Wang YH, Zhou Z, Liu L *et al.* Fault tolerance and recovery for group communication services in distributed networks. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 27(2): 298–312 Mar. 2012. DOI 10.1007/s11390-012-1224-1

Fault Tolerance and Recovery for Group Communication Services in Distributed Networks

Yue-Hua Wang[1,2] (王跃华), *Student Member*, *IEEE*, Zhong Zhou[1,2] (周　忠), *Member*, *CCF*, *ACM*, *IEEE*
Ling Liu[3], *Senior Member*, *IEEE*, and Wei Wu[1,2] (吴　威), *Member*, *CCF*

[1] *State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing 100191, China*

[2] *School of Computer Science and Engineering, Beihang University, Beijing 100191, China*

[3] *College of Computing, Georgia Institute of Technology, Atlanta 30332, U.S.A.*

E-mail: yuehua.research@gmail.com; zz@vrlab.buaa.edu.cn; lingliu@cc.gatech.edu; wuwei@vrlab.buaa.edu.cn

Received April 28, 2011; revised December 4, 2011.

Abstract Group communication services (GCSs) are becoming increasingly important as a wide field of promising applications has emerged to serve millions of users distributed across the world. However, it is challenging to make the service fault tolerance and scalable to fulfill the voluminous demand of users in a distributed network (DN). While many reliable group communication protocols have been dedicated to addressing such a challenge so as to accommodate the changes in the network, they are often costly or require complicated strategies to handle the service interruptions caused by node departures or link failures, which hinders the service practicability. In this paper, we present two schemes to address the challenges. The first one is a location-aware replication scheme called NS, which makes replicas in a dispersed fashion that enables the services on nodes to gain immunity of failures with different patterns (e.g., network partition and single point failure) while keeping replication overhead low. The second one is a novel failure recovery scheme that exploits the independence between service recovery and structure recovery in time domain to achieve quick failure recovery. Our simulation results indicate that the two proposed schemes outperform the existing schemes and simple alternative schemes in service success rate, recovery latency, and communication cost.

Keywords fault tolerance, failure recovery, replication, location, group communication

1 Introduction

The last two decades have witnessed an increasing demand for high reliable and scalable communication services in many emerging applications, such as emergency altering, E-commerce, location-based advertising, VoIP, traffic control, and stock quote dissemination. These applications have potentially thousands of users spanning wide area networks that require reliable support by applications even in a network of nodes with dynamic behaviors and heterogeneous properties; hence, the quality of the group communication services (GCSs) (i.e., reliability and scalability) is of great importance.

However, a major research challenge is how to make the service fault tolerance and scalable to fulfill the voluminous demand of users in a dynamic distributed network (DN). To solve this, a large number of studies have been preformed. Examples include ESM[1], Scribe[2], Scattercast[3], Yoid[4], NICE[5], OMNI[6], Overcast[7], PeerCast[8], Splitstream[9], Bullet[10], Coolstream[11] and ChainSaw[12]. By carefully examining these diverse research efforts, we observe two important facts.

First, the reliability of information dissemination path tends to dominate the quality of the services built on the top of that. It is common that the services are interrupted by the failures of nodes in the information dissemination paths. One intuitive approach to achieve fault tolerance is to construct and maintain a

Regular Paper

This work is partially supported by National Science Foundation (NSF) grant from CISE NetSE Program and CyberTrust Cross-Cutting Program of USA, IBM faculty award, IBM SUR grant, grant from Intel Research Council, the National Basic Research 973 Program of China under Grant No. 2009CB320805, the National Natural Science Foundation of China under Grant No. 61170188, the National High Technology Research and Development 863 Program of China under Grant No. 2012AA011803, and Fundamental Research Funds for the Central Universities of China. The first author was supported by China Scholarship Council (CSC) and performed part of the work as a visiting Ph.D. candidate in 2007∼2009 at the Distributed Data intensive Systems Lab (DiSL) in Georgia Institute of Technology.

©2012 Springer Science + Business Media, LLC & Science Press, China

new structure or redundant information dissemination path such that the failures of node can be restored in a small region or the information can transmit through the redundant path to reach the users once the original path fails. However, this approach has a main drawback, that is, it is often costly or requires complicated management strategies, which hinders the service practicability and scalability.

Second, besides the overhead related to the reliability enhancement, the scalability of services greatly depends on the overhead caused by the service recovery. Given the fact that the resource limitation and unpredictable behaviors of the nodes in the network, the service recovery overhead should be small so as to avoid the system from a burst of messages sent for restoring the service interrupted by the node failures. Thus, it is essential to provide a quick and gracefully recovery strategy by taking the overhead into account. Surprisingly, we find most of the existing work to date[4-5,8,13-14] devote their attention to either the problem of minimizing the overhead caused by the reliability enhancement, or the problem of service recovery overhead minimization.

In this paper, we use the concept of replication and propose two schemes that can be employed as components to enhance the reliability and scalability of services provided by applications without changing the current infrastructures. In particular, our main contributions are: 1) a location-aware replication algorithm that makes replicas in a dispersed fashion, which enables the services on nodes to gain immunity of failures with different patterns (e.g., network partition and single point failure) while keeping replication overhead low. 2) a novel failure recovery scheme, which achieves fast failure recovery by utilizing the independence between service recovery and path recovery; and 3) an extensive evaluation that validates the effectiveness and efficiency of the proposed schemes. Specially, we apply them to GeoCast[15], a geographically distributed overlay system, and the experimental results show that the proposed schemes greatly improve the system performance in terms of service success rate, recovery latency, and communication cost, compared with the existing neighbor-based replication scheme, random replication scheme, and the simple alternative schemes.

