN [15], N nodes dynamically partition the entire GeoCast coordinate space into N disjoint rectangle regions such that each node "owns" one such rectangle region.

A new node can join GeoCast by initiating a joining request with its own geographical coordinates to an entry node obtained from a bootstrapping server [14]. The joining request is then routed towards the region that covers the coordinates of the new node. After identifying the region to which the new node belongs, the owner node of the region compares its $unit\_capacity$ to that of its neighbors and forwards the request message with split tag to the node with least $unit\_capacity$. $unit\_capacity$ represents the relative capacity of the node, defined by $unit\_capacity_i = E_i.capacity/(E_i.R.w \times E_i.R.h)$, where $E_i.capacity$ refers to the specify attribute of E_i. It may represent CPU power, memory, bandwidth. In GeoCast, we use it to denote the available bandwidth of the node. $E_i.R.w$ and $E_i.R.h$ refer to the width and height of region R owned by E_i.

After receiving the tagged message, the split node divides its region into two halves and assigns one half to the new joining node. Then the notification messages are sent out from split node to notify of the arrival of new node. In such a way, the new node can be included in $Peernodelist$ of other nodes. Readers may refer to our technical report [16] for detailed construction process and examples of node arrival and departure.

2) Routing Lookup Protocol: In GeoCast, each node keeps a set of information about other nodes in the network in its $Peernodelist$, denoted by $Peernodelist(i) = \{S_i^0, S_i^1, \ldots, S_i^j, \ldots, S_i^Q\}$, where Q is defined by $\log_2(G.w * G.h/(E_i.R.w * E_i.R.h))$. For each subset S_i^j $(1 \leq j \leq Q)$, it contains the nodes in the geographical enclosing zone EZ_i^j with the size of $1/2^j$ of the geographical plane G. We define $EZ_i^j :< x, y, w, h >$, where (x, y) represent the coordinates of top left vertex of enclosing zone and (w, h) refer to the width and height of EZ_i^j. Nodes in the subset S_i^j are viewed as representatives of the enclosing zone EZ_i^j. If a message needs to be routed from node i to a destination node contained in the enclosing zone EZ_i^k, it is more likely that a shortcut node in the subset S_i^k be chosen as the next message forwarding hop.

Fig.1(a) provides a snapshot of GeoCast overlay network with three end system nodes. At this point, there is no shortcut node included in the $Peernodelist$ for any of nodes. For node 2, $Peernodelist(2) = \{S_2^0\} = \{\{1, 3\}\}$, where node 1 and node 3 are its neighbors included in EZ_2^0 that is identical to the entire plane G. With the arrival of node 4 as shown in Fig.1(b), node 3 is no long a neighbor of node 2 and is now recorded as a shortcut node in $Peernodelist(2)$. Given the locations of nodes, S_2^0 is represent by: $S_2^0 = \{1, 3, 4\}$, where $E_1, E_3, E_4 \in EZ_2^0$,. Similarly, node 4 becomes another shortcut node of node 2 after node 5 joins the network as shown in Fig. 1(c). Now $Peernodelist(2) = \{\{1, 3, 4\}, \{5\}\}$, where $E_5 \in EZ_2^1 \cup E_5 \notin EZ_2^0$. In this way, each node in the system keeps building up its $Peernodelist$ as the topology of network evolves with the arrival or departure of nodes.

Shortcut nodes need to be maintained in order to keep the desired routing efficiency. Every node periodically checks the state of its shortcut nodes. If one of shortcut nodes is overloaded or has moved out of the system, a new shortcut node is selected from the same enclosing zone as the replacement by using random walk [17]. In this paper, we omit the algorithm and detailed discussion about the $Peernodelist$ construction for lack of space. Readers may refer to our technical report [16] for further details.

Routing in Basic GeoCast In the basic GeoCast, each service request message has specified a destination point or region in the space G. We define geo-distance $Gdist$ as a routing metric to discover the best routing path for each message. $Gdist$ is the distance between two end system nodes E_i and E_j on space G, and is represent by $Gdist_{i \rightarrow j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$, where (x_i, y_i) and (x_j, y_j) are the unique identifier of E_i and E_j respectively.

Once a node p wants to route a message to the node with the given destination coordinates, it first checks if the coordinates are contained by the region it owns. If not, it looks up the routing nodes in its $Peernodelist$ and chooses the node with the shortest distance to the destination as its next hop to routes the message. This routing process repeats until the message reaches its destination.

B. Multicast Management

This is the higher layer for multicast service publication, subscription management, multicast payload delivery and group membership management. It is built on top of the overlay network management substrate and uses its API to carry out management functions.

In general, there are four basic operations for multicast service establishment and maintenance:

Publishing the Multicast Service Multicast sources can join GeoCast as peers or select nodes in GeoCast to be their delegates for the purpose of information dissemination. Each multicast service is associated with two identifiers: service identifier and group identifier. The service identifier will be used to advertise and publish meta-information about the service, whereas the group identifier is used by other peer nodes to subscribe or unsubscribe the multicast service.

