Mobile Netw Appl
DOI 10.1007/s11036-011-0348-9

Scaling Group Communication Services with Self-adaptive and Utility-driven Message Routing

Yuehua Wang · Ling Liu · Calton Pu

Abstract Group communication services typically generate large multicast data streams. Delivering such massive data streams to the end system nodes at the edge of the Internet has been a challenging problem in terms of high stress on the network links and high demand on network resources and routing node capacities. Most of existing research has been dedicated on geo-distance based routing with various optimizations to alleviate the performance impact on geo-distance based routing due to unpredictable network dynamics. Most representative techniques are targeted at reducing the delivery path length or optimizing routing path by utilizing network locality. In this paper, we identify the inefficiency of geo-distance based routing protocols in many existing multicast overlay networks in terms of both resource utilization and group communication efficiency. To address this issue, we develop a utility-based routing scheme (UDR) that can provide efficient group communication services in a decentralized geographical overlay network. Our approach makes three unique contributions. First, we introduce a utility function to refine the geo-distance based routing in such a way that the routing path selection can carefully incorporate both geo-distance based metric and the network latency. Second, we enhance our utility driven routing scheme with self-adaptive capability by considering the nodes' state and network density. Thus, nodes in the multicast network can dynamically accommodate the changes of network conditions based solely on their local knowledge about the network. Third, we devise a suite of optimization techniques to minimize the maintenance cost and computational complexity of our self-adaptive and utility-drive routing scheme. We evaluate our approach through extensive experiments based on a realistic network topology model and show that the UDR method is highly scalable and it effectively enhances the multicast delivery efficiency for large scale group communication services compared to existing geo-distance based routing protocols.

Keywords distributed networks · utility function · self-adaptive routing · group communication services

Y. Wang (✉) · L. Liu · C. Pu
College of Computing, Georgia Institute of Technology, Atlanta, USA
e-mail: yuehua.wang1981@gmail.com

L. Liu
e-mail: lingliu@cc.gatech.edu

C. Pu
e-mail: calton.pu@cc.gatech.edu

Y. Wang
State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing, China

1 Introduction

The rapid growth of wireless communication technology and increasing popularity of hand-held devices are continuously escalating multi-party group communication applications, such as multiplayer online games, proximity based advertising, real-time conference, IPTV, and instant messaging. Typically, in those applications, a large number of participants who are widely distributed, loosely coupled, and inherently

Published online: 07 January 2012

unreliable often access the services and require communicating with each other regardless of their locations.

To fulfill the voluminous demand of participants, a considerable amount of work on designing group communication systems has reported in recent years. Examples include Splitstream [1], Bullet [2], NICE [3], Scribe [4], Peercast [5], Coolstream [6], ESM [7], Overcast [8], TAG [9] and Chainsaw [11]. All these systems have shown three important observations.

First, for distributed wide-area group communication services that depend on the services of overlay networks for operation, the properties of overlay networks (e.g., the communication efficiency, system scalability and fault resilience), greatly dominate their performance. Given node arrivals and departures incurred either by network partition or node failures are common cases in a dynamic network environment, it is essential for the overlay network to adapt to the changes of network conditions and node states.

Second, majority group communication systems rely on the construction of the multicast tree for data dissemination and the efficiency of the multicast tree heavily depends on the efficiency of the routing protocol provided by each specific overlay network. However, most of existing work has dedicated 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.

Third, most of the overlay network topology does not match well with the underlying IP network topology, and thus the messages communicated among nodes may have to traverse the IP network from one end to another multiple times to reach their destinations. This is particularly true when the group communication participants are widely distributed across the Internet. Therefore, it is essential for the group communication services to develop an efficient way to deliver message among the participants by taking into account the routing path length and path latency.

In this paper, we articulate that link latency is an important indicator of the network conditions and node states, especially when the nodes are widely distributed across the network and lack global knowledge about the network. Given that the geo-distance based routing protocols used in the existing overlay networks are inefficient in terms of both resource usage and environmental accommodation. We consider the link latency in the message routing path selection for supporting the applications with group communication services, and develop a scheme named utility driven routing (UDR) to improve the efficiency of geo-distance based routing protocols and enhance the performance of applications. Such scheme is highlighted with three features.

- UDR refines the geo-distance based routing and a utility function is defined based on a careful combination of geo-distance and network latency.
- We enhance our utility driven routing scheme with self-adaptive capability by considering the nodes' state and network density. Specifically, each message's specific situation at nodes is taken into account so as to determine the most promising routing path. Thus, nodes in the multicast network can dynamically accommodate the changes of network conditions based solely on their local knowledge about the network
- The performance of UDR is further optimized by three novel optimization techniques.

We demonstrate the effectiveness of our routing protocols by applying to GeoCast, a CAN [12] like geographical overlay network prototype for group communication built on top of GeoGrid [13]. Such geographical overlay networks have an important property attractive to group communication applications (e.g., location-based advertisement applications, IPTV applications, and online gaming applications). Specifically, the use of geographical proximity of nodes in the networks enables the nodes to reside close to their physical network neighbors in the overlay network with high probability so that the communication cost caused by the topology mismatching and the link stress impressed by the overlay communication can be reduced.

The rest of the paper is organized as follows. Section 2 briefly reviews the related work in link latency-aware routing scheme. Section 3 describes the structure of GeoCast system and the motivation of the utility-driven routing scheme. In Section 4, we give the definition of the utility function, discuss the setting of the tunable influence parameter, and present the detail of the UDR algorithm. Three optimization techniques are discussed in Section 5 to further improve the performance of UDR. Section 6 presents the experimental evaluation. We conclude the paper in Section 7 with a summary.

2 Related work

A fair number of works has been focused on providing an efficient routing scheme for disseminating message among nodes in the overlay networks. Examples include Chord [14], Pastry [15], Tapestry [16], CAN [12], and Kademlia [17]).

In Chord, each node maintains $logN$ fingers in its table. With a message and a key, a Chord node can efficiently route the message to the finger node with an

identifier that is numerically closest to the key among all currently live finger nodes. It ensures a message can be routed to the destination in $logN$ hops because each hop cuts the distance to the destination in half. Each node in the Pastry network has an unique identifier. It keeps track of its immediate neighbors in Pastry and maintains routing tables with about $O(logN)$ entries. Similar to Chord, Pastry ensures that the message can research destination network within $O(logN)$ steps. However, since the next hop is unique and deterministic on each step, both Chord and Pastry can not work well in a network of nodes with high churn rate. The upper-level services are prune to be interrupted in such a situation.

In contrast to Pastry, Chord, and Tapestry, CAN offers higher flexibility in routing selection. Each node along the routing path has multiple next hop choices. In a network of d-dimension CAN can route the message to any node in $O(dN^{1/d})$ routing hops. However, we argue that CAN can result in a relatively longer routing path than other DHTs in terms of hop count when d is small, which could potentially involve significant delays in message routing.

Kademlia is proposed based on the XOR-based metric topology. Each node is identified by a number or node ID. It provides a direct map to the file hashes and shows the information where to fetch the file or resource. Like many other DHTs, Kademlia contacts only $O(log(n))$ nodes during the search out of a total of n nodes in the system.

While the hop count is an important metric for measuring the processing at the peers, it does not adequately address the problem of topology mismatching. In some cases, one overlay hop could involve high delay in the message transmission. Therefore, 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. To deal with that, many research efforts [12, 15, 18, 19, 21] have been made to improve the efficiency of DHT routing protocols. Generally, the existing literature can be broadly classified as three categories: proximity neighbor selection (PNS), proximity route selection (PRS) and proximity identifier selection (PIS).

PNS exploits the flexibility in the choice of nodeIds to fill routing table slots. The idea is to route the message to the closest node in the underlying network from among those whose nodeIds have the required prefix [15, 19, 21]. For a given node p, the closest node refers to the node that has the shortest propagation delay to node p among the nodes in p's routing table. PNS is appropriate since it can achieve both low delay message transmission and low bandwidth usage. However, it is often costly to identify the closest nodes for the nodes in a larger system since the size of the subsets grows exponentially with the system size.

In PRS, nodes are dedicated to minimizing the hop routing latency under the constraint that each hop should be closer to the destination. In [12, 22], the next hop is the one out of all possible neighbors that results in the maximum progress towards the destination, where the progress is defined as the ratio of the distance in identifier space to the physical latency. Similarly, [23] devises a network-delay routing (NDR) to minimize the routing latency among nodes. In the process of message routing, the neighbor with least network delay is favored to be chosen as the next hop, which subjects to the constraint that the message moves closer to the destination on each hop. However, since the naive routing approach may lead to a long routing path in terms of hop count, the end-to-end latency of message routing could be higher than that of the DHT routing schemes. Gummadi et al. [18] conducts a comparative study of two approaches PNS and PRS in a variety of overlay networks. The experimental results show that PNS is significantly more effective than PRS in dealing with the nodes' proximity. However, we argue that PNS may become inefficiency when applying to the topology aware overlay networks [28]. Given that, we adopt the method of PRS, and one of our key technical contributions is the introduction of utility function, which allows the nodes to make the most optimal routing selection with regards to the specific condition of each message. Instead of routing the message to the node linked with the minimum routing latency, our approach takes the distance to the destination node and local network density into account so that the length of message routing path and the latency of message transmission in PRS are reduced.

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 [5, 24, 25]. Ren et al. [24] uses the metric of latency to adaptively construct a structured P2P overlay network. Different with [24], *landmark ordering* [5, 25] is to address the problem by fixing a well known set of nodes that act as landmarks on the Internet. The rationales behind these schemes are that topologically close node are likely to have the same ordering and hence will belong to the same slot. But how to place those landmarks in an efficient way is still an open problem nowadays. In addition, the PIS, though has potential to reduce the link latency between the nodes in the overlay network, has one main drawback in load balancing, where nodes in PIS may be

imposed on heavy load when their contents are highly desired by users. In such a situation, a great number of the messages in PIS might be delayed which may lead to significant degradation of system performance. The approaches presented in this paper differ from these in two folds. Firstly, the geographical proximity of the nodes is exploited to construct the structure of GeoCast. It ensures the neighbor nodes that are topologically nearby on the underlying IP network tend to locate closely in the overly network. We will describe it in detail in the following section. Secondly, as one important indicator of the network conditions and node states, the metric of link latency is introduced in message routing path selection so as to accommodate the changes of network conditions and minimize the message routing latency.