The rest of this paper is organized as follows. Section 2 discusses the existing studies related to our work. Section 3 presents the system model and defines the problems of service reliability enhancement. We describe the structure of GeoCast in Section 4, and present the details of replication and failure recovery scheme in Section 5. In Section 6, several important issues related to the design of the proposed schemes are discussed. Section 7 presents the extensive performance evaluations. We conclude the paper with a summary and outline of our future work in Section 8.

2 Related Work

Application level Multicast (ALM) has emerged as an alternative paradigm to IP Multicast to provide group communication services for the end nodes that are widely distributed across the network. In ALM, the multicast related features, such as message routing, message propagation, and group membership management, are implemented at end nodes instead of routers, that overcome the limitations of IP multicast caused by the network infrastructure and business model. Over the last two decades, researchers have proposed many schemes in this literature. Based on the mechanism of multicast group construction, generally, they can be classified into two categories: gossip-based multicast scheme[11-12] and tree-based multicast scheme[1-3,5-10].

With gossip-based multicast scheme, each node periodically chooses a number of random nodes to whom it relays the data that has received. Once the failure occurs, such data duplications at the random nodes are used to recover the data loss caused by the failure. ChainSaw develops an efficient gossip-based multicast protocol to eliminate the data duplication transited among the end nodes. Instead of disseminating the data to the random nodes, each node in [12] collects the information about the data stored on its adjacent nodes and uses such information to guide the data transmissions among the nodes with the similar interests. In Coolstream, the gossip-based multicast scheme is used for supporting live media streaming services. Based on data availability and demanding information on nodes, Coolstream enables the streaming contents to be adaptively delivered to the end nodes in a high reliability manner. Although this scheme achieves good load balancing and high reliability in the presence of network dynamics (e.g., message loss, traffic congestion and node failure), it imposes high load on both the end nodes and network as many duplicate/notification messages are propagated among the nodes.

With tree-based multicast scheme, the information dissemination trees are explicitly constructed for data dissemination. For instance, in Overcast[7], a source-rooted multicast tree is built using end-to-end measurements. In such a way, the bandwidth between the source node and the group members is optimized. ESM[1] constructs spanning trees on top of a mesh structure. In each tree, all group members know about each others, which limits the scalability of the ESM. To address that, Scribe[2] designed a scalable ALM infrastructure based on Pastry to support the services with large numbers of group members. It enables the nodes

to maintain information of only a small subset of other nodes in the same group, which reduces the group management complexity and overhead. SplitStream[9] extends Scribe in which it requires the data to be divided into several disjoint sections and build a tree for each section. To receive the complete stream, a node must join every tree. Recent work, Wen *et al.*[16] provided a hybrid tree based explicit routed multicast mechanism for television content dissemination. It combines the advantages of two basic multicast trees (i.e., RP (rendezvous point) based shared tree (RPT) and source specific shortest-path tree (SPT)) to achieve required QoS guarantee and resource utilization efficiency.

The tree-based multicast scheme, though is efficient in multicast group management and resource utilization, has a major drawback, that is, it lacks robustness. With arrival and departure of the nodes, the tree structure is prone to fail, which may cause a burst of messages generated for tree management and reconstruction. To improve the resilience of multicast dissemination, Fei *et al.*[17] devised a dual-tree scheme. For each group, it constructs a secondary tree in addition to the primary tree normally used. Once the failure occurs, it activates the paths in the secondary tree. PRM[18] is a multicast data recovery scheme proposed to achieve high delivery ratio in the presence of node/link failures. In PRM, each node forwards data to both its child nodes and a constant number of random nodes during content dissemination. However, the volume of extra data duplications and traffic might be significant for the applications like on-demand news propagation and multi-party online game. By taking the replication creation and maintenance cost into consideration, Peercast[8] developed a neighbor-based replication scheme. However, this scheme has two drawbacks. First, it ignores the nodes heterogeneity in lifetime. Second, it subjects to a difficulty in determining how many replica nodes. To address those issues, [19] proposes a self-managing replication algorithm. But this scheme may appear inefficiency in dealing with service interruptions when network partitions happen. Overcast is motivated to provide service for bandwidth-intensive content dissemination. In overcast, nodes perform content caching and failure recovery operations to alleviate the influence of node or link failures. The drawback of this technique is the increased latency of content distribution as the data has to traverse all those extra nodes before reaching the rest of overcast nodes. In Scattercast[3], the multicast problems are partitioned into a set of smaller and simpler sub-problems that can be easily addressed with local knowledge of nodes in the same region. But such divide-and-conquer approach might be expensive, especially in a network with high link and node failure rate. A large number of additional messages may be generated for the service recovery and new region formation. In recent work, Hirokazu *et al.*[20] propose a distributed interval tree replication scheme for adaptively setting the replicated objects considering the scale of networks. The $O(\log N)$ replicas are saved against the scale of networks by storing all intervals to the tree where the number of network nodes N is known. However, it is difficult for such a scheme to set the scale of the interval tree space if the number of network nodes cannot be assumed.

There are several work proposed to cope with node failure and keep high data availability by using the clustering-based replication technology[21-25]. The main idea of this strategy is to place multiple object replicas on the nodes (servers) in the same cluster to reduce the distances between accessing sites and data (file) and maximize the availability of the data (file) in the network. Compared to cluster-based replication approaches, our approach has two distinct characteristics. First, instead of keeping data copies at cluster data servers[21,23-24] or nodes in the same cluster[22,25], these data are replicated onto neighbor nodes and shortcut nodes, so that they become widely available and the impact of different network partitions on the service quality is reduced. Second, unlike the cluster-based replication method that replicates data on demand[21] or according to the popularity[22-23], the proposed approach enables each node to take into account nodes' location to improve the replication flexibility and reduce the cost caused by a large number of replicas' creation and maintenance.