Tree-based Subscription and Un-Subscription Ech multicast tree is formed by the routes from the subscribe nodes to the
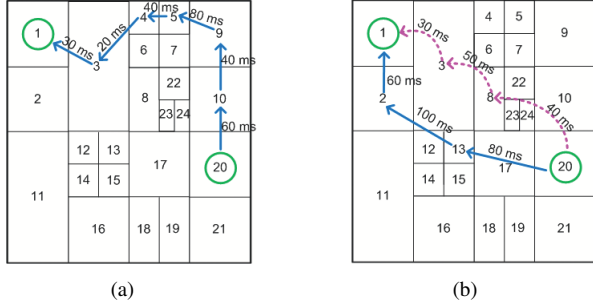
Fig. 2. Examples to illustrate neighbor-based routing and geo-distance routing

source node. Nodes in GeoCast can subscribe to any multicast service published in the network by initializing a subscription request. Such request is routed to the multicast source in two ways. First, if the request encounters a node p that has already subscribed to the multicast service of interest, the routing of request stops. Otherwise, the request message is routed to the multicast source. A node can unsubscribe to a multicast service by removing itself from the corresponding multicast tree.

Dissemination of Multicast Payload The source of a multicast service uses the corresponding multicast tree for delivering the multicast data to all the subscribers. It injects the data at the root of the multicast tree, which gets disseminated through the tree and reaches all the subscribers.

Multicast Group Management Every Node in the multicast tree maintains the information about its parent node and children nodes Periodically, the node exchanges heartbeat messages with those nodes and updates the information about their state. To reduce the overhead introduced by multicast maintenance, the nodes' update information is piggybacked in heartbeat messages used by shortcut maintenance or in data messages transmitted among nodes.

C. Motivation for Utility-driven Routing

In GeoCast, each node has an average of $O(2d)$ neighbors and $O(logN)$ short nodes maintained in its *Peernodelist*, where d is the dimensions of coordinate space and N is the number of nodes currently in the system. Unlike CAN like geo-distance based routing, the scheme of geo-distance routing with shortcut ensures that any node in the system can be reached in less than $O(logN)$ hops, achieving similar performance to Chord [1] and Expressway [13]. Fig.2 illustrates the difference of the routing schemes using an example. Given source node S and destination D in a system of 24 nodes. The scheme of geo-distance routing with shortcut only needs 3 hops to reach the destination (see solid line in Fig. 2(b)). On the contrary, CAN like geo-distance routing needs two times as many hops as that of geo-distance routing with shortcut nodes, as shown in Fig. 2(a).

Even though the geo-distance routing with shortcut in the basic GeoCast is more efficient than the CAN like geo-distance routing in terms of routing path length (hop counts), we argue that the shortest geo-distance routing path may be a long forwarding path in terms of network latency, which leads to inefficient routing performance. Concretely, as shown in

Fig. 2(b), by using the shortest geo-distance routing path, the message reaches the destination in 240 ms, denoted by the solid line in Fig. 2(c). In contrast, the dashed line gives a faster routing path by incorporating network latency in routing selection algorithm, only 120 ms.

This is because the link transmission latency in the first two hops of the shortest geo-distance routing path are relatively high with up to 100 ms delay, which can be caused by either bad IP traffic in those regions or low capacities of the forwarding nodes along the routing path. We observe from this example that combining the shortcut and latency enables us to enjoy the benefit from one hop and yet fast routing jump to the region close to the destination. Thus we conjecture that the best routing path should be the one that have both short geo-distance (minimizing path length or hop counts) and short network latency (minimizing path latency). Clearly, introducing latency metric for CAN like geo-distance routing as suggested by previous studies [15] [18] is insufficient due to the long routing length.

This example also shows that neither the routing path with the shortest routing length in hop counts nor the routing path with each hop optimized by the shortest link latency is the best route in terms of routing efficiency. Thus, it is a necessity of having an efficient way to combine the shortcut and latency together for the purpose of optimizing the routing performance. To this end, our Utility-Driven Routing (UDR) protocol is proposed.

IV. UTILITY DRIVEN ROUTING IN GEOCAST

In this section, we describe the design of our Utility-Driven Routing (UDR) protocol. We first give a design overview of the utility function, then discuss how to utilize this utility function to set up the best message delivery path in terms of short path length and short path latency. For reference convenience, we refer to the version of GeoCast powered by utility driven routing as the *enhanced GeoCast* in the rest of the paper.

A. GDV Utility Function

1) Definition of GDV: The utility function is designed to compute a utility value for each routing entry in the *Peernodelist* of a given node by taking into account of the link latency and the geo-distance from the next hop to the destination. We refer to this utility value as the latency-enhanced geo-distance (GDV) score, which is designed to measure the qualification of entry node in the *Peernodelist* to be a forwarding node for a message with source node S and destination node D. We define the GDV utility function by introducing a tunable influence parameter μ, ranging from 0 to 1, to adjust the importance of *geo-distance factor*, denoted by f_{dis}, and *latency factor*, denoted by f_{la}. With those notions, the GDV function is defined as follows:

$$GDV_i = (1-\mu) * f_{la}(i) + \mu * \eta * f_{dis}(i) \quad (1)$$

$$f_{la}(i) = \frac{RTT_{\rightarrow i}}{2} \quad (2)$$

$$f_{dis}(i) = Gdist_{i \rightarrow D} \quad (3)$$

Where η is the *normalization parameter* designed to unify the dimensions of the two factors, defined by: $\eta = RTT_{avg_{dis}} /(2 * avg_{dis})$, where avg_{dis} and $RTT_{avg_{dis}}$ denote the average geo-distance among the neighbor nodes and its associated average round trip time (RTT) respectively.