3 Basic GeoCast system

GeoCast is a geographical overlay system built on top of GeoGrid [13] for providing scalable group communication services. In GeoCast, each node is equipped with a GeoCast middleware, composing of two substrates: *overlay network management* and *end system node multicast management.*

3.1 Overlay network management

The overlay network management substrate is the lower tier of the GeoCast middleware. It provides services such as overlay membership management, resource lookup, and communication among nodes. This substrate incorporates two protocols, namely *membership protocol* and *routing lookup protocol*.

3.1.1 Membership protocol

GeoCast system uses this protocol to organize the widely distributed end system nodes into an overlay network that provides group communications services.

In GeoCast, each node represents as a point in a two dimensional geographical coordinate space called G and owns a rectangular region that contains that point. A node E_i is described as a tuple of five attributes, denoted by $E_i : (x, y, R, IP\_Port, capacity)\ \ i \in [1, n]$, where (x, y) refers to the unique identifier of end system node E_i that can be generated based upon the node's IP address or geographical location, and n is the number of the nodes in the system. R denotes the region that node E manages. It is described as a quadruple: (x, y, w, h), where (x, y) represents the coordinates of top left vertex of region and (w, h) refers to the width and height of R. The relationship between R and 2D geographical coordinate space G subjects to $\sum_i^n E_i.R.w \times E_i.R.h = G.w \times G.h$, where $G.w$ and $G.h$ denote the width and height of 2D geographical coordinate space respectively. In GeoCast, two nodes are considered to be immediate neighbors when their intersection is a line segment. IP_Port is node E_i's IP address and port used to communicate with other network nodes. *Capacity* refers to the specific attribute of E_i, which quantifies the amount of resource of that node E_i is willing to dedicate for serving other nodes. It may represent CPU power, memory, bandwidth and so on. In GeoCast, we use it to denote the available bandwidth of a node.

Similar to GeoGrid, GeoCast is constructed incrementally. It starts with one node who owns the entire GeoCast space. As a new node joins the system, it uses its own unique identifier to map itself to a rectangular region of the GeoCast, which corresponds to a geographical area in which it physically resides. After identifying the region to which the new node belongs, the owner node of region checks if its *unit_capacity* is smaller than that of its immediate neighbors, where *unit_capacity* represents the workload capacity per region unit, defined by: $unit\_capacity = \frac{E_i.capacity}{E_i.R.w \times E_i.R.h}$. $E_i.R.w$ and $E_i.R.h$ denote the width and height of region R owned by E_i respectively. If so, it splits the region in half, hands one half to the new node, and notifies its immediate neighbor nodes of the arrival of the new node so that the new node can become one part of the system. Otherwise, the owner node of region checks its immediate neighbors and selects the node with lowest *unit_capacity* to partition its region in half by following a certain ordering of the dimensions such as latitude dimension first and then longitude dimension. Readers may refer to our technical report [28] for detailed construction process and examples of node arrival and departure.

3.1.2 Routing lookup protocol

In the basic GeoCast, we use *geo-distance based routing with shortcut* (called SGD) to deliver messages for end system nodes. Inspired by the analysis [14, 15, 25], we employ the concept of *shortcut* to speed up the message routing procedure. The main idea is to maintain more information about the nodes in the system on each node such that these nodes can be used as the shortcuts in the procedures of message routing. In GeoCast, the term *shortcut node* refers to the node which is an *old immediate neighbor node* for a given node. As nodes arrive or depart, the immediate neighbor nodes may

become shortcut nodes when they are not adjacent to the given node.

Each node in the basic GeoCast maintains a list *peernodelist*. Such list contains two kinds of nodes: immediate neighbor node and shortcut node. Instead of removing old immediate neighbors from the lists, GeoCast keeps those nodes in *peernodelists* and uses them to speed up the procedures of message delivery. For a node E_i, its *peernodelist* is 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 $(1 \le j \le Q)$, it contains the nodes in the geographical enclosing zone EZ_i^j with the size of $1/2^j$ of the geographical plane G. EZ_i^j is defined by: $EZ_i^j :< x, y, w, h >$, where (x, y) represents the coordinates of top left vertex of enclosing zone and (w, h) refers 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 likely that a shortcut node in the subset S_i^k be chosen as the next message forwarding hop.

Figure 1a provides a snapshot of GeoCast overlay network with three end system nodes. At this point, there is no shortcut node included in the *peernodelists* of the nodes. For E_2, $peernodelist(2) = \{S_2^0\} = \{\{E_1, E_3\}\}$, where node E_1 and node E_3 are its neighbors included in EZ_2^0 that is identical to the entire plane G. With the arrival of node E_4 as shown in Fig. 1b, node E_3 is no longer a neighbor of node E_2 and is now recorded as a shortcut node in $peernodelist(2)$. For node E_2, shortcuts are the nodes whose regions have no intersection with region 2. Given the locations of nodes, S_2^0 is represented by: $S_2^0 = \{E_1, E_3, E_4\}$, where $E_1, E_3, E_4 \in EZ_2^0$. Similarly, node E_4 becomes another shortcut node of node E_2 after node E_5 joins the network as shown in Fig. 1c. Now $peernodelist(2) = \{\{E_1, E_3, E_4\}, \{E_5\}\}$, where $E_5 \in EZ_2^1$ and $E_5 \notin EZ_2^0$. In this way, each node keeps building up its *peernodelist* as the topology of network evolves with the arrival or departure of nodes.

Different nodes may have their *peernodelist* in different sizes. The exact length of the list for a node E_j is decided by the relative size of the region R owned by E_j. If the region R is $1/2^Q$ of the size of the geographical plane, the list length is Q. This allows to cover the entire geographical plane by the shortcuts of E_j according to the following equation: $\sum_{i=1}^{Q} 1/2^i + 1/2^Q = 1$ (The full version of the proof can be found in our technical report [28]).

Let $Gdist_{i \to j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ be the geo-distance between two nodes E_i and E_j on the space G, where (x_i, y_i) and (x_j, y_j) are the unique identifier of E_i and E_j respectively. Let destination be a point or a region in the space G [13]. In GeoCast, we consider the destination as a spatial point in the plane and tag its coordinates with a request message, consisting of a spatial query point D(x, y), a description of request data, information of the node who issued such request. When the end system node that covers D(x, y) receives this request, it sends the requested information related to its region back to the request sender. In location-based service (LBS), an example of such request is "send me the current traffic state of I-85 on exit 94."

When a node E_i 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 smallest *Gdist* to the destination as its next routing hop to route the message. This procedure is executed repeatedly until the message reaches the node whose region covers the destination. In SGD, the use of shortcuts ensures that at each hop, the routing search space can be reduced

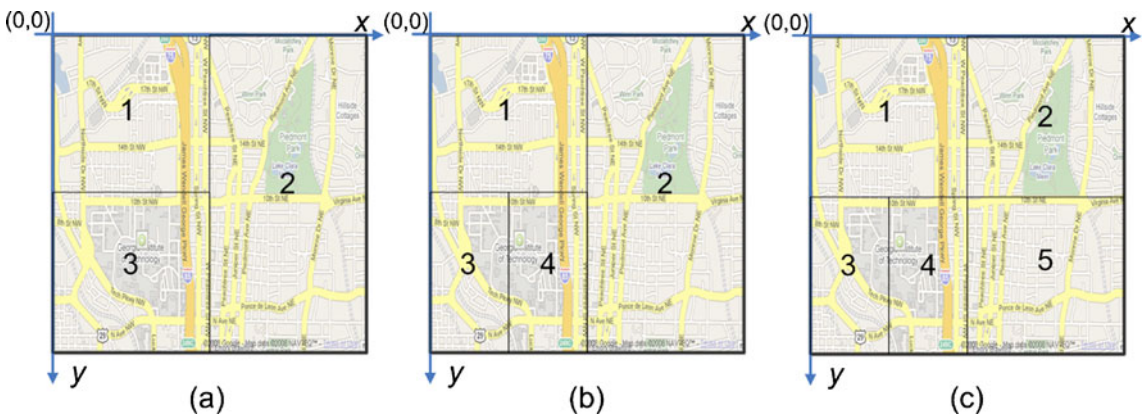


Fig. 1 A GeoCast overlay network

by half. Any node in the system can be reached in $O(log_2 N)$ hops.

3.2 Multicast management

This substrate is the higher layer of the middleware, being responsible for multicast service publication, subscription management, multicast information delivery and group membership management. This layer utilizes the services provided by the overlay network management layer and provides group communication services for the nodes with the same interests.

In general, there are four basic operations for multicast service establishment and maintenance: *multicast service announcement*, *tree-based multicast service subscription and un-subscription*, *multicast information dissemination*, and *multicast group management*.

Multicast service announcement This operation is to enable nodes to create new multicast services and serve the nodes with interests in the announced services. In GeoCast, any node is allowed to use this operation to become a service publisher who is responsible for announcing its information and providing services for the other nodes in the system. Each service publisher 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 multicast service subscription and un-subscription This operation is to let the nodes subscribe (unsubscribe) to the existing multicast groups. Nodes in GeoCast allow to subscribe (unsubscribe) to any multicast service published in the network by initializing a subscription (unsubscription) requests. Such kind of request may be routed either to a node that has already subscribed to the multicast service of interest or to the publisher.

Multicast information dissemination The publishers use this operation to deliver the multicast data to all the content demanding nodes (called subscriber). They inject the data at the root of the multicast trees that are formed by the routes from the subscribers to the publishers.

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

The goal for this layer establishment is two-fold. The first one is to release high link stress caused by message transitions among nodes in the group communication sessions and consequently improve network resource utilization. The second one is to reduce the delay of the message delivery from the publishers to the subscribers. Instead of transmitting the data from the publishers, the subscribers often get it from their parent nodes within a short delay. For each session, there exists a spanning tree that is an acyclic overlay connecting all the participants of the session. It is used by the publisher node for content dissemination.