Our work presented in this paper bears similarity to previous work[8,19]. For example, we all use the concept of replication to achieve fault tolerance. However, two important features distinguish our approach from those proposed approaches. First, by taking the locations of and relationships between the replicas into account, we use the replication scheme to improve the resilience of the multicast dissemination while minimizing the cost of replication creation and maintenance. Second, by intelligently avoiding the delay of finding new nodes to restore the services from interruption and invoking the operations of service un-subscription and re-subscription, we greatly reduce the failure recovery overhead. The two schemes proposed in this paper are used as components to augment the performance of applications instead of changing the infrastructures of the current applications as well as systems.

3 Problem Definition

Since the service reliability and resource utilization demands of systems with group communication services

may not be known in advance, in this section, we introduce a general model, which aims at capturing the typical characteristics of the service reliability. In our work, a distributed system with group communication services is modeled as an undirected, connected graph $G = (V, E, L)$. V is the set of nodes, $E = V \times V$ is the set of edges, and $L : E \to \mathbb{R}^+$ is the set of propagation delay of the edges. Nodes represent users who participate in the system G, and edges represent links between the nodes. Weights represent communication cost of the links. Concretely, for each weight ω, it refers to the propagation delay $l_{j,k}$ required for a packet to transmit from node v_j to node v_k through the link $e_i = (j, k)$. In our model, the link between any pair of nodes is bidirectional, that is, $l_{j,k} = l_{k,j}$.

Each node E_i that enters the system has associated capacity C_i that is used to refer to the storage capacity of the node v_i since the storage capacity is one main factor in the Internet applications like content searching and media streaming dissemination. In our work, we denote C_i as the number of files can be stored on node E_i. Clearly, this may be quite different for the applications in reality. We use this assumption to construct the system model such that we can formulate the problem of replication and study the efficiency of the replication scheme with the consideration of node capacity limitation. Let $c_i(t)$ denote the current stored file number at time t. If $c_i(t) \geqslant C_i$, node E_i drops $c_i(t) - C_i$ files randomly and rejects all the replication requests issued from the other nodes in the system.

Given the dynamic nature of the network, we consider G as a system where the available capacities of the nodes may change as the nodes continuously join or leave the network. In such a dynamic network, the problem of fault tolerant is quite difficult to solve and, to the best of our knowledge, has not been addressed before. Formally, we define those problems respectively as follows:

Problem Statement. Given a system of n nodes with values of $c_i(t)$, $i \in [1, n]$ at time t. The replication problem is to find a replica placement strategy $RPS = \{e_1, e_2, \ldots, e_r\}$ for node E_i that maximizes the reliability of service S_i offered by node E_i while minimizing the replication cost under node capacity limitations.

However, solving the above problem is challenging as it requires carefully handling of the unpredictable node/link failures and time-varying load of the nodes in a dynamic network with network partitions. The random replication schemes, though have advantage in dealing with the failures in different patterns as mentioned come at high cost of replica detection and maintenance. To avoid that, in our scheme, both the neighbor node and the remote node are used as replicas to improve the reliability of the services offered by the unreliable nodes while keeping the overhead low. The benefits are two-fold. First, it reduces the replication cost by employing the nodes in vicinity. Second, it alleviates the influence of network partitions on the service quality by deploying the data copies on shortcut nodes.

In the rest of this paper, we focus the discussion on the schemes described above and apply to an existing system, GeoCast, to illustrate their efficiency in supporting reliable and scalable group communication services in a geographical overlay network with network dynamics. We leave other problems such as parameter optimization as our future work.

4 Background

In this section, we describe architecture of GeoCast and introduce multicast mechanism on which the algorithm of this paper is based.

4.1 Architecture

GeoCast is a structured geographical proximity-aware overlay network, consisting of nodes equipped with GeoCast middle-ware. In GeoCast, nodes are equivalent in functionalities, each of which can perform: message publishing, message subscribing, message routing or all of them. Each node keeps a set of information about other nodes in the network in its *peernodelist* that is a list containing two kinds of nodes: immediate neighbor node and shortcut node. Similar to CAN, two nodes are considered to be immediate neighbors when their intersection is a line segment. The term *shortcut node* refers to the node which is *old neighbor node* for a given node. As nodes arrive or depart, neighbor nodes may become old when they are not adjacent to the given node. Instead of removing old neighbors from the lists, GeoCast keeps those nodes in *peernodelist*s and uses them to speed up the procedure of message delivery.

GeoCast network is constructed incrementally. It starts with one node who owns the entire GeoCast space. When a new node arrives, it first initiates a join request destined for its own unique identifier (x, y), and sends it to an existing end system node, which is randomly selected from the list obtained from a bootstrapping sever[26-27]. Such a node acts as an entry node that helps new node find the region which covers its identifier (x, y). After identifying the region to which the new node belongs, the owner node of region checks its neighbors, selects the neighbor node with lowest *unit_capacity* to partition its region in half and assigns it to the new node. *unit_capacity* represents the workload of node per region unit, defined by: $unit\_capacity = \frac{E.workload}{E.R.w \times E.R.h}$. $E.workload$ refers to the

workload of node E (e.g., the number of link connections, the amount of space on disk used for file storage) and $E.R.w$ ($E.R.h$) refers to the width (height) of region R owned by node E. Then, update messages are issued to the neighbors of the partitioned node for notifying this modification so that the new node can be included in the procedure of message routing afterwards.

As node departs from system, the remaining nodes are responsible to take over the region occupied by that node. Similar to [27], its neighbor node with the smallest size is preferred to be selected as new responsible node for the region (named *island*) owned by the departure node.