The latency factor $f_{la}(i)$ refers to the delay of link between the current node and the next hop node E_i. Given that RTT_i denotes the time required for a message from current node to an entry node E_i in *Peernodelist*, and back to the current node, we have $f_{la}(i) = RTT_{\rightarrow i}/2$. f_{dis} represent the geo-distance from the next hop node E_i to the destination D. In GeoCast, every node treats $f_{la}(i)$ as soft state. Every T seconds, $f_{la}(i)$ is updated via heartbeat message, where T is a system parameter configured by default.

(a)   (b)

(c)   (d)
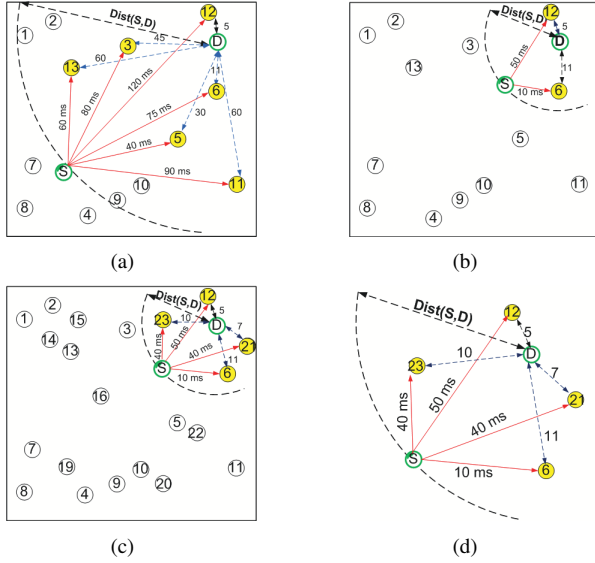
Fig. 3. Significance of setting μ parameter adaptively

2) Setting of Parameter μ: The decision on how to set μ involves a trade-off between the importance of *latency factor* and *distance factor*. The larger the μ value is set, the more weight of importance the distance factor will get, and thus the shorter routing path length (hop counts) is preferred. In the extreme case of $\mu = 1$, the UDR routing is identical to basic Geo-distance based routing with shortcuts, which is good at delivering the message with minimal routing hops but may take longer to deliver the message from source S to destination D in terms of latency. In contrast, with setting of $\mu = 0$, the node connected with the lowest delay link is selected to be the next forwarding node in every routing step regardless of its distance factor, which likely leads to the routing path with larger number of hops from source S to destination D, and consequently longer overall routing latency. The value of μ can not be too small if we want to keep the routing length in an order of $O(logN)$. Furthermore, the value of μ also needs to be set for different pairs of source S and destination D to adapt to the real-time network dynamics.

We use an example to illustrate the significance of selecting good value for the influence parameter μ. Fig. 3 shows three different scenarios of source S and destination D. In each scenario, source node S wishes to route a message to destination node D. For node S, a set of nodes { 3, 5, 6, 11,12,13} can be used as routing node for message forwarding based on its local knowledge about the network, each of which has shorter distance to the destination than $Dist(S, D)$, as shown in Fig.3(a). $Dist(S, D)$ refers to the geo-distance between node S and D. The values carried by solid-line arrows denote link latency required for message transmitted from S to the entries in its *Peernodelist*. We denote the distance between an entry in the *Peernodelist* of S to the destination D by the value carried by dash-line arrows. To prevent the message delivery from long routing path in terms of hop counts, in this example, node 6 is selected to be the best node for message forwarding with the setting of μ to be 0.8 (see Fig. 3(a)) such that the nodes locating in the vicinity of source S has less probability to be selected due to their larger geo-distance to the destination D.

However, this setting of μ may not be optimal when source node S resides closely to the destination node D as shown in Fig. 3(b). Now setting μ to large is not good since the link latency from S to node 12 is the worst (50 ms). To prevent the message delivery from long transmission delay, we set μ to small (say 0.2) and route the message to node 6 through the link with lowest delay. In contrast, such setting is not appropriate when the source node is located in a densely populated area as shown in Fig. 3(c) and Fig. 3(d), a partial enlargement of Fig. 3(c). Now a larger value of μ, such as 0.8, is preferred in order to reduce the number of nodes involved in the message delivery and at the same time minimize the end-to-end latency. In Fig. 3(d), node 21 is selected as the next forwarding node with such a setting and the routing procedure repeats until the message reaches destination node D.

This example shows that the constant setting of the influence parameter μ is impractical in a highly dynamic environment where any two nodes may communicate with one another at any time and the system may have unpredictable churn rate. It is also interesting to observe that for any message, the number of nodes that are suitable to be selected as routing nodes decreases when destination D is approaching.

Inspired by this, we define the parameter μ by using the following equation, aiming to provide an efficient way for nodes to adjust the setting of μ based on their local knowledge about the overlay network.

$$\mu = \frac{\sum_{j=0}^{Q} NS_j}{2Q} + \frac{1}{2^{Q-\tau-1}} \qquad (4)$$

$$NS_j = \begin{cases} 1 & \text{if } \exists E_m \in S_j, f_{dis}(m) < f_{dis}(i) \\ 0 & \text{otherwise} \end{cases} \qquad (5)$$

where $\sum_{j=0}^{Q} NS_j$ denotes the number of subsets in the *Peernodelist* that contains the nodes locating closer to the destination than current node i. NS_j is set to 1 when subset S_j satisfies: $\exists\ e_m \in S_j,\ f_{dis}(m) < f_{dis}(i)$. Q refers to the number of subsets kept in $Peernodelist(i)$ for a given

end system node E_i. The larger the system is, the smaller the responsible region node has, the bigger Q is. τ denotes the maximum index of the subsets whose enclosing zone contains both destination and current node. As the destination is approaching gradually, the value of τ is increased.