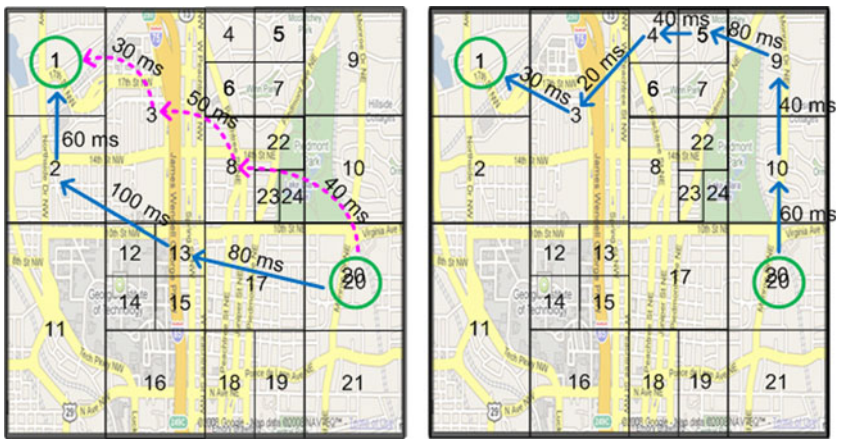
3.3 Motivation for utility-driven routing

In GeoCast, each node has an average of $O(2d)$ neighbors and $O(logN)$ shortcuts maintained in its *peer-nodelist*, 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 based routing with shortcut ensures that any node in the system can be reached in less than $O(logN)$ hops, achieving similar performance to Chord [14] and Expressway [25].

Figure 2 illustrates the differences of the routing schemes using an example. Given source S and destination D in a system of 24 nodes. The scheme of SGD only needs 3 hops to reach node D (see solid line in Fig. 2b) whereas CAN-like geo-distance based routing needs two times as many hops as that of SGD, as shown in Fig. 2a.

However, we argue that the path formed by the SGD might be a long forwarding path in terms of network latency, which leads to inefficient routing performance. Consider the example shown in Fig. 2b. The geo-distance based routing with shortcut takes 240 ms to deliver the message to the destination (see solid line in Fig. 2b). In contrast, the dashed line shown in Fig. 2b gives a faster routing path by incorporating network latency in routing selection algorithm, only 120 ms. This is due to that the link latency in the first two hops of the shortest geo-distance routing path that are relatively high with up to 100 ms.

Given that the SGD might not necessarily deliver the shortest network latency among all the alternate routing paths. One may suggest that an efficient routing algorithm might be the one that take link latency



(a) (b)

Fig. 2 Examples to illustrate CAN-like geo-distance based routing and geo-distance based routing with shortcut

into consideration. Clearly if we only use link latency as the metric to choose the forwarding node and the routing path for message delivery, then the messages are routed to the node whose connecting link to the source node has the minimum latency. However, this naive approach may lead to a long routing path in terms of hop counts since the forwarding nodes in this case are the neighbor nodes. Similar to the CAN-like geo-distance based routing, it may take at most $O(dN^{\frac{1}{d}})$ hops to reach the destination.

Though introducing latency metric for the routing algorithm as suggested by previous studies [12, 22] is insufficient to short the long message transmission latency, we are motivated to design a routing algorithm that combines the geo-distance and link latency carefully for the purpose of optimizing the routing performance, Therefore, 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). To this end, our utility-driven routing (UDR) protocol is proposed.

4 Utility driven routing

In this section, we describe the design of the utility-driven routing protocol. We begin the section with a definition of utility function introduced by UDR protocol and method of how the parameters are setting is then introduced after the definition. At last, we present the details of the UDR protocol. For reference convenience, we refer to the version of GeoCast powered by UDR as *enhanced GeoCast* compared to the basic GeoCast mentioned in Section 3.

4.1 Utility function definition

To make the routing algorithm practical, the first question we need to answer is how to best combine geo-distance and link latency so that the nodes in the system can assess their needs and make the decisions based only on its local knowledge about the network. In our work, a utility function is introduced by taking into account the link latency and the geo-distance from the next hop to the destination, which is defined as:

$$
\begin{aligned}
GDV_i &= (1-\mu) * f_{\text{la}}(i) \; + \; \mu * \eta * f_{\text{dis}}(i), s.t. \\
f_{\text{la}}(i) &= \frac{RTT_{\rightarrow i}}{2} \\
f_{\text{dis}}(i) &= Gdist_{i \rightarrow D}
\end{aligned} \tag{1}
$$

where μ is a tunable influence parameter, ranging from zero to one. It is used to adjust the importance of *geo-distance factor* f_{dis}, and *latency factor* f_{la}. f_{dis} represents the geo-distance from the next hop node E_i to the destination D. $f_{\text{la}}(i)$ represents the delay of link between the current node and the next hop node E_i. Note $RTT_{\rightarrow i}$ is 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 $f_{\text{la}}(i) = RTT_{\rightarrow i}/2$. In GeoCast, every node treats $f_{\text{la}}(i)$ as soft state. Every T seconds, $f_{\text{la}}(i)$ is updated via heartbeat message, where T is a system parameter configured by default. η is a *normalization parameter* designed to unify the dimensions of the two factors, defined by: $\eta = RTT_{\text{avg}_{\text{dis}}} / (2 * avg_{\text{dis}})$, where avg_{dis} and $RTT_{\text{avg}_{\text{dis}}}$ denote the average geo-distance among the immediate neighbor nodes and its associated average round trip time (RTT) respectively.

Based on Eq. 1, every node along the routing path calculates the utility value for the entry nodes in its *peernodelist*. For each entry node, the calculated utility

value refers to the qualification of the node to be a forwarding node for a message with source node S and destination node D. The larger it is, the more likely the message will be routed to the node. In the following section, we will discuss the setting of parameter μ in the utility function before introducing the proposed routing protocol in detail given the importance of influence parameter μ.

4.2 Setting of parameter μ

Note that the decision on how to set μ involves a trade-off between the importance of *latency factor* and *distance factor*. Here, we use an example to illustrate the significance of selecting good value for the influence parameter μ.

Figure 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 $\{E_3, E_5, E_6, E_{11}, E_{12}, E_{13}\}$ can be used as routing nodes 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. 3a. $Dist(S, D)$ 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 *peer nodelist*. We denote the distance between an entry in the *peernodelist* of S and 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 E_6 is selected to be the best node for message forwarding with the setting of $\mu = 0.8$ (see Fig. 3a) such that the nodes locating in the vicinity of source S have less probability to be selected due to their larger geo-distance to the destination node D. However, this setting of μ may not be appropriate when source node S resides closely to the destination node D as shown in Fig. 3b. While the setting of a larger μ is also not good choice since the link latency from S to node E_{12} is the worst (50 ms). To avoid that, we set μ small (say 0.2) and route the message to node E_6 through the link with lowest delay. However, such setting is no longer suitable when the source node is located in a densely populated area as shown in Fig. 3c and d, a partial enlargement of Fig. 3c. 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. 3d, node E_{21} is selected as the next forwarding node with such setting and the routing procedure repeats until the message reaches destination node D.

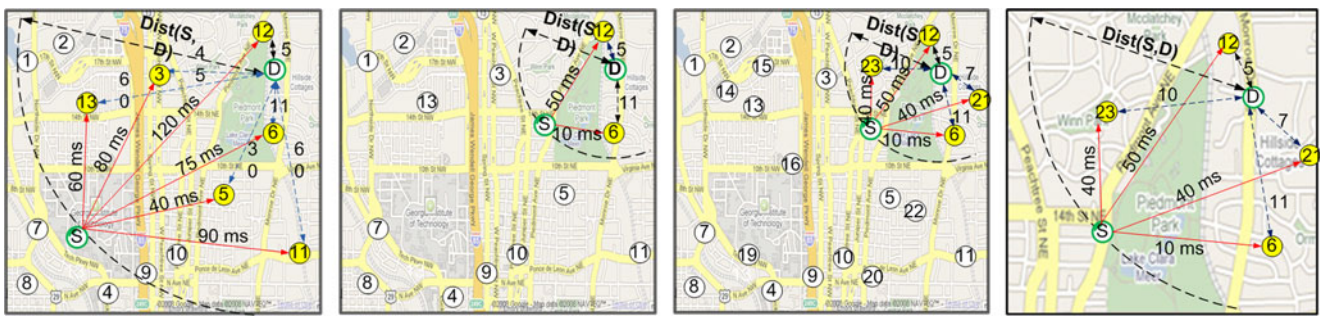
This example shows that the constant setting of the influence parameter μ is impractical in a highly dynamic environment where any two nodes may communicate with one another at any time and the system may have unpredictable churn rate. It is also interesting to observe that for any message, the number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

$$\mu = \frac{\sum_{j=0}^{Q} NS_j}{2Q} + \frac{1}{2^{Q-\tau-1}} \tag{2}$$

$$NS_j = \begin{cases} 1 \text{ if } \exists E_m \in S_j,\ f_{\text{dis}}(m) < f_{\text{dis}}(i) \\ 0 \text{ otherwise} \end{cases} \tag{3}$$

where $\sum_{j=0}^{Q} NS_j$ denotes the number of subsets in the *peernodelist* that contain the nodes locating closer to the destination than current node E_i. NS_j sets 1 when subset S_j satisfies: $\exists\ E_m \in S_j,\ f_{\text{dis}}(m) < f_{\text{dis}}(i)$. Q refers to the number of subsets kept in *peernodelist*(i) for a given node E_i. The larger the system is, the smaller the responsible region node has, the larger Q is. τ denotes



(a) (b) (c) (d)

Fig. 3 Significance of setting μ parameter adaptively

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 increases.

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 larger 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 smaller 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 larger network and consequently they might have less knowledge about the network. The factor of $1/2^{Q-\tau-1}$ may take a larger value when both Q and τ are small. In such 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. 3c, 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-distance (τ and Q are small).

4.3 Routing protocol design

In the enhanced GeoCast, a three-step procedure is carried out by the node when it hosts a message with specific destination.

Step 1: The node initiating the request message checks to see if it is the destination node. If so, it stops the routing. Otherwise, it continues to decide on which node to forward the massage next.

Step 2: Compute the geo-distance based filter for the current node and use it to generate a candidate list for node selection. This candidate list contains all nodes that are closer to the destination node than the current node in terms of geo-distance.

Step 3: Calculate the utility value for every node in the candidate list produced in the step 2. The message is then routed to the node with the smallest utility value (called forwarding node). When the message arrives at the forwarding node, this routing procedure repeats until the destination whose region contains the coordinates tagged with request message is reached.