4.2 Multicast Service

In GeoCast, the multicast trees are explicitly created to provide multicast services for the nodes demanding the same contents (called *subscribers*). Nodes can subscribe to any multicast service published in the network by initializing a subscription request. Such a request is continuously routed towards the content providing node (called *publisher*) until it encounters a node that is in a same tree. In each tree, no node has global knowledge about the nodes in the same group, that is, each node is only aware of a fraction of tree information and stores in a multicast routing table. In GeoCast, the multicast trees are formed by the routes from the publisher nodes to the subscriber nodes. As the new data arrives, the publisher injects the data at the root of the multicast tree, which gets disseminated through the tree and reaches all the subscribers.

The multicast service in GeoCast is introduced as one component for supporting group communication between nodes. It has two benefits. First, it releases high link stress caused by message transitions among nodes in group communication session and consequently improves network resource utilization. Second, it reduces the delay of the message delivery from the publishers to the subscribers. Instead of transmitting the data from the publishers, the subscribers often can get it from their upward 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. Detailed algorithms and examples of the multicast service establishment and maintenance process with node arrivals and departures are presented in [15].

5 Location-Aware Replication Scheme

In this section, we first introduce the pattern of failures and then present details of our replication scheme.

5.1 Failure Pattern

We consider a network consisting of a number of individual nodes that are inherently unreliable. In the network, any node can become unavailable for various reasons such as node departure, computer crash, improper program termination, like error, and traffic congestion. In the event of node departure, a number of messages are issued to the other nodes related to the departed node for notification. On receiving the message, the nodes actively cooperate together and restore the services that are interrupted by the node departure. However, due to the dynamic nature of the network and unpredictability of node behavior, node may leave the network without notification. In such an event, the failure recovery procedure of services related the failed node is booted in the absence of heartbeat or response message for a long time. For simplicity, we use the term failure to mean either departure or improper leave given the fact that the similar operations will be performed when they occur.

To provide the best quality of service, two patterns of failures are considered: distributed failure (i.e., random single node failure) and centralized failure (i.e., network partition). In the pattern of distributed failure, node failures are randomly scattered over network. There always exist surviving nodes around the failed node that can detect and repair its failure. In the centralized failure pattern, a fraction of nodes in the network appear to be reachable from the certain nodes but not others. Once it happens, the entire network might be partitioned into multiple, isolated overlay network parts. Though nodes within a network part can communicate with each other, there is no communication across the nodes in different network parts.

Note the difference between the failure patterns, it is essential to take them into account and design an efficient replication scheme to deal with them while keeping cost low. However, one intuitive problem is how to achieve that. In [28], a random replication scheme is proposed, where a set of nodes is randomly selected and employed as the replica nodes. This scheme is resilient to the centralized failure, but its performance may be impaired by the increased replication creation and maintenance cost, especially when the majority of the nodes in the system are heavily loaded. Contrarily, the neighbor-based replication approach[8] replicates the nodes' data to the nodes with available capacities that locate in adjacent so as to reduce the replica creation and management cost. However, it suffers from the difficulty in handling the network partitions. Therefore, it is desirable for nodes to maintain both the neighbor replica node and the remote replica node to improve replication efficiency in terms of both reliability and

scalability.

5.2 Replication Scheme

Two main components of the replication scheme design are present in this subsection. The first one is the parameter definition. In our design, a tunable parameter is introduced to adjust the importance of neighbor and shortcut replica nodes. Instead of randomly selecting remote nodes as replica nodes, in our scheme, we place some replica on shortcut nodes that are not located in the adjacent region of the replica placing node (called *host node*). The goal is to reduce the high cost of remote replica creation and maintenance while keeping the service reliability high. The second one is the replica placement that specifies how the replication scheme is performed in a distributed manner.

5.2.1 Parameter Definition

To leverage the benefits of neighbor and shortcut node, a tunable parameter α is introduced in our design. Its value ranges from 0 to 1. Consider a node p who has a replica placement task with replication degree r. When increasing α, node p reduces the number of neighbor replica nodes which causes the probability of service remaining available in the presence of network partitions to increase. However, larger α also leads to more shortcut replica nodes which in turn increases replica creation and maintenance cost. Based on those facts, we find the value of α cannot be too large or too small for the purpose of minimizing the replication cost and optimizing the reliability of service. Given an average of $O(2d)$ immediate neighbors is kept on each node in GeoCast[15], we have, for any node p,

$$\alpha = \begin{cases} 0.5, & \text{if } r \leqslant 2, \\ \dfrac{1}{r}, & \text{if } 2 < r \leqslant \tau, \\ \dfrac{r-\tau}{r}, & \text{if } r > \tau, \end{cases} \tag{1}$$

where τ is the number of neighbor nodes in the *peernodelist* of node p. The motive behind that is to relatively reduce the importance of shortcut replica so as to minimize the link stress imposed by placing and maintaining shortcut replicas. Typically, for a given τ, as increasing r, the number of neighbor replica nodes also increases. After r reaches $\tau + 1$, neighbor replica number reaches its upper bound τ, that is, no neighbor node is left in the *peernodelist* that can be selected as the replica node. In this case, more shortcut nodes tend to be involved in the replica placement.

5.2.2 Replica Placement

Replica placement performs by means of two-phase procedure: *neighbor selection* and *shortcut selection*. The *neighbor selection* is to select several nodes with available capacity in adjacent area as replica nodes. The *shortcut selection* is to place several replicas on the shortcut nodes. Nodes in the network may differ in the number of neighbor and shortcut replicas due to their states. Next, we will describe those phases in detail.