Now we analyze the formula for determining μ. For a given Q, the first component in the formula indicates relative position of the current node to the destination. Typically, a large value of $\sum_{j=0}^{Q} NS_j$ means that node E_i locates in a region that is relatively far away from destination, and it has many nodes in its *Peernodelist*, which are suitable to be selected as routing nodes. To reduce the routing path length, a bigger value of parameter μ is set in such situation. On the contrary, when the latency factor is more important than distance factor in determining the routing path, a small value of μ is preferred.

The second component in the formula is $1/2^{Q-\tau-1}$, which is introduced for two purposes. On the one hand, we observe that nodes locating in the sparsely populated area tend to have fewer nodes contained in their *Peernodelist*s even in a large network and consequently they might have less knowledge about the network. The factor of $1/2^{Q-\tau-1}$ takes a larger value when both Q and τ are small. In this case, the node with shorter geo-distance to the destination has higher priority to be selected as the forwarding node. On the other hand, the factor of $1/2^{Q-\tau-1}$ prevents μ from stumbling in a small value. Given the scenario 3 in Fig. 3(c), it is desirable for μ to be set to a larger value even when the node E_i is not far away from the destination node D. Note that the first component of the μ formula is designed to make μ converge to a smaller value when the destination is nearby and set μ larger when the destination is relatively far away from the current node. Thus, the second component is employed to alleviate the influence of the first component when the distance factor should play a more important role in routing node selection even though the destination is nearby in terms of geo-disance. (τ is small and Q is small).

Comparing with existing approaches that utilize link latency, our approach to tune the influence parameter μ in the GDV utility function is unique and highly effective because it allows a message to be routed at different nodes with different μ depending on multiple factors, including link latency, distance to destination, node density nearby the destination. Thus our utility driven routing protocol is highly adaptive to both the real time network dynamics and the location of the source and destination of a message.

In enhanced GeoCast, nodes use the utility based routing algorithm to forward the message, until it reaches destination node specified by message. Along the routing path, each node keeps computing the geo-distance based filter for itself by examining the entries in its *Peernodelist*, and uses it to generate a GDV candidate list. This candidate list contains all nodes that are closer to the destination node in terms of geo-distance than the current node. By calculating the GDV value for every node in the candidate list produced above, the message is routed to the node with the smallest GDV value.

This procedure repeats until the destination node is reached. Due to the limitations of space, the detailed routing algorithm of UDR is omitted here and the detail can be found in [16].

V. Performance Optimizations

In this section, we introduce two optimization techniques implemented in GeoCast to enhance the performance of the UDR algorithm. The first optimization is to use shortcut clustering to enable the different applications to control the amount of shortcuts to be created and maintained at each node by balancing the storage and maintenance cost of shortcuts. The second optimization focuses on reducing the impact of network dynamics, especially the oscillation of link latency on the stability of multicast trees in GeoCast.

A. Shortcut Clustering

The idea behind the shortcut clustering optimization is to maintain "enough" shortcuts for each node instead of all shortcut nodes in addition to their neighbor nodes. It offers end system nodes with ability to keep those shortcut nodes that have higher capacity and thus can handle more routing workload and also to reduce the shortcut nodes that are representative for the same enclosing zone or for the close-by regions of small sizes.

In the first prototype of GeoCast, we introduce a system defined parameter m for shortcut clustering such that each node maintains no more than m shortcuts. The setting of m allows us to limit the size of each *Peernodelist* maintained in the system. As the network grows, the new shortcuts are being grouped into m clusters of shortcuts, each of which selects one representative shortcut to keep in the *Peernodelist*.

The shortcut clustering algorithm is triggered at node p once the size of node p's *Peernodelist* is above m. The shortcut clustering algorithm first groups the entries in *Peernodelist* into m clusters with respect to their geographical proximity and then computes the probability of nodes to be cluster representor by using $Prob_m^i = (i+1)/\sum_{k=1}^{Q+1} k$, where $i = 0, 2, \cdots, Q$. It allows low probability to those subsets that are far away from node p and higher probability to those nodes closer to node p in terms of the geo-location of shortcuts. Through a comparison, the shortcut with the lowest probability is select to represent its cluster and the others with higher probability are discarded.

Obviously, the choice of parameter m has significant influence on the routing performance of UDR. The higher m value means more shortcut nodes are maintained in the *Peernodelist*, the more accurate the GDV values are, and the higher probability the UDR can find the best forwarding path in terms of routing efficiency. However, by setting m to be a reasonable value with respect to the size of the network, our experimental results show that the UDR routing algorithm optimized with shortcut clustering can offer comparable performance of UDR without shortcut clustering.