Table 1 describes the process of forwarding node selection. In this process, all the nodes in candidatelist are collected by a geo-distance filter. Such filter is computed by using the geo-distance (*Gist*) between the current node *i* and the destination node as the filtering criterion to build the candidatelist from the peernodelist of the current node E_i (line 3∼7). In UDR, the filter is viewed as a circle function of variable radius, initialized by $dist_i$. As the algorithm proceeds, the current node is changed from node E_i to its forwarding node, and the circle shrinks since the geo-distance from the forwarding node to the destination node is getting smaller.

Note that the candidatelist of a node contains only the nodes that are both in its peernodelist and residing inside the associated circle of the node. In Step 3, the utility function is used to calculate and assign the utility value for each node in the candidatelist (line 9), and at each node along the routing path, the message is routed

Table 1 Pseudo code for forwarding node selection

Function: select forwarding node for the destination node p(destX, destY)	
1	candidatelist=null
1	min=Integer.Max,temp=null
2	dist=$Gist_{i\to p}$
3	for(node: peernodelist)
4	if ($dist_{node} < dist$)
5	add node to candidatelist
6	end
7	end
8	for(node : candidatelist)
9	val=the utility value of the node
10	if ($val < min$)
11	min=val
12	temp=node
13	end
14	end
25	return temp

to the node in the candidatelist with the smallest utility value.

We illustrate this routing algorithm through an example in Fig. 4. To deliver a message from S to D, node S first looks through all the entries in its list: $peernodelist(12) = \{E_1, E_2, E_3, E_9, E_{10}, E_{13}, E_{14}, E_{15}, E_{16}\}$. Using the filter of radius defined by the geo-distance between S and D, we discard all nodes that are outside the circle of this radius: E_1, E_{14}, E_{15} and E_{16}, which are far away from the D than S. Instead of forwarding to node E_3, the node closest to the destination node D in terms of geo-distance, the UDR chooses E_{10} after comparing the utility values of nodes E_2, E_3, E_9, and E_{10}. After the message routed to E_{10}, the similar routing process is executed at node E_4. E_7, E_{10}, E_{11} and E_{12} are filtered out and the message is then routed to E_5. This procedure is executed iteratively until the message arrives at D.

In the enhanced GeoCast of N nodes, each nodes maintain information about $O(logN)$ other nodes in its *peernodelist* and a lookup requires $O(\log N)$ messages. By properly setting the influence parameter μ, UDR optimizes the routing performance by ensuring that the routing path length is less than or equals to O(log N) hops even in the worst case of $\mu = 0$. In GeoCast with 8,000 end system nodes, only 5 hops are needed to reach the destination node on average. It outperforms the CAN-like geo-distance based routing [12], achieving almost same performance of Pastry, Tapestry, Chord and Expressway. This benefits greatly from the geo-distance filter and utility function. Comparing to existing link latency-aware approaches, our approach to tune the influence parameter μ in the utility function is unique and highly effective because it allows every 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 diversity of the locations of the source and destination.

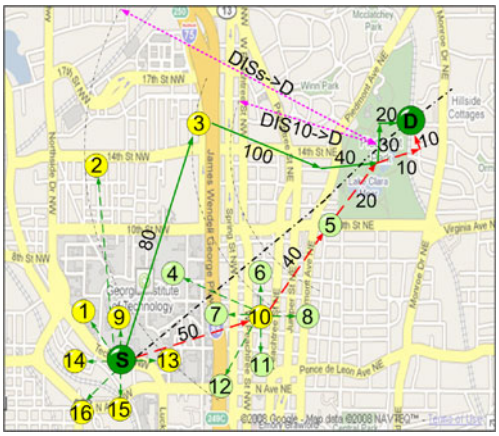


Fig. 4 Utility-driven routing

In UDR, each node along the routing path needs to take about $O(2\log N + \log(\log N))$ operations to make the routing decision before forwarding the message to the forwarding node. The routing decision is based on enumerating all entries and calculating the utility value for each entry in the candidatelist of the message host node. In the worst case, the number of operations can be estimated as $O(2\log^2 N + \log N \times \log(\log N))$, with its routing performance of $O(\log N)$. However, in many cases, the computational complexity of UDR is much less than $O(2\log^2 N + \log N \times \log(\log N))$ due to the geo-distance filter. As the forwarding node is closer to the destination, the number of nodes in the candidatelist is getting smaller, and only a few calculations are required.

Similar to the basic GeoCast, mutlicast tree in the enhanced GeoCast is also formed by joining routes from group subscribers to the root node. By taking advantage of UDR routing, our tree construction scheme ensures that join requests take at most $O(logN)$ steps to reach their roots, resulting in a relatively constant depth of multicast tree in the enhanced GeoCast, which makes it scale well in terms of the overall size of the network.

5 Performance optimizations

In this section, we introduce three optimization techniques that we have implemented in the enhanced GeoCast to optimize the performance of the UDR algorithm. They are shortcut clustering, top-k routing, and automated latency scaling for coping with network dynamics.

- Shortcut clustering. Given that the shortcut are redundant in some sense, node clustering algorithm is introduced as an optimization technique to minimize the maintenance cost in the presence of high churn rates. It offers the nodes with the ability to choose those nodes with higher capacity to handle more routing workload.
- Top-k routing. It is proposed to further reduce the computational complexity of the UDR routing algorithm based on the observation that the node with shorter distance to the destination is favored as forwarding node, while the nodes located

somewhat far away from the destination might be useless for routing and delivering the message.
– Automated latency scaling for coping with network dynamics. It focuses on reducing the impact of network dynamics, especially the oscillation of link latency on the stability of multicast trees in GeoCast.

5.1 Shortcut clustering

The idea behind the shortcut clustering optimization is to maintain "enough" shortcuts for each node instead of all shortcuts in addition to the immediate neighbor nodes. It offers the nodes with ability to keep those shortcut nodes that has higher capacity and thus can handle more routing workload and also to reduce the shortcuts that are representatives for the same enclosing zone.

In the enhanced GeoCast, we introduce a system defined parameter m for shortcut clustering. The setting of m allows us to limit the size of each *peernodelist* maintained in the system [14, 15]. Initially, there is no shortcut node dropped before the size of *peernodelist* reaches m. Each node builds up its own *peernodelist* as a new node joins or leaves the network. The shortcut clustering algorithm is triggered at the node p once the size of the node p's *peernodelist* exceeds m. To meet the requirement, the following steps are performed:

Step 1: Count the number of shortcut nodes contained in the node p's *peernodelist*, and assign the value to a variable *totalnum*.

Step 2: Create a new empty list *templist*.

Step 3: Check if $m \geq Q$. If so, go to Step 4. Otherwise, go to Step 7.

Step 4: Node p looks though the *peernodelist* and removes S_i to *templist* if $S_i.size = 1, \forall i = 1, 2, \cdots, Q$. After moving, there are $totalnum - templist.size$ shortcuts in the *peernodelist*, where *templist.size* refers to the number of shortcuts in *templist*.

Step 5: Denote Num^i_{del} to the estimated number of shortcut nodes that should be removed from subset S_i, and compute Num^i_{del} for S_i, by using the following equation:

$$Num^i_{\text{del}} = \frac{i * (totalnum - m)}{\sum_{k=1}^{Q+1} k * b_i}, \quad \forall\, k = 1, \cdots, Q \tag{4}$$

where level Q is formed by latest split. b_i is set to 1 if S_i is in the *peernodelist*. Otherwise, b_i is set to 0.

Step 6: If $Num^i_{\text{del}} \geq 1$, compare the *unit_capacity* of shortcuts in S_i, and move the nodes with less *unit_capacity* from S_i. Otherwise, remain one node with highest *unit_capacity* in the subset and discard other nodes from the subset in the descending order of Num^i_{del}. Then, go to Step 10.

Step 7: Denote P^i to be the probability of shortcut node that should be removed from subset S_i, and compute P^i for each node by the following equation:

$$P^i = \frac{i}{\sum_{k=1}^{Q+1} k * b_i}, \; quad \forall\, k = 1, \cdots, Q \tag{5}$$

Step 8: Remove the shortcuts in *peernodelist* into the *templist* and group the nodes into m clusters in terms of the geo-distance to node p [30, 33].

Step 9: For each cluster, select the shortcut with highest P^i to be the representative of the cluster, and remove the other nodes from the cluster.

Step 10: Add the subsets in the *templist* to the *peernodelist*, and terminate this procedure.

Based on Eq. 4, the subset S_i that locates far away from the node p is assigned to a smaller Num^i_{del}. It allows the shortcut nodes in S_i to have high probability to be reserved in the *peernodelist* than that of nodes in subsets that reside in the adjacency area of node p. In such a way, the redundancy of shortcuts can be eliminated without degrading the routing performance.

Generally, it is possible that the value of m is smaller than Q. In such a case, all the shortcuts at each level are about to be considered for node filtering. By utilizing the value of $Gdist_{\rightarrow p}$, the shortcut nodes are dynamically classified into m clusters. And the node with highest P_i in each cluster is selected and maintained in the *templist*, as described in step 9. Intuitively, a node with more *unit_capacity* should be preferred to remain in each cluster. If there are more than one shortcut node with highest P^i, the one with higher *unit_capacity* would be reserved in the cluster.

Figure 5 illustrates an example of GeoCast with 13 nodes to further explain the operations of the shortcut clustering scheme. Let m be 9. Node E_{12} triggers the scheme after the subset $S_7 = \{E_{11}\}$ is created because of the latest split. First, subsets $\{E_2\}$, $\{E_5\}$, $\{E_{10}\}$ and $\{E_{11}\}$ move to *templist* to maintain the accessibility of those subsets. As the result of that, *peernodelist*(12) is changed to: $\{\{E_1, E_4\}, \{E_3, E_7\}, \{E_6, E_9\}\}$. Then, step 6 is performed, Given E_6's low *unit_capacity*, it is deleted from the subset that has highest Num^i_{del}. For

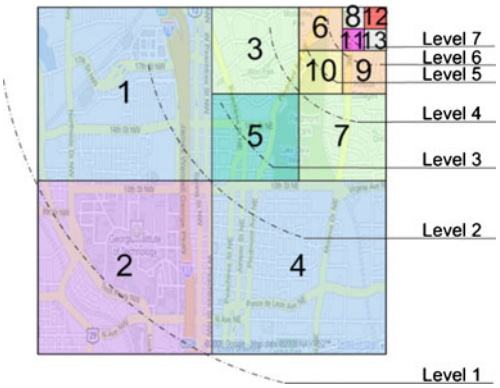


Fig. 5 Shortcut clustering

a smaller m, this clustering procedure will be executed in the similar manner. Here, it terminates. We have *peernodelist* (12) = {{E_2},{E_1, E_4},{E_5},{E_3, E_7},{E_{10}}, {E_9}, {E_{11}}}.