Neighbor Selection. Given the location relationship between nodes, a new notation of k-hop neighbor B^k is introduced and defined as:

$$B^k_{E_i} = \begin{cases} \{\text{immediate neighbors of } E_i\}, & \text{if } k = 1, \\ \cup_{j=1}^{m_1} B_{B^1_{E_i}(j)} - B^1_{E_i}, & \text{if } k = 2, \\ \vdots \qquad\qquad \vdots & \\ \cup_{j=1}^{m_{l-1}} B_{B^{l-1}_{E_i}(j)} - B^{l-1}_{E_i}, & \text{if } k = l, \end{cases} \tag{2}$$

where E_i denotes a node in the network and m_{l-1} is the number of nodes in $B^{l-1}_{E_i}$. When $k = 1$, all immediate neighbors of node E_i are 1-hop neighbors of E_i. As mentioned earlier, two nodes are considered as immediate neighbors when the intersection of their regions is a line segment. If $k = 2$, $B^2_{E_i}$ consists of the nodes that are the immediate neighbors of $B^1_{E_i}$'s entry nodes. For consistency, it requires the nodes in $B^1_{E_i}$ are not in $B^2_{E_i}$.

Fig.1 gives a simple example of network to illustrate the relationship between nodes. In Fig.1, nodes E_9, E_{16}, E_{21} and E_{14} are the immediate neighbors of node E_{15}. Nodes E_2, E_{10}, E_{17}, E_{22}, E_{27}, E_{20}, E_{13} and E_8 are immediate neighbors of the nodes in $B^1_{E_{15}}$, which are included in $B^2_{E_{15}}$.

The process of neighbor selection is divided into two main phases, *selecting* and *extending*. In *selecting*, a set of nodes with available capacities in $B^k_{E_i}$ are selected as replica nodes. In *extending*, $B^k_{E_i}$ extends to a larger set by including the immediate neighbors of the nodes in $B^k_{E_i}$. It enables more replica nodes to be detected in *extending*.

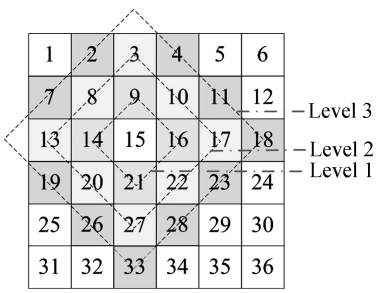

Fig.1. Relationship among nodes.

Algorithm 1 sketches the neighbor selection

algorithm. For node E_i, it starts with $B^1_{E_i}$. Node E_i first looks through $B^1_{E_i}$ to see if there are $(1-\alpha)r$ nodes with available capacities (line 2). If so, node E_i adds them into its *replicalist* R and replicates its multicast information on them, where R is a list created for reordering the information of nodes which are selected as the replica nodes (lines 4∼6). Then this selection procedure terminates (line 13). Otherwise, E_i adds all capable nodes at the first level into the *replicalist* and increases k by 1. Then the neighbor selection is performed at the second level (lines 3∼10). This procedure is executed repeatedly until either $(1-\alpha)r$ neighbor replica nodes are fetched or k reaches its maximum K. See Section 6 for a further discussion of K.

Algorithm 1. Neighbor Replica Placement $(B^1_{E_i}, r, \alpha, K)$
Input: $B^1_{E_i}$: the set of immediate neighbors of node E_i,
r: the replication degree,
α: the tunable parameter,
K: the maximum number of iterations
Output: R: the list of replicas
1 $l \leftarrow 0$; $k \leftarrow 1$;
2 **while** $k < K$ **do**
/*select neighbor replicas with sufficient capacities*/
3 **for** each node $p \in B^k_{E_i}$ and $p \notin R$ **do**
4 **if** $\max\{\frac{lo^s_p(t)}{c^s_p}, \frac{lo^{bw}_p(t)}{c^{bw}_p}\} < 1$ **then**
5 add node p into list R;
6 **end if**
7 **if** $l \geqslant (1-\alpha)r$ **then**
8 jump to line 13;
9 **end if**
10 **end for**
\\extending search region
11 $k \leftarrow k+1$
12 **end while**
13 employ nodes in R as neighbor replicas

Shortcut Selection. The shortcut selection is performed in a similar manner to the neighbor selection. Algorithm 2 starts with the set $S^1_{E_i} = peernodelist_{E_i} - B^1_{E_i}$, where $peernodelist_{E_i}$ is the *peernodelist* of node E_i. Then this procedure is continuously executed at the next level (lines 2∼11) as there is no enough capable shortcut nodes. Similar to the definition $B^k_{E_i}$, $S^k_{E_i}$ is defined as $S^k_{E_i} = \cup_{j=1}^{m_{k-1}} B_{S^{k-1}_{E_i}(j)} - S^{k-1}_{E_i}$. This selection procedure is terminated when either αr shortcut replica nodes are fetched or k reaches its maximum K.

Fig.2 gives an example to illustrate the shortcut selection with the setting of $r = 8$ and $\alpha = 0.25$. Node H first looks through $S^1_H = \{E_1, E_2, E_4, E_{14}, E_{18}\}$ and checks if there exist 2 shortcut nodes with available capacities. If so, no selection is necessary as enough replica nodes are detected. However, due to the heavy loads of nodes E_1, E_2, E_{14}, E_{18}, only E_4 is selected at the first level. To meet the requirement of $\alpha r = 2$, node

Algorithm 2: Shortcut Replica Placement$(S^1_{E_i}, r, \alpha, K)$
Input: $S^1_{E_i}$: the set of shortcuts of node E_i,
r: the replication degree,
α: the tunable parameter,
K: the maximum number of iterations
Output: R: the list of replicas
1 $l \leftarrow 0$; $k \leftarrow 1$;
2 **while** $k < K$ **do**
/*select shortcut replicas with sufficient capacities*/
3 **for** each node $p \in S^k_{E_i}$ && $p \notin R$ **do**
4 **if** $\max\{\frac{lo^s_p(t)}{c^s_p}, \frac{lo^{bw}_p(t)}{c^{bw}_p}\} < 1$ **then**
5 add node p into list R;
6 **end if**
7 **if** $l \geqslant (1-\alpha)r$ **then**
8 jump to line 13;
9 **end if**
10 **end for**
/*extend search region*/
11 $k \leftarrow k+1$
12 **end while**
13 employ nodes in R as shortcut replicas

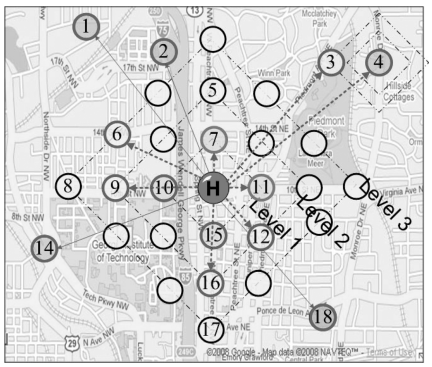

Fig.2. Shortcut selection.