B. Accommodating the Dynamics of Network Link

We observe that the link latency metric used in our UDR algorithm can at times lead to serious instability of multicast

TABLE I
PERFORMANCE METRICS

Term	Description
Path length	The average number of routing hops required for a message transmission from one node to another node
Processing latency	The sum of processing delay at each node along the routing path
Routing latency	The sum of propagation delay of individual overlay links along routing path
End-to-End latency	The sum of processing latency and routing latency
Multicast latency	The average time consumed for messages transmission from publisher to its subscribers in a tree.
Shortcut availability	The ratio of m to the number of shortcuts in *Peernodelist* without shortcut clustering
Maintenance cost	The number of heartbeat messages consumed on the network
Link stress	The ratio of the number of messages generated by an overlay multicast tree to the number of messages generated by an IP multicast tree
Churn rate	The proportion of failure nodes to the total system nodes

tree and most of time such instability is unnecessary. By using an exponential smoothing algorithm as the one employed in [19], we can keep the advertised link latency unchanged until the estimate value differs from the current latency by a significant amount. However, it comes at the expense of serious delay in reacting to the network changes and the significant degradation of the routing performance.

Our approach to address this problem is to employ the concept of latency levels. The motivation is to determine the adequate threshold value for capturing the significant changes in the advertised link latency statistics. In terms of latency distribution, the links among nodes in the system is divided into L levels. The link latency is rounded up to the nearest latency level. We move up a level immediately when the measured value exceeds the current level, and move down a level only if the value is significantly below the next level. Such discrimination latency ensures that all overlay links can fall into a small set of equivalence latency intervals, represented by $(i * \rho, (i + 1) * \rho]$. ρ is a system parameter determined by default set at the system configuration. We use the upper bound $(i+1) * \rho$ to represent the link latency that belongs to interval $(i * \rho, (i + 1) * \rho]$. For example, with setting of $\rho = 20$, an overlay link with a measured latency of 94 ms may be viewed as having latency of 100 ms, in a system with levels corresponding to $\{20ms, 40ms, 60ms, 80ms, 100ms, 120ms, \cdots\}$. Two different latency values are considered equivalent if they map to the same level. This approach not only enables greater stability in multicast trees supported on the overlay, but also allows $Gdist$ to be a deterministic factor when different links have similar but not identical latency values.

VI. PERFORMANCE EVALUATION

In this section, we report our experimental evaluation of the utility driven routing (UDR) scheme with respect to effectiveness, scalability and robustness by conducting companions with CAN like geo-distance routing (NB) [15], RTT weighed neighbor-based routing (RNB) [9][10][15], and geo-distance routing with shortcut (SGD).

A. Experiment Setup

We use Transit-Stub graph model from the GT-ITM topology generator to generate network topologies for our simulation. All experiments in this paper are run on 10 topologies with 8080 routers. The results are measured by averaging 100 runs (10 runs in each topology with same setting in such a way

TABLE II
PATH LENGTH FOR ROUTING ALGORITHMS

N	NB	RNB	SGD	UDR with random setting
1000	11.61	14.16	3.96	4.259
2000	14.16	18.23	3.92	4.219
4000	20.19	25.71	4.44	4.458
8000	28.1	35.49	4.76	4.94

the inaccuracy incurred by stochastic selection is minimized). Each topology consists of 8080 nodes with heterogeneous capabilities. At the top level, there are 10 transit domains, each of which contains 8 routers on average that is attached by about 10 stub domains. Given that there is no linear relationship between the nodes' location and their link latency, we assign the link latency by following uniform distribution on different intervals based on their type: [50 ms,80 ms] for intra-transit domain links, [10 ms,20 ms] for transit-stub links, and [1 ms,5 ms] for intra-stub links. Nodes are randomly attached to the stub domain routers and organized into the GeoCast overlay networks.

The metrics used to in our experimental evaluation are summarized in Table I.

B. Effectiveness of UDR

Delay Penalty The impact of routing schemes on the application performance is first investigated based on three metrics: path length, processing latency and routing latency.

We simulated the overlay networks consisting of 1,000 to 8000 nodes. Table II shows the results for four routing schemes: NB, RNB, SGD and UDR with random setting. In the scheme of UDR with random setting, nodes set the parameter μ by following a uniform distribution on the interval [0,1]. Such scheme is the general case of UDR routing, employed as a representative of UDR routing for routing scheme comparison before the impact of adaptive parameter setting is studied in detail. From the results in the Table II, we can see that with random setting, UDR exhibits a better performance than both NB and RNB in terms of path length. Even in the larger system, the differences between UDR and SGD is less than 1 hop. It is important to note that both NB and RNB perform quite poorly in comparison even if the network is small. This is because in those schemes, only neighbors are taken into consideration when selecting the next hop for message delivery, which may result in routing paths that are longer than that of others.
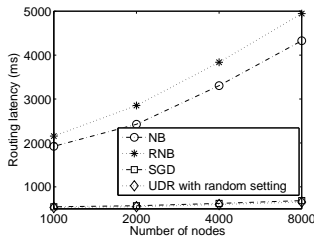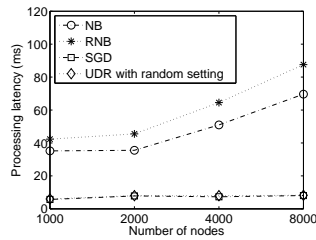
Fig. 4. Routing latency
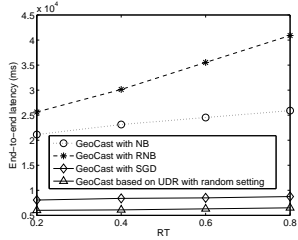

Fig. 5. Processing latency

(a)

(b)

Fig. 6. The latency of multicast tree

(a)

(b)