Obviously, the choice of parameter m has significant influence on the routing performance of UDR [31, 32]. The higher m value means more shortcut nodes are maintained in the *peernodelist*, the higher probability the UDR can find the best forwarding path in terms of routing efficiency. However, with setting of m 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.

5.2 Top-k routing scheme

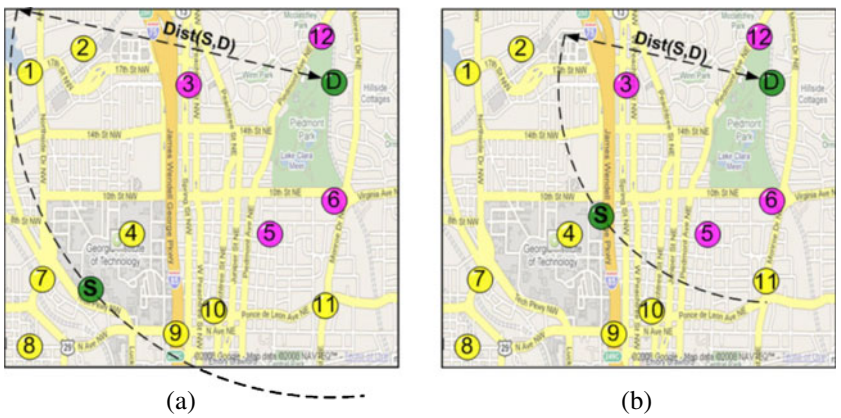
Note that UDR computes a utility value for each node in the node's *peernodelist* to determine the best node for message dissemination. In some situations, it results in a high computational complexity [27, 34], especially when the system is large. To overcome that, top-k routing scheme is proposed, which can be viewed as an extension of the UDR scheme mentioned in Section 4.

The idea of top-k routing is to intelligently pick k nodes from the node's *peernodelist* and route the message to one of k nodes which has the smallest utility value so that the complexity of computing is reduced [26]. Specifically, the top-k routing scheme first ranks the entries in the node's *peernodelist* in ascending order of geo-distance. Then top-k nodes of the ranked list are picked as candidate nodes for the forwarding node selection. In the selection procedure, the utility value of each candidate node is calculated by using Eq. 1. By comparison with the candidate nodes, the node that has the smallest utility value is selected, and receives the message from the message carrying node. This process is repeatedly executed until the message reaches the destination node.

Figure 6 is an illustration of the top k routing with $k = 4$. Node S and node D refer to the source node and distinction node respectively. Initially, node S has 10 candidate nodes except nodes E_7 and E_8, which are filtered out by the Geo-distance filter (see in Fig. 6a). Instead of calculating utility value for those candidates, top k-node routing only picks up 4 nodes: E_3, E_5, E_6 and E_{12}, for the forwarding node selection.

It is interesting to note that when we set k to 1, top-k routing is identical to the SGD, in which the node that is closest to the destination in terms of geo-distance will be selected to route the message at each hop along the routing path. On the contrary, if we set k a value that is larger than the total number of entries in the peernoselist, all the entry node are considered in the procedure of the forwarding node selection. Given that, we conjecture that it might be essential to determine a better k for the top-k routing scheme to prevent the nodes in the system from long message transmission delay and high time complexity.

Fig. 6 Top-k routing



However, the experiential results show that the efficiency of top-k routing scheme is not highly related to the setting of parameter k. Specifically, with setting of k in [2,6] top-k routing provides great saving in terms of routing latency while maintaining good routing path length. Furthermore, it is worth noting that the performance of top-k routing scheme degrades to that of UDR when the destination node is located in the vicinity of message sender or forwarder, as shown in Fig. 6b. Since only k nodes are considered at each hop, the time complexity of UDR can be reduced from $O(2\log^2 N + \log N \times \log(\log N))$ to $O(2klogN)$. It indicts that the top-k routing serves as a super efficient optimization for multicast tree construction.

5.3 Automated latency scaling for coping with network dynamics

We observe that the link latency metric used in the UDR algorithm can at times lead to serious instability of multicast trees [20, 35]. To deal with such instability, the exponential smoothing algorithm in [29] can be employed, where the advertised link latency is unchanged until the advertised 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 level [10, 36]. The motivation is to determine the adequate threshold value for capturing the significant changes in the advertised link latency. Our automated latency scaling technique consists of two components. First, 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. Second, 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 latency discrimination ensures that all overlay links 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 $\{20\ \text{ms}, 40\ \text{ms}, 60\ \text{ms}, 80\ \text{ms}, 100\ \text{ms}, 120\ \text{ms}, \cdots\}$. Specifically, two latencies are considered equivalent if they map to the same level. This automated latency scaling approach not only enables greater stability in multicast trees on the overlay networks, but also allows the metric of geo-distance to become a determining factor when different links have similar but not identical latency. In this case, the structure of multicast can be maintained in a relatively stable level. Each node along the multicast tree route actively detects the change of network by monitoring its links' latency variation. The stale routing path can be abandoned in a timely fashion when it has a relatively longer latency than that of other links. Our experimental results in the next section show that the automated latency scaling among the L levels can significantly enhance the ability of URD in response to the link dynamics in the overlay networks, as shown in Fig. 21.

6 Performance evaluation

In this section, we investigate the performance of the proposed UDR algorithm. Specifically, the experiments are designed to measure the following three subjects: the effectiveness of the UDR in routing messages and tolerating node failures in terms of metrics (e.g., latency, link stress, churn rate, and maintenance cost) by comparing to three algorithms (i.e.,CAN like geo-distance based routing (NB) [12], RTT weighed neighbor-based routing (RNB) [12, 18, 19], and geo-distance based routing with shortcut (SGD)), the impact of adaptive setting, and the effect of optimization techniques. We start with describing the setting of experiments.

6.1 Experimental setting

We use GT-ITM topology generator to generate 10 internet-like physical networks, and then build GeoCast on top of the physical networks. Each topology consists of 8080 routers 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. Nodes are randomly attached to the stub domain routers through LAN link and organized into the GeoCast overlay networks.

Given 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 types: [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. The results are measured by averaging 100 runs (10 runs in each topology with same setting in such a way the inaccuracy incurred by stochastic selection is minimized).

Table 2 Performance metrics

Term	Description
Path length	The average number of routing hops required for a message transmitted 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 transmitted 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 given a subscriber-publisher set.
Utilization ratio	The ratio of the number of shorter delay routing pathes formed by the UDR to the number of routing pathes formed by the SGD given a source-destination set
Churn rate	The proportion of failure nodes to the nodes in the system

By following independent and identical exponentially distribution, a sequence of node departures is generated to simulate node failure. The failure times of sequence nodes that are randomly chosen are assigned, which range from [0T, 60T] simulated seconds. T = 120 is the parameterized heartbeat period. In our figures, each data point represents the average of measurements over 50 trials (5 trials in each topology with same setting in such a way the inaccuracy incurred by stochastic selection is minimized).

The metrics used to in our experimental evaluation are summarized in Table 2.

6.2 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 8,000 nodes. Table 3 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 an uniform distribution on the interval [0,1]. Such scheme is the general case of UDR routing, employed as a representative of UDR routing in routing scheme comparison before the impact of adaptive parameter setting is studied in detail. From the results in the Table 3, we can see that both SGD and UDR with random setting exhibit better performance than NB and RNB in terms of path length. In a system of 8,000 nodes, no more than 5 hops are needed to delivery the message from the source node to the destination node. This is because in both NB and RNB, only immediate 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 the others.

Table 3 Path length for routing algorithms

N	NB	RNB	SGD	UDR with random setting
1,000	11.61	14.16	3.96	4.259
2,000	14.16	18.23	3.92	4.219
4,000	20.19	25.71	4.44	4.458
8,000	28.1	35.49	4.76	4.94

Figure 7 shows the effect of system size on routing latency. The shorter the routing latency is, the quicker the message can be received by the destination node and the better the routing scheme is. In Fig. 7, we find that both SGD and UDR outperform the others due to their shorter forwarding routes. When the system size is larger, the differences become more pronounced.

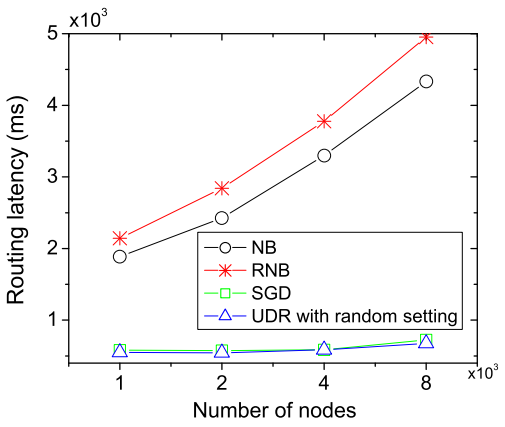


Fig. 7 Routing latency

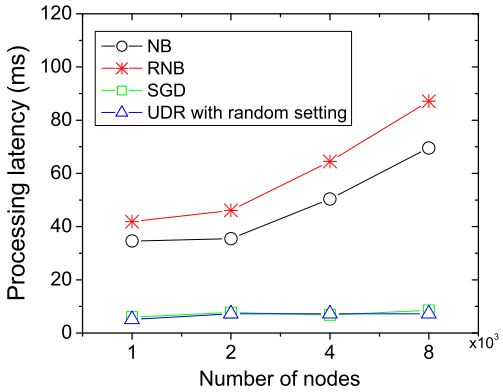


Fig. 8 Processing latency

This is because the routing path is getting longer as the system size increases as shown in Table 3. Specifically, in the system of 8,000 nodes, RNB needs to take about 5,000 ms to transmit messages to the destination on average, which is about 8 times as many as that of routing schemes with shortcut (SGD or UDR).