H extends its search region and E_3 resided at the second level is then included in the *replicalist*. Similarly, nodes E_6, E_7, E_9, E_{11}, E_{12}, E_{16} that have available capacities are selected and then added into the *replicalist* after neighbor selection.

Our algorithm makes two effects to ensure scalability of the applications. One is to reduce the replica creation cost by intelligently using the information of the other nodes stored on the nodes. Rather than broadcasting its query request to number of nodes in the network, each node in the proposed scheme first inquires the nodes in its *peernodelist* for replica placement, which can improve the resource utilization. The other is to avoid high cost caused by remote replica maintenance in message delay. In the procedure of replica placement, the nodes with available capacities residing in vicinity have high priority to be selected as the replica nodes. It enables either data updating or replicas' communication to be completed within a short delay.

5.3 Managing Replicas and Recovering Failure

Rather than introducing extra additional overhead, we append the maintenance information of replicas to existing heartbeat message in our scheme. We do not eagerly notify replicas to update theirs backup information if the relationship of multicast tree has been charged. In our scheme, multicast messages also serve as implicit heartbeat messages avoiding the need for the explicit heartbeat messages. Long-time absence of the heartbeat or response indicates that the node is gone and its corresponding region becomes an island, and thus boots the recovery process. According to the diversity of node functionalities in the group session, we divide the recover process into two phases: path recovery and region recovery. Path recovery is to restore service path from the interrupted multicast routing path when the node failure occurs. In the process of the path recovery, only the living nodes are considered. The node with high available capacity owns higher preference to be selected as new member node. Region recovery is to search one node to take over the *island*.

5.3.1 Path Recovery

After capturing the event of node failure, one of replicas is randomly elected to manage the service recovery. It is responsible to find a replacement node for taking over the forwarding functionalities of the failed node. Ideally, it is expected to use a node with the lowest load to replace the failed node while not sacrificing routing performance. Given that, two factors, namely load factor and distance factor, are considered in the process of path recovery. Load factor refers to the willingness of the replica node to accept more multicast workloads considering its current load. Distance factor refers to the distance between failed node and its parent nodes. Thus, we have:

$$\mathrm{SPF}_i(t) = \frac{1 - c_i(t)}{\mathrm{Max}\{1 - C_i\}} + \frac{\mathrm{Min}\{\sum_{j=1}^{n} Gdist_{i \to P_j}\}}{\sum_{j=1}^{n} Gdist_{i \to P_j}}, \tag{3}$$

where $1 - c_i(t)$ denotes available capacity of the replica node, and $\sum_{j=1}^{n} Gdist_{i \to P_j}$ denotes the sum of distances from E_i to the failed node's parents at time t. The smaller it is, the closer they are. Given the GeoCast construction scheme ensures that a node can reside close to its physical network neighbors in GeoCast network with high probability, choosing a replica node located near tree nodes as replacement node should be a better choice to repair the broken path. It allows the new replacement node to directly subscribe to its physical network neighbors that are already in the multicast group.

By comparing $\mathrm{SPF}(t)$ among replica nodes, the node with maximum SPF is selected to take over all failed node's multicast functionalities. To notify such event, update messages are sent out to related nodes so that the new tree node can be included in the tree path and cancel messages are also delivered to other replicas to release the useless replications.

5.3.2 Region Recovery

In this step, we seek to find a living node to take over the service of the *island*. Rather than searching for the leaf node of the island, we try to merge the free region locally with the help of failed node's neighbors.

Firstly, let the immediate neighbor with smallest region (called *agent*) check if it can merge the island into its own region. If so, do it and advertise such event to all related nodes around (similar to [27], in our scheme, only sibling nodes are allowed to be merged so as to keep the system consistency). Otherwise, merging request is initiated and broadcasted to all its neighbors. Those nodes then check theirs region to see if two of them can be merged. If so, merge them and form a new region. Such region inherits all services offered by those two original separated regions and is assigned to one of two owners by considering theirs capacity. It is intuitive that the node with higher available capacity should have a larger region. Correspondingly, the other owner takes over the island. It initializes itself to be a simple node. However, it is common that there is no sibling nodes among those 1-hop neighbors. In such a case, more neighbor nodes on the next hop are considered. This process is repeated until the new region owner is found. If more than one pair of siblings are around island region, the pair with highest *unit_capacity* is elected to handle such failure.

We use Fig.3 to illustrate an example of region recovery in GeoCast consisting of 23 nodes. To get a better perspective, we depict the virtual space as a tree structure. Two leaves are siblings when they share the same parent. In Fig.3, E_{12} and E_{14} are siblings. If E_{12} departs, leaf node E_{14} is used to handle E_{12}'s failure. It merges with island into a single region. For E_{17}, leaf node E_{20} is elected to repair E_{17}'s failure in [27]. This is due to no sibling node can be found to recover its failure. But if E_{20} also leaves at that time, the recovery of E_{17} will be interrupted. This procedure cannot be continued until the leaf node is recovered first. To avoid that, in our scheme, we issue merging request to search two living sibling nodes for region recovery. After E_{15} merges E_{13}'s region, E_{13} can take over E_{17}'s region.