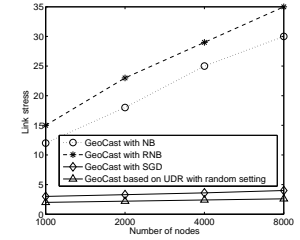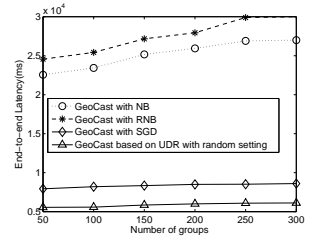Fig. 7. Churn rate on multicast performance
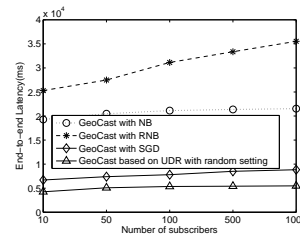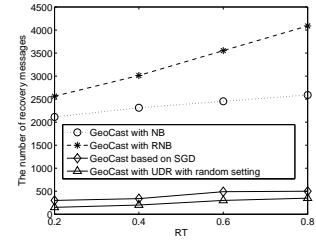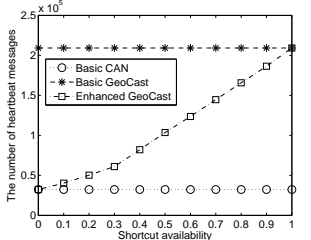

Fig. 8. Link stress

Fig. 9. Maintenance cost

Fig.4 shows the effect of system size on routing latency. Both SGD and UDR outperform others due to their shorter forwarding route in terms of hop counts. We can see that the differences become pronounced when the system size is larger. This is because the routing path is getting longer as the system size as shown in Table II. Specifically, in the system of 8,000 nodes, RNB needs to take about 5000 ms to transmit messages to the destination on average, which is aout 8 times as many as that of shortcut based routing schemes (SGD or UDR).

Next, we measure processing latency incurred by message forwarding among nodes. Fig.5 demonstrates that shortcut based routing schemes perform better than it competitors. With the growth of system sizes, the shortcut based routing scheme keeps a relatively steady performance and take no more than 15 ms for message processing during the entire routing procedure. In contrast, NB and RNB schemes need to take much more time for message processing and this situation gets worse as the system size increases.

We now compare the efficiency of three multicast construction schemes: GeoCast with NB, GeoCast with SGD routing and GeoCast based on UDR with random setting.

Multicast Latency This set of experiments is done by varying the group size from 10 to 1000. In each simulation, we set N to 4,000 and built up 10 trees on average. From the results showed in Fig.6(a), we observe that the latency of GeoCast with NB is significantly higher than that of the other three schemes and GeoCast with RNB improves GeoCast with NB in terms of latency but its latency is still much higher than GeoCast with SGD or UDR, due to the long routing path in hop counts. GeoCast based on UDR with random setting exhibits best performance than NB, RNB, and SGD in all cases. Even if the group size increases to 1,000, it is able to manage the latency variation within a small range. On the contrary, the GeoCast with SGD routing scheme needs to take longer time to deliver multicast services to its subscribers along the tree. Fig.6(b) shows how the latency of multicast tree

changes with the number of groups when we fix the group size to 10.

Fault Resilience Simliar to [3], we now generate a sequence of node failures to study the effects on these three multicast schemes. We randomly choose the failure times of sequence nodes by following independent and identical exponential distribution. We vary the churn rate (CR) from 0.2 and 0.8. As shown in Fig.7, we observe that the performance of our scheme still outperforms others. As CR increases, the latency of multicast trees increase. This is, essentially, because the multicast tree needs to take longer to recover itself from service interruption caused by tree nodes' departure. Moreover, it is important to note that our UDR scheme consumes less recovery messages than that of GeoCast with SGD, as shown in Fig.7(b). Potentially, it demonstrates the efficiency of the UDR routing scheme.

Link Stress To study the impact of application level multicast on physical link, we make a comparison among those schemes in terms of link stress. From the results shown in Fig.8, we can see both GeoCast with SGD routing and GeoCast based on UDR routing exhibit a flat curve, which are consistently about as many as 3 times of that of IP multicast. Contrarily, the links in either GeoCast with NB or GeoCast with RNB are having high load, which confirms the results showed in Table II. Since shorter forwarding path usually incurs fewer IP messages, our scheme is efficient in eliminating redundant physical links.

Maintenance Cost Now, we simulate multicast session to investigate the impact of shortcut clustering on the routing performance of UDR in the system with 4,000 nodes. In each simulation, there are 10 trees consisting of 20 subscribers on average During runtime, root node issues 50 M meta-data to their groups and we measure the messages generated for data transmission and topology maintenance.

Fig.9 and Fig.10 depict the maintenance cost and data transmission cost of different systems as function of shortcut availability respectively. We notice that the more shortcut nodes
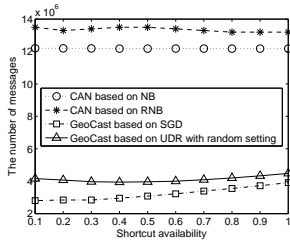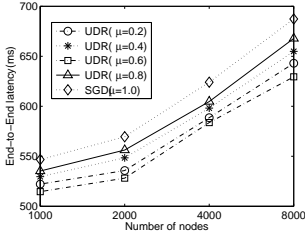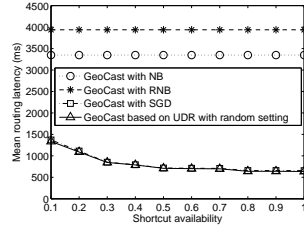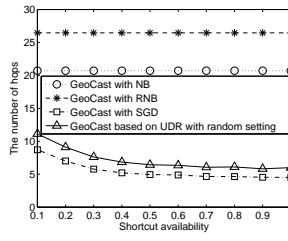
Fig. 10.   Data transmission cost