Figure 8 shows the processing latency for the routing schemes as a function of system size. The results demonstrate that the shortcut based routing schemes (e.g., SGD and UGR) perform better than its competitors. With the growth of system sizes, the shortcut based routing scheme keeps a relatively steady performance and takes no more than 15 ms for message processing during the entire routing procedure. In contrast, NB and RNB 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 1,000. In each simulation, we set $N = 4{,}000$. Figure 9a and b measure the multicast latency as function of group size and the number of groups respectively. From the results, we observe three interesting facts: (1) GeoCast based on UDR with random setting exhibits the best performance in all cases. The results show that the end-to-end latency of GeoCast based on UDR with random setting is always lower than the others. Even if the group size increases to 1,000, it is able to manage the latency variation within a small range. (2) The lack of link latency consideration leads to a performance degradation of GeoCast based on SGD, even though with shortest message routing pathes (see the results of GeoCast with SGD). (3) The end-to-end latency of GeoCast with NB is much higher than that of the other schemes given the long routing path of NB.

Fault resilience We now study the effects of node failure on the routing performance. We vary the churn rate (CR) from 0.2 and 0.8. As shown in Fig. 10, we observe that the performance of our scheme still outperforms the others. As CR increases, the end-to-end latency of multicast trees increase. This is, essentially, because the multicast tree needs to take longer to recover itself from the service interruptions caused by the departures of tree node. In addition, it is important to note that our UDR scheme consumes less recovery messages than that of GeoCast with SGD, as shown in Fig. 10b. Potentially, it demonstrates the efficiency of the UDR routing scheme.

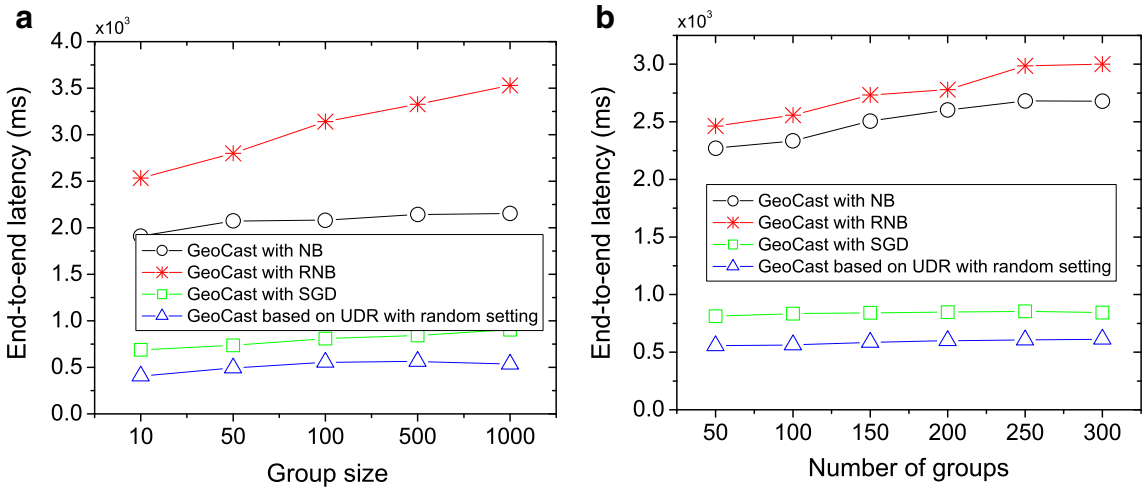


Fig. 9 The latency of multicast tree

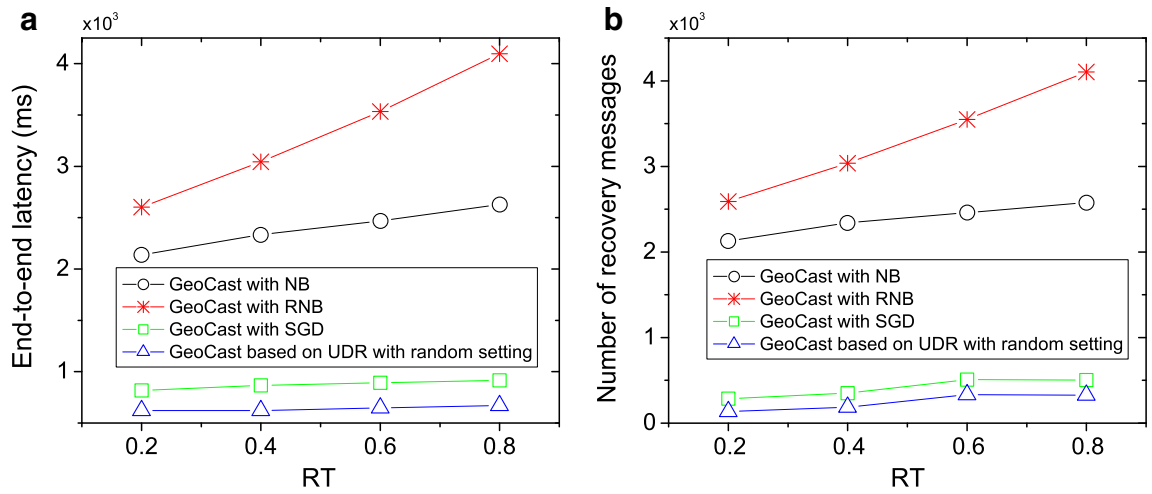


Fig. 10 Churn rate on multicast performance

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. 11, we can see both GeoCast with SGD routing and GeoCast based on UDR with random setting routing impress lower stress on the physical links. Contrarily, the links in either GeoCast with NB or GeoCast with RNB are having high load, which confirms the results showed in Table 3. Since shorter forwarding path usually incurs fewer IP messages, our scheme is efficient in eliminating the redundant messages on the physical links.

6.3 Impact of adaptive setting

Figure 12 examines the effect of parameter setting on routing performance. We observe that UDR with the setting of $\mu = 0$ does not perform as well as that of other settings in all cases as shown in Fig. 12a. It is because with such setting, messages tend to be routed to the immediate 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. 12b, we observe that the end-to-end latency can be minimized in all cases when μ is set to 0.6. Additionally, we find that such setting enables the routing path length to be limited within $O(\log N)$ hops (see results in Table 3), which makes it more attractive than the existing approaches. However, as we discussed in Section 4, the setting of $\mu = 0.6$ is unlikely to be the optimum parameter of our routing scheme in another different scenarios. Figure 13 shows that the optimum parameter setting of μ is changed to 0.4 when the number of nodes in the system decreases to 800.

In Fig. 14, we combine the approach of theory analysis with experiment results to study the effect of the system size on the optimum influence parameter μ. The solid line marked with circle is plotted by using results that we collected in experiments and the slash line marked with star is nonlinear fitting curve plotted by the analysis of those results.

From the experiment results, we observe that the optimal influence parameter is episodically increasing with the number of nodes in the system because of the relative stable distribution of system nodes. The topology of system is less likely to change when the number of nodes is getting larger. This is also the reason why the optimum parameter converges to 0.6 in the system of 8,000 nodes. However, the question is how to automatically adapt μ to the changes of network condition.

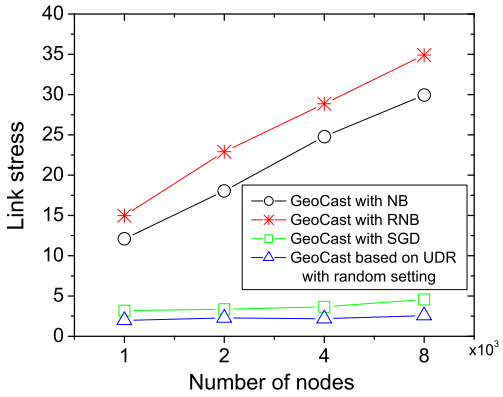


Fig. 11 Link stress

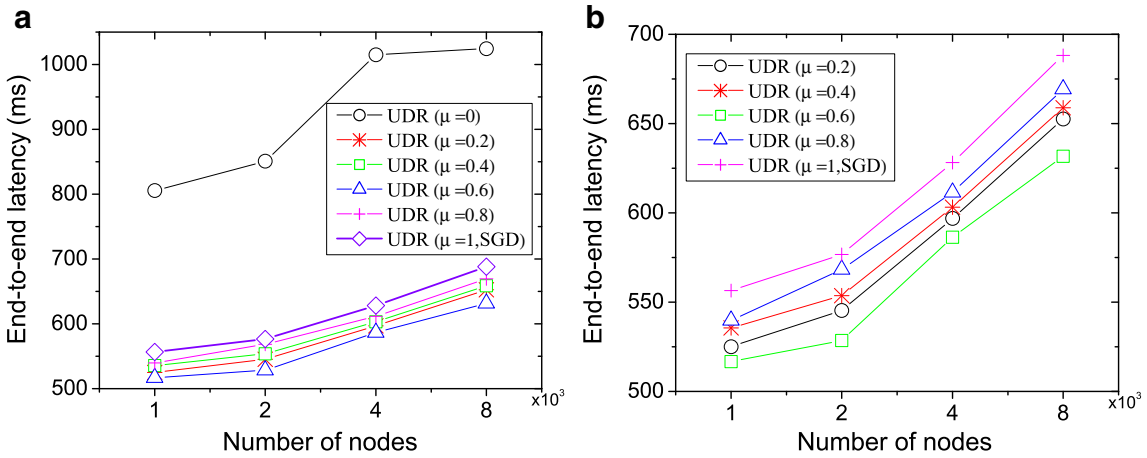


Fig. 12 End-to-end latency

To answer that question, we conduct a performance comparison among UDR with optimal setting, UDR with adaptive setting, and SGD routing ($\mu = 1$). In Fig. 15, the solid line marked with diamond is plotted by using the 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 tightly follows the UDR with measured optimal setting. The SGR incurs higher end-to-end 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% and a maximal improvement ratio of 83.98% in terms of end-to-end latency.

The effectiveness of UDR with adaptive setting is also studied by using the metric of utilization ratio. The results in Fig. 16 show that up to 34% of routing paths have been improved among nodes in the system with 8,000 nodes by using our adaptive solution. In our work, a routing path is considered to be improved for a source-destination pair if the end-to-end latency between them is reduced. To make the routing pathes comparable, same set of source-destination pairs is used in each round. A total of 1,000 rounds are conducted in each simulation.

Given that, we use the technique of UDR with adaptive setting to study the performance of the optimization techniques mentioned earlier in the following experiments. For the sake of brevity, the term UDR is used to refer to the UDR with adaptive setting in the rest of this paper.

6.4 Effect of optimization techniques

6.4.1 Shortcut clustering

Since it is hard to specify the value of m in each simulation, we use shortcut availability to investigate the performance of three different schemes: shortcut clustering, random filtering and distance based filtering [30].