In terms of nodes' functionality in the system, there are three scenarios.

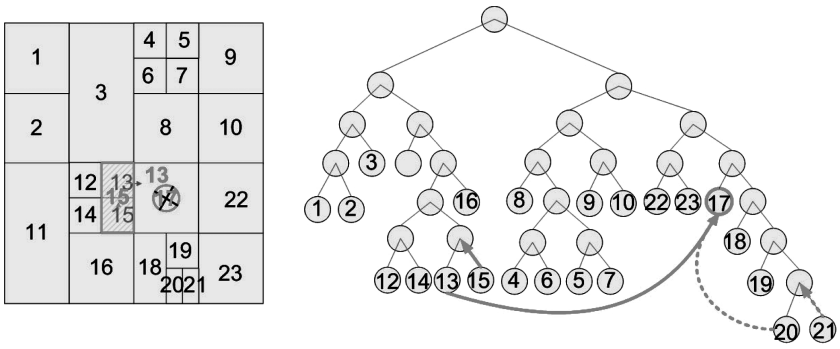

Fig.3. Tree structure of group communication session.

• *Path Recovery and Region Recovery.* This process is initiated by the tree node failures. Both of region recovery and path recovery are needed when a tree node moves out of system. In particular, region recovery in our scheme is conducted in parallel with path recovery so as to avoid the delay of searching a living node to taking over the island region. It allows the service interruptions to be resorted within a short delay.

• *Region Recovery and Replica Creation.* This is for the replica nodes, but not the tree nodes. Only region recovery is needed in this case. The owner of replica node would reuse the replication scheme and add new replica nodes into its *replicatlist*, as mentioned earlier.

• *Region Recovery Only.* This is activated when the simple node fails, which has no responsibility for multicast services or for the tree node failure recovery.

The advantages of the proactive failure recovery are two-fold. First, it achieves fast failure recovery by utilizing the independence between service recovery and connectivity recovery. It avoids the delay of finding a new node to restore the service from interruption, which is especially important for the real time applications like location-based notification. Second, the recovery overhead is greatly reduced by avoiding invoking the operations of service un-subscription and re-subscription. Given the experimental results shown in Section 7, we note that it is worthwhile to pay a small amount of maintenance overhead for achieving fast failure recovery while eliminating the overhead of service overhead.

6 Discussion

In this section, we briefly discuss several important issues related to the design of the schemes.

File vs Chunk Replication. For simplicity, file replication is employed in this study. We assume each node in the network has a unique collection of multicast session information and small data files that are treated as indivisible objects with uniform size. Those files represent the knowledge of the node about the network (like the traffic state in a vehicle's local region). For any two nodes in the network, it satisfies the condition that to there is no interaction between their collections. For simplicity, file replication is employed in this study. Without loss of generality, the files can be replicated at an unequal number of times, which depends on the hotness of the data files. If there is no node interesting the file of the node, only the multicast session information on the node is replicated to improve node capacity utilization.

The benefit of this scheme is two folds. First, the global cost of deploying replication mechanism is reduced as the tree nodes only place their copies. There might be a vast majority of the nodes that never need to be replicated throughout the whole procedure. Second, it enables nodes with data copies to easily keep consistent with the host nodes without introducing high overhead since only a small amount of information is needed to be updated each time. For the chunk replication, we view it as a component used to further improve the system performance that will be described in detail in the next study.

Periodic vs Emergency Replication. To reduce the replication cost and improve the efficiency of replication, scheduled replica replacement is preferred since this gives the nodes more flexibility in determining the strategy of replica replacement, thus producing better performance. However, this comes with the expense of efficiency (i.e., the average response time of node failure) and reliability (i.e., the number of service interruptions). This obviously obeys the design of the replication algorithm. Therefore, emergency replication is resorted in our work.

Parameter Setting. α, r, K are the parameters introduced to control the tradeoff between the efficiency of the replication and the cost of replica creation and maintenance. Intuitively, the setting of smaller value of α and lager values of r and K provides a better performance at the expense of higher cost. To reduce the cost of replica creation and maintenance, we define the parameter α as in (1). It allows the nodes to dynamically adapt α to the state of network and the requirements

of the system. We use the parameter r to refer to the number of replica nodes placed by the nodes for high reliability. The larger it is, the higher reliability will be achieved. Given the fact that the service reliability can be greatly improved by the use of a small number of replica nodes, we set $r = 4$, which enables $R_s(t) = 1 - \prod_r^{i=1}(1 - P_i(t)) = 0.97$ with $F(x) = 1-(1+x/0.05)^{-1.1}$ and $\text{Min}\{P_i(t)\} = 0.5$, where $R_s(t)$ and $P_i(t) = \frac{\int_0^{t+\tau}(1-F(s))\mathrm{d}s}{\int_0^t(1-F(s))\mathrm{d}s}$ denote the reliability of service node p with *replicalist* $RPS = \{E_1, E_2, \ldots, E_r\}$ and the probability of node i in set RPS remaining in the network in next time slot τ. It is clearly seen that the service reliability is greatly improved by the use of a small number of replica nodes. This will also be confirmed by our experimental results in Section 7. The parameter K is a system constant which is configured by default. The purpose of introducing K is to limit the cost of replica selection within a certain level. As suggested by [29], we set it to 2 in order to limit the replication searching cost within $O(4nd^2 - 2nd)$, where n is the network size can d is the dimension of the GeoCast space.

So far, all our efforts attempt to give a better understanding of the new concepts about the replica placement. In fact, it is possible to optimize the performance of replication scheme by combining with one of the threshold algorithms[30-31], rather than just statically configure those parameters as suggested by [19, 26, 29]. We will detail this optimization in our next work.