(a)

(b)

Fig. 11.   Effect of shortcut clustering
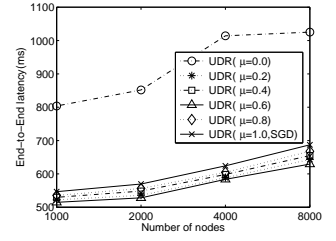
Fig. 12.   End-to-end latency
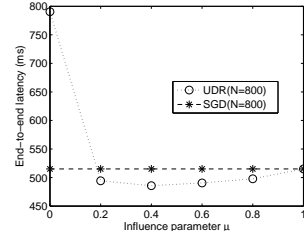

Fig. 13.   Influence parameter
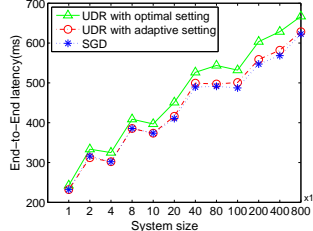
Fig. 14.   System size
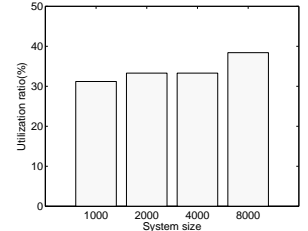
Fig. 15.   Effect of adaptive setting

Fig. 16.   Utilization rate

are maintained by nodes, the more maintenance messages are required in both basic GeoCast and enhanced GeoCast than in CAN system, but on the contrary the less data messages are transmitted in GeoCast. Only about one-third message of those schemes is consumed by GeoCast based on UDR with random setting, which benefits from the shorter message delivery routes formed by the UDR routing scheme. But compared to SGD, the method of UDR with random setting generates more messages during data transmission. It is due to that more intermediate nodes are involved during the multicast session. However, we find that it is negligible given the better performance of GeoCast based on UDG with random setting in terms of end-to-end latency and adaptive ability that are studied below. It is viewed as side-effort of UDR routing.

In Fig. 11(a), we observe that even with a small value of shortcut availability, the schemes of UDR can deliver the messages to the destination nodes at a far more speed. They save around 57% of transmission time required by NB. Interestingly, after shortcut availability reaches 0.4, increasing the shortcut availability further does not achieve dramatic improvement in end-to-end latency. From Fig. 11(b), we also argue that with such setting, the maintenance cost can be constrained within an acceptable level, which relates to the requirement of the applications.

C.The Impact of Adaptive Setting Fig.12 examines the effect of parameter setting on routing performance by comparing the performance of two shortcut based geo-distance routing schemes. We observe that UDR with the setting of $\mu = 0$ does not perform as well as that of other settings in all cases. With such setting, messages tend to be routed to the neighbor nodes through the links with shorter delay, given the fact that nodes that locate closely in the underlay network have a high probability of being neighbor nodes in the geographical overlay network. In Fig.13, we observe that the end-to-end latency can be minimized in all cases when μ is set to 0.6. Rather than reducing the routing performance, as shown in Table II, it limits the routing length of UDR within $O(\log N)$, which makes it more attractive than existing approaches.

Given geo-distance routing (SGD) fails to address the network latency issue in the routing algorithm, it achieves a poor performance. As we discussed in Section 4, the setting of $\mu = 0.6$ is unlikely the optimum parameter of our routing scheme for different scenarios. Fig.14 shows that the optimum parameter setting of μ is changed to 0.4 when the number of nodes in the system decreases to 800.

Fig.15 conducts a performance comparison among three routing schemes: UDR with optimal setting, UDR with adaptive setting, and SGD routing ($\mu = 1$). The solid line marked with diamond is plotted by using best results which we measured by emulating all of the probabilities of parameter setting, with the aim of minimizing the end-to-end latency of message delivery. Clearly, UDR with adaptive setting very closely follows the UDR with measured optimal setting. The SGR incurs higher latency as the system size increases. Based on the statistics of experimental results, we find that in all experimented systems, our adaptive solution can yield better results with a mean improvement ratio of 10%, a maximal improvement ratio of 83.98% in terms of end-to-end latency.

We also examine the effect of system size on the system performance by using the metric of utilization ratio. Utilization ratio refers to the ratio of the number of shorter delay routing path detected by using our adaptive routing to the total number of routing measured in simulation. Each simulation is repeated 1,000 rounds on different source and destination pairs. To make them comparable, the same source-destination pair is used in each round. Fig.16 shows that up to 34% of routing paths have been improved among nodes in the system with 8000 nodes by using our adaptive solution.