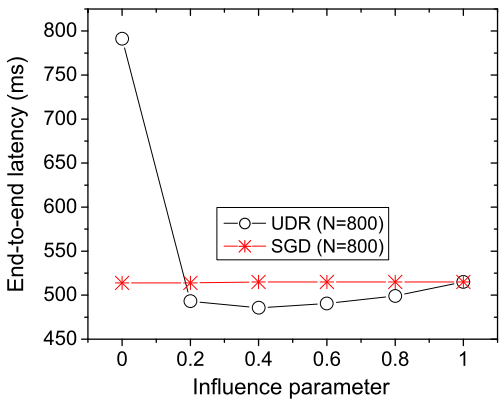


Fig. 13 Influence parameter

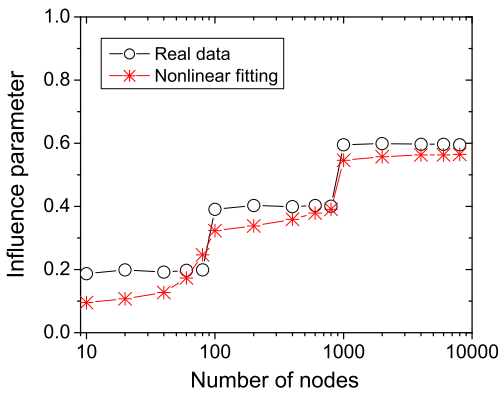


Fig. 14 Effect of system size

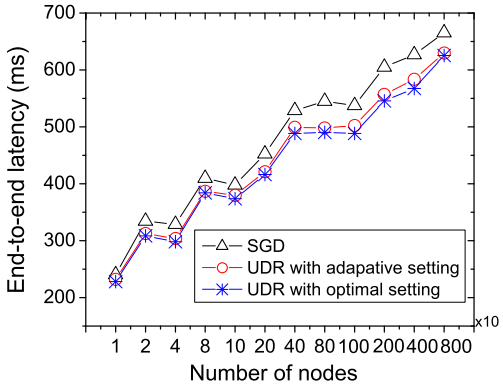


Fig. 15 Effect of adaptive setting

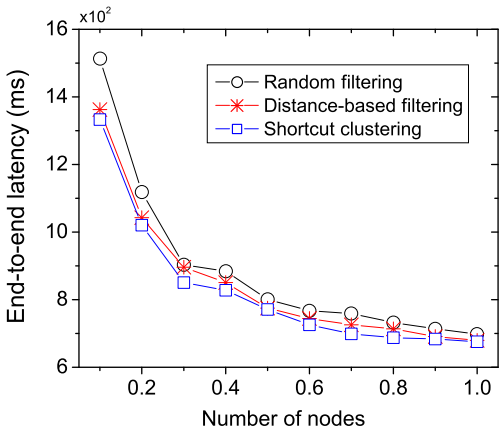


Fig. 17 Comparison of node filtering schemes

In the random filtering approach, each node randomly chooses shortcut nodes from its *peernodelist* to discard. We set N to 4,000. In Fig. 17, it is seen that the shortcut clustering steadily exhibits best performance as the shortcut availability increases because of its single node subset reservation. In the scenario that the region owned by a node is over split, the shortcut clustering tends to remove the redundant shortcuts on the lower levels from the *peernodelist*. Instead, the shortcuts that are located together are likely to be discarded by using the technique of distance-based filtering, which often increases the message end-to-end latency and consequently degrades the performance of routing schemes.

Figure 18 measures the maintenance cost of different systems as a function of shortcut availability. In the simulation, there are 10 trees consisting of 20 subscribers on average. At runtime, each root node issues 50 M meta-data to their groups and we measure the messages generated for data transmission and topology maintenance. In Fig. 18, we observe that the more shortcut nodes are maintained by nodes, the more maintenance messages are required in both basic GeoCast and enhanced GeoCast than in CAN system. 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 this is negligible given the better performance of GeoCast based on UDR with random setting in terms of end-to-end latency and adaptive ability that are studied below.

In Fig. 19, 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 routing latency. 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.

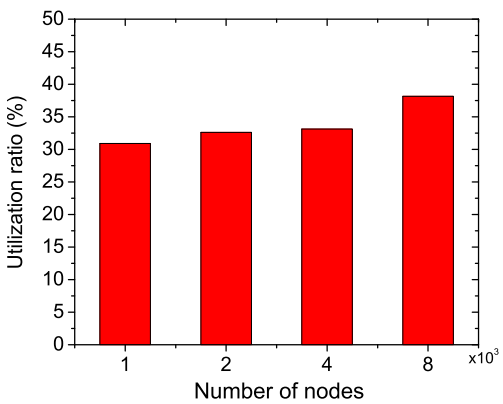


Fig. 16 Utilization rate

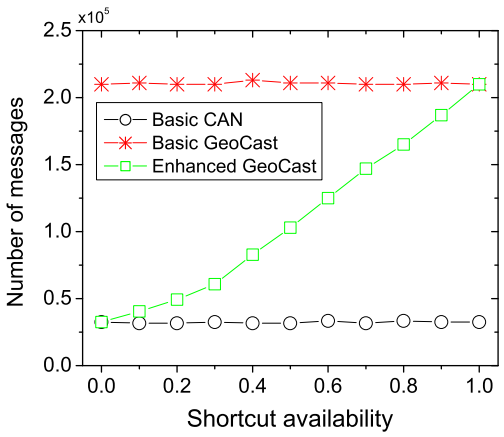


Fig. 18 Maintenance cost

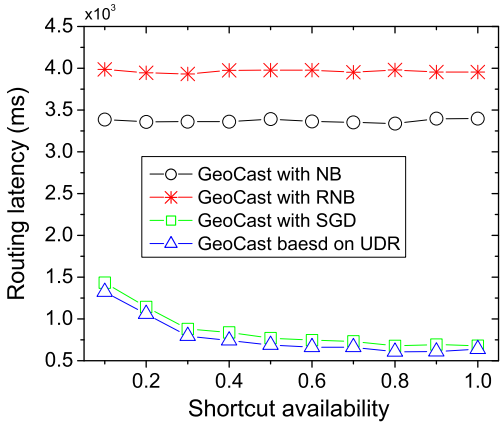


Fig. 19 Effect of shortcut clustering

6.4.2 Top-k routing

Figure 20a displays how parameter k affects the routing performance of top-k routing. It is seen that the performance of UDR is significantly improved by introducing the parameter k in terms of end-to-end latency. With the setting of k = 2, top-k routing achieves better performance than the other settings because of its low end-to-end latency. This is caused for two reasons. First, it avoids to involve many additional hops in the routing path when taking the latency factor into consideration. Second, the influence of the short coming inherited from SGD routing can be eliminated by utilizing the latency factor.

We also have interests in the impact of system size on the optimum parameter k. From the results plotted in Fig. 20b, we can see that the optimum parameter drops with the system size in an episodic manner. When the system size is small, the value of optimum parameter is equal to 6 and it changes to 2 when the system size increases to 1,000. Since the routing length deviation of top-k schemes is quite small when system size is less than 100, the effect of long routing path on the total latency can be ignored.

6.4.3 Automated latency scaling

Figure 21 depicts the end-to-end latency of the discretization scheme with different values of parameter ρ. The lower the end-to-end latency, the fewer the reconstruction operations and the better the discretization scheme. For simplicity, we set $N = 8{,}000$ and vary ρ from 0 to 150. When ρ is set to 0 ms, the scheme is viewed as a special case of our UDR scheme, where the mechanism of multicast tree reconstruction is triggered whenever the change of network is detected. In Fig. 21a, we observe that the schemes with $\rho > 0$ improve the performance of the multicast trees, and consequently reduce 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 is 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

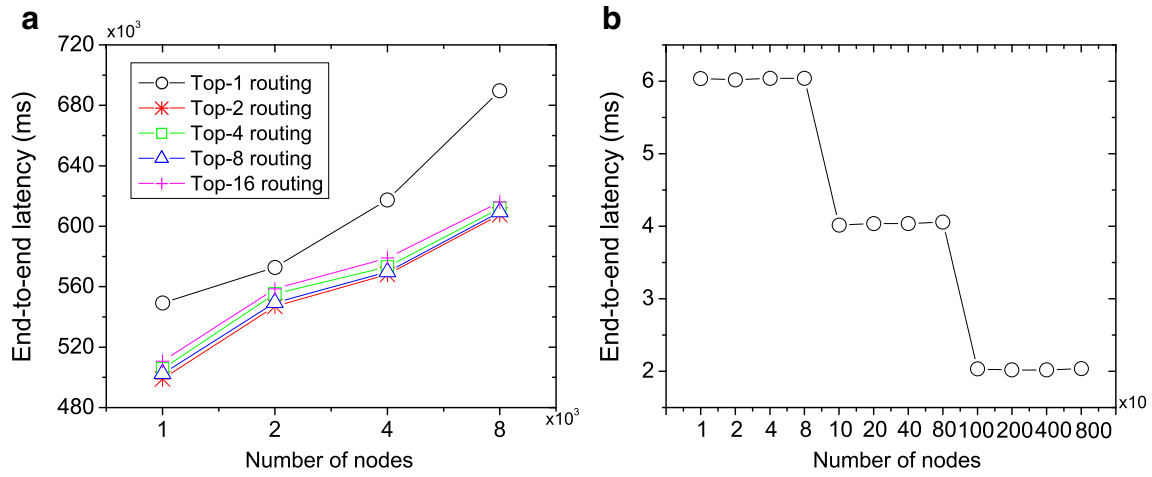


Fig. 20 Effect of parameter k

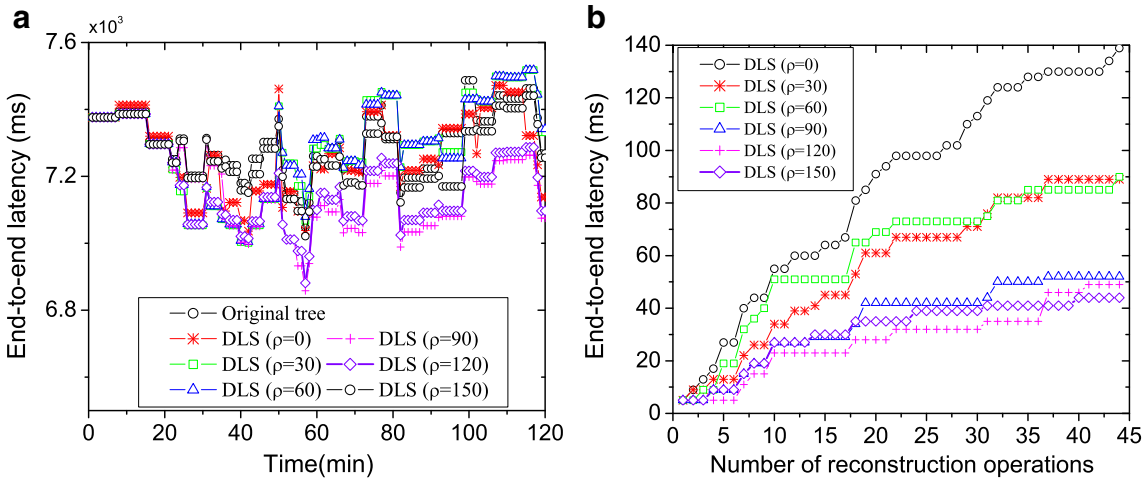


Fig. 21 Effect of discretization parameter

argue that after about 30 rounds, the reconstruction cost converge to a stable state with setting of $\rho = 90$, as shown in Fig. 21b.

7 Conclusion

In this paper, we highlighted the importance of network latency in the decision of message delivery for communication efficiency, and presented UDR, a self-adaptive utility driven routing scheme, to support group communication applications. UDR achieves communication efficiency by carefully taking into consideration the network latency in the message delivery path selection process in the face of frequent network dynamics and unpredictable node arrivals and departures. We further improved the performance of UDR with three optimization techniques (shortcut clustering, top-k routing and automated latency scaling). Through extensive experiments based on a realistic network topology model, we showed that the UDR scheme enables the nodes to make the near-optimal routing decision with respect to their specific circumstances and network scenarios and the UDR powered by its three optimization techniques delivers efficient and effective group communication services compared with existing geo-distance based routing protocols.

We are continuing our efforts in understanding the relationship between geo-distance and other factors such as bandwidth, inquiry rate, and node mobility, and developing more intelligent utility functions for providing scalable group communication services. As part of our future work, we plan to apply UDR with the new defined utility functions to different types of applications and environments to evaluate its efficiency.

Acknowledgements The first author performed this research as a visiting PhD student at Georgia Institute of Technology. This work is partially supported by grants from NSF CISE NetSE program, NSF CISE CyberTrust program, and an IBM faculty award, an IBM SUR grant, a grant from Intel Research Council, a grant from the National 973 Program of China (Grant No. 2009CB320805), a grant from the Natural Science Foundation of China (Grant No.61073070, 61170188) and Fundamental Research Funds for the Central Universities of China.

References

1. Castro M, Druschel P, Kermarrec AM, Nandi A, Rowstron A, Singh A (2003) Splitstream: high-bandwidth multicast in cooperative environments. ACM SIGOPS Oper Syst Rev 37(5):298–313
2. Kostic D, Rodrigues A, Albrecht J, Vahdat A (2003) Bullet: high bandwidth data dissemination using an overlay mesh. In: Proceedings of the 19th ACM symposium on operating oystemsorinciples (SOSP'03), pp 282–297
3. Banerjee S, Bhattacharjee B, Kommareddy C (2002) Scalable application layer multicast. In: Proceedings of the annual conference of the special interest group on data communication (SIGCOMM'02), pp 205–217
4. Castro M, Druschel P, Kermarrec AM, Rowstron AIT (2002) Scribe: a large-scale and decentralized application-level multicast infrastructure. IEEE J Sel Areas Commun 20(18):1489–1499
5. Zhang J, Liu L, Pu C, Ammar M (2004) Reliable peer-to-peer end system multicasting through replication. In: Proceedings of the 4th IEEE international conference on peer-to-peer computing (P2P'04), pp 235–242
6. Zhang X, Liu J, Li B, Yum TSP (2005) Coolstreaming/donet: a data-driven overlay network for efficient live media streaming. In: Proceedings of the 24th annual IEEE international conference on computer communications (INFOCOM'05), pp 13–17
7. Chu Y, Rao SG, Seshan S, Zhang H (2002) A case for end system multicast. IEEE J Sel Areas Commun 20(8):1456–1471

8. Jannotti J, Gifford DK, Johnson KL, Kaashoek MF (2000) Overcast: reliable multicasting with on overlay network. In: Proceedings of the 4th conference on symposium on operating system design & implementation (OSDI'00), pp 14–29
9. Castro M, Druschel P, Hu YC, Rowstron A (2003) Topology-aware routing in structured peer-to-peer overlay networks. Future directions in distributed computing, pp 103–107
10. Venkataraman V, Yoshida K, Francis P (2006) Chunkyspread: heterogeneous unstructured tree-based peer-to-peer multicast. In: Proceedings of the 14th IEEE international conference on network protocols (ICNP'06), pp 2–11
11. Pai V, Kumar K, Tamilmani K, Sambamurthy V, Mohr AE (2005) Chainsaw: eliminating trees from overlay multicast. In: Proceedings of the 4th international workshop on peer-to-peer systems (IPTPS'05), pp 127–140
12. Ratnasamy S, Francis P, Handley M, Karp R, Schenker S (2001) A scalable content-addressable network. In: Proceedings of the annual conference of the special interest group on data communication (SIGCOMM'01), pp 161–172
13. Zhang J, Zhang G, Liu L (2007) GeoGrid: a scalable location service network. In: Proceedings of the 27th IEEE international conference on distributed computing systems (ICDCS'07), pp 60–67
14. Stoica I, Morris R, Liben-Nowell D, Karger DR, Kaashoek MF, Dabek F, Balakrishnan H (2003) Chord: a scalable peer-to-peer lookup protocol for internet applications. IEEE/ACM Trans Netw 11(1):17–32
15. Rowstron A, Druschel P (2001) Pastry: scalable, distributed object location and routing for large-scale peer-to-peer systems. In: Proceedings of IFIP/ACM international conference on distributed systems platforms (Middleware'01), pp 329–350
16. Zhao BY, Kubiatowicz J, Joseph AD (2001) Tapestry: an infrastructure for fault-tolerant wide-area location and routing. IEEE J Sel Areas Commun 74(46):11–20
17. Maymounkov P, Mazieres D (2002) Kademlia: a peer-to-peer information system based on the xor metric. In: Proceedings of the 1st international workshop on peer-to-peer systems (IPTPS'02), pp 53–65
18. Gummadi K, Gummadi R, Gribble S,Ratnasamy S, Shenker S, Stoica I (2003) The impact of DHT routing geometry on resilience and proximity. In: Proceedings of the annual conference of the special interest group on data communication (SIGCOMM'03), pp 381–394
19. Dabek F, Li J, Sit E, Robertson J, Kaashoek MF, Morris R (2004) Designing a DHT for low latency and high throughput. In: Proceedings of the 1st symposium on networked systems design and implementation (NSDI '04), pp 7–20
20. Wu J, Liu B, Zhang S, Lu Z, Zhong Y (2007) KadStreaming: a novel Kademlia P2P network-based VoD streaming scheme. In: Proceeding of the 7th Iconference on computer and information technology (CIT'07), pp 405–410
21. Castro M, Druschel P, Hu YC, Rowstron A (2002) Exploiting network proximity in peer-to-peer overlay networks. Technical report MSR-TR-2002-82
22. Dabek F, Kaashoek MF, Karger D, Morris R, Stoica I (2001) Wide-area cooperative storage with CFS. ACM SIGOPS Oper Syst Rev 35(5):202–215
23. Castro M, Jones MB, Kermarrec AM, Rowstron A, Theimer M, Wang H, Wolman A (2003) An evaluation of scalable application-level multicast built using peer-to-peer overlays. In: Proceedings of the 22nd annual IEEE international conference on computer communications (INFOCOM'03), pp 1510–1520
24. Ren S, Guo L, Jiang S, Zhang X (2004) SAT-Match: a self-adaptive topology matching method to achieve low lookup latency in structured P2P overlay networks. In: Proceedings of the 18th international parallel & distributed processing symposium (IPDPS'04), pp 83–91
25. Xu Z, Zhang Z (2002) Building low-maintenance expressways for p2p systems. Technical report HPL-2002-41
26. Han P, Xie B, Yang F, Wang J, Shen R (2004) A novel distributed collaborative filtering algorithm and its implementation on p2p overlay network. In: Proceedings of 8th Pacific-Asia conference on advances in knowledge discovery and data mining (PAKDD'04), pp 106–115
27. Nakao A, Peterson L, Bavier A (2003) A routing underlay for overlay networks. In: Proceedings of the annual conference on applications, technologies, architectures, and protocols for computer communications (SIGCOMM'03), pp 11–18
28. Wang Y, Liu L, Pu C, Zhang G (2009) GeoCast: an efficient overlay system for multicast applications. Technical report GIT-CERCS-09-14
29. Chu Y, Rao S, Seshan S, Zhang H (2001) Enabling conferencing applications on the internet using an overlay muilticast architecture. In: Proceedings of the annual conference of the special interest group on data communication (SIGCOMM'01), pp 55–67
30. A tutorial on clustering algorithms. http://home.dei.polimi.it/matteucc/Clustering. Accessed on Aug 2009
31. Bandyopadhyay S, Giannella C, Maulik U, Kargupta H, Liu K, Datta S (2006) Clustering distributed data streams in peer-to-peer environments. Inf Sci 14(176):1952–1985
32. Ramaswamy L, Gedik B, Liu L (2005) A distributed approach to node clustering in decentralized peer-to-peer networks. IEEE Trans Parallel Distrib Syst 9(16):814–829
33. Datta S, Giannella C, Kargupta H (2006) K-means clustering over a large, dynamic network. In: Proceedings of the 6th SIAM international conference on data mining (SDM'06), pp 153–164
34. Löser A, Staab S, Tempich C (2005) Semantic methods for p2p query routing. In: Proceedings of 3rd German conference on multiagent system technologies (MATES'05), pp 15–26
35. Rzadca K, Yong J, Datta A (2009) Multicast trees for collaborative applications. In: Proceeding of the 9th IEEE/ACM international symposium on cluster computing and the grid, pp 60–67
36. Ren D, Li Y, Chan S (2008) On reducing mesh delay for peer-to-peer live streaming. In: Proceeding of the 27th conference on computer communications (INFOCOM'08), pp 1058–1066