D.GeoCast Adaption With the same setting of the previous experiment, we evaluate our scheme on a network of 8,000 nodes and the latency of the selected links is dynamically changed in the different ranges associated to the link type. Fig.18 depicts the end-to-end latency for discretization scheme with different parameter ρ ranging from 0 to 150 over the runtime. If ρ is set to 0 ms, the scheme is viewed as a special case of our UDR scheme, in which the mechanism of
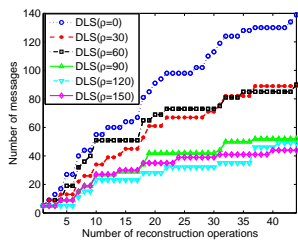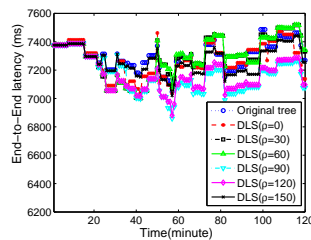
Fig. 17.   Adaption overhead

Fig. 18.   Effect of discretization parameter

multicast tree reconstruction is triggered whenever the change of network is detected. In Fig.18, we observe that the schemes with $\rho > 0$ improve the performance of the multicast trees, and consequently reduces the end-to-end latency in the presence of network dynamics. It is interesting to note that they all have similar tendency to react the changes of network. This is because only a small part of branches are being rebuilt at runtime while the majority of branches remained in the multicast trees do not have any change. We also find that after ρ reaches 90, increasing the value of parameter further does not achieve dramatic improvement in terms of end-to-end latency.

Comparing the cost of tree reconstruction for the UDR scheme with different parameter setting, we argue that after about 30 rounds, the reconstruction cost converge to a stable state with setting of $\rho = 90$, as shown in Figure 17.

VII. CONCLUSION

We have presented a utility driven routing scheme, which is unique in two aspects. First, it improves existing CAN-like geo-distance routing and existing proximity based routing protocols by carefully combining shortcut, geo-distance metric, with link latency metric in the message forwarding path selection process. Second, a utility function is designed by using a tunable influence parameter to provide an adaptive way for nodes to make the near-optimal routing decision with respect to their specific circumstances and network scenarios. Our experiments show that the utility driven routing scheme is scalable and latency efficient for large scale multicast applications compared to existing routing protocols. In our next work, further studies on the evaluation of the utility driven routing (UDR) scheme in the wide area network will be conducted.

Finally, we would like to point out that the main ideas behind our UDR scheme can be applied to both CAN-like DHT networks and other DHT overlays such as Chord-like networks [1]. Furthermore, the multicast optimization methods developed in [3][4][7] can deliver even more performance advantages by running on top of GeoCast powered by UDR compared to other DHT overlay networks.

VIII. ACKNOWLEDGMENT

REFERENCES

[1] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Transactions on networking*, vol. 11, no. 1, pp. 17–32, 2003.

[2] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron, "Scribe: a large-scale and decentralized application-level multicast infrastructure," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, pp. 1489–1499, 2002.

[3] J. Zhang, L. Liu, C. Pu, and M. Ammar, "Reliable peer-to-peer end system multicasting through replication," in *Proc. of P2P'04*.

[4] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: High-bandwidth multicast in cooperative environments," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 298–313, 2003.

[5] D. Kostic, A. Rodrigues, J. Albrecht, and A. Vahdat, "Bullet: high bandwidth data dissemination using an overlay mesh," in *Proc. of SOSP'03*.

[6] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *Proc. of SIGCOMM'02*, vol. 32, no. 4, pp. 205–217.

[7] X. Zhang, J. Liu, B. Li, and T. Yum, "Coolstreaming/donet: A data-driven overlay network for efficient live media streaming," in *Proc. of INFOCOM'05*, vol. 3.

[8] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. Mohr, "Chainsaw: Eliminating trees from overlay multicast," *Proc. of IPTPS'05*.

[9] S. G. S. R. S. S. K. Gummadi, R. Gummadi and I. Stoica, "The impact of DHT routing geometry on resilience and proximity," in *Proc. of SIGCOMM'03*.

[10] F. Dabek, J. Li, E. Sit, J. Robertson, M. Kaashoek, and R. Morris, "Designing a DHT for low latency and high throughput," in *Proc. of NSDI'04*.

[11] M. Castro, P. Druschel, Y. Hu, and A. Rowstron, "Exploiting network proximity in peer-to-peer overlay networks," in *Tech. Rep. MSR-TR-2002-82, Microsoft Research*.

[12] S. Ren, L. Guo, S. Jiang, and X. Zhang, "SAT-Match: a self-adaptive topology matching method to achieve low lookup latency in structured P2P overlay networks," in *Proc. of IPDPS'04*.

[13] Z. Xu and Z. Zhang, "Building low-maintenance expressways for p2p systems," *Hewlett-Packard Labs, Tech. Rep. HPL-2002-41*.

[14] J. Zhang, G. Zhang, and L. Liu, "GeoGrid: A Scalable Location Service Network," in *Proc. of ICDCS'07*.

[15] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A scalable content-addressable network," in *Proc. of SIGCOMM'01*, pp. 161–172.

[16] Y.h. Wang, L. liu, C. Pu and G. Zhang, "GeoCast: An Efficient Overlay System for Multicast Application," *Tech. Rep., GIT-CERCS-09-14, Georgia Tech*.

[17] C. Gkantsidis, M. Mihail, and A. Saberi, "Random walks in peer-to-peer networks," in *Proc. of INFOCOM'04*, vol. 1.

[18] F. Dabek, M. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Wide-area cooperative storage with CFS," in *ACM SIGOPS Operating Systems Review*, 2001.

[19] Y. Chu, S. Rao, S. Seshan, and H. Zhang, "Enabling conferencing applications on the internet using an overlay muilticast architecture," in *Proc. of SIGCOMM'01*, vol. 31, pp. 55–67.