7 Performance Evaluation

In this section, we perform extensive experiments to study the efficiency of our schemes proposed in this paper.

7.1 Experimental Environment

The experiments in this paper run on 10 topologies that are generated by the GT-ITM topology generator based Transit-Stub graph model[2]. Each topology contains 8 080 routers. At the top level, there are 10 transit domains. In each domain, it contains 8 routers on average, each of which is attached by about 10 stub domains. According to the types of routers, we assign the link latencies by following a uniform distribution on different ranges: [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.

A number of end system nodes with heterogeneous capacities are randomly attached to the routers and organized into the GeoCast overlay network. Similar to [8], the nodes in the network are assigned with various resource capacities: 5% nodes have 1 000 units of capacity, 15% nodes possess 100 units of capacity, 30% nodes have 10 units of capacity, and the rest of nodes has 1 unit of capacity. The processing capacity of node is proportional to its resource capability. Each unit of resource capacity allows nodes to maintain 10 files in their local memory.

To simulate the service interruptions, a sequence of node failures is generated. By following independent and identical exponentially distribution, the failure time of sequence nodes that is randomly chosen is assigned, which ranges from $[0, 60T]$ simulated seconds. $T = 120$ is the parameterized heartbeat period.

7.2 Results

In the following subsection, we focus on investigating three main subjects: the effect of multicast construction scheme on the system performance, the efficiency of replication scheme, and the performance of failure recovery scheme.

7.2.1 Effect of Multicast Construction Scheme

Fig.4 compares GeoCast with CAN-like system

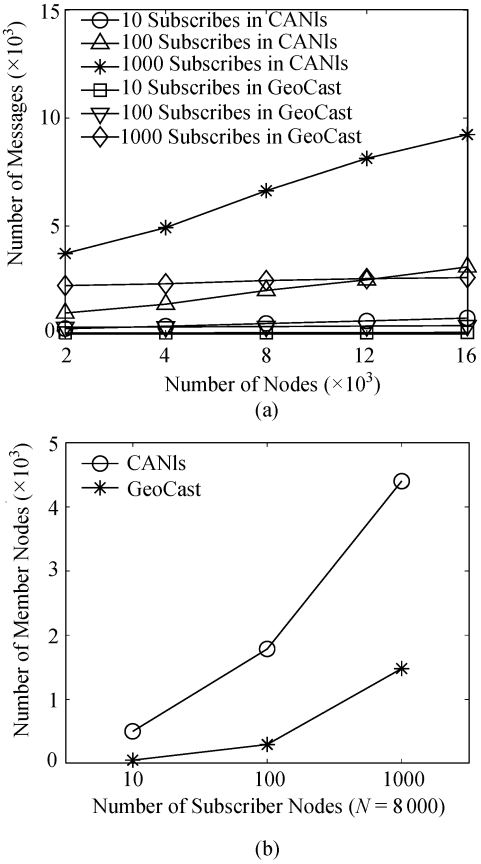

Fig.4. Overhead of multicast group construction. (a) Different system size. (b) Different group size.

(CANls)[35-36] in terms of multicast construction cost. We vary system size (N) from 2 000 to 16 000, group size from 10 to 1 000, and set the group number to 50. Fig.4(a) shows the multicast construction cost of the schemes as a function of system size, where multicast construction cost and system size represent the number of messages generated for multicast tree construction and the number of nodes in GeoCast respectively. We observe that GeoCast multicast mechanism produces much less overhead than neighbor-based mechanism[27]. Unlike CANls, it exhibits a flat curve. As the group size increases, the differences between GeoCast and CANls become more obvious. This benefits from the efficient routing strategy employed by GeoCast. It can always deliver the message to destination within a few steps. While neighbor-based approach employed by CANls has to involve a great quantity of member nodes in the routing path. In such a case, the nodes' resources in CANls are wasted.

Fig.4(b) shows how the multicast construction scheme affects the number of nodes involved in the multicast. We can see that GeoCast multicast mechanism constantly outperforms its competition. It includes only a few tree nodes in each multicast group even when there are 1 000 subscribe nodes in the group. This potentially confirms the reason mentioned before. Given that, we discuss our proposed schemes based on GeoCast in the following.

7.2.2 *Effect of Replication*

To improve service reliability in the presence of failures in different pattern, we introduce location-aware replication scheme (called NS) into GeoCast. Such a scheme enables nodes to adaptively make replicas with consideration of both replication cost and service reliability. Fig.5 shows the effect of the proposed replication scheme on the quality of services in terms of service interruption number. RT refers to the time required for node initialization right after it is selected as replacement node to take over the *island* previously owned by the failed node. r refers to the number of replica nodes that are required to be placed by each node in the network. To simulate the node failures occurring in the network, a failure sequence is generated, where each node of the sequence is selected from the system node set randomly and assigned a failure time by following independent and identical exponentially distribution. We use the term failure probability (FP) to denote the faction of nodes failed at the runtime. The value of FP ranges from 0 to 1. For example, if FP is set to 0.60 and system size is set to 16 000, 9 600 nodes will leave from the system at runtime.

From the results in Fig.5, we observe three interesting facts. First, with growth of RT, the number of service interruptions increases. This is because the probability of both the nodes and their replicas fail in a short time interval is higher when RT is larger. In such a case, more multicast services interruptions occur and greatly reduce the quality of services. It is common that the messages disseminated along the multicast trees may suffer from long latency before reaching the destination, and a large amount of re-subscription requests is issued for service recovery. To avoid this, lower RT is preferred. Second, we find that the number of service interruptions significantly drops as the time increases. This is due to the fact that majority of service interruptions occur at the beginning stage of simulation. Third, it is important to note that the metric become smaller when r is larger. When it increases to 4, the number of service interruptions is steadily below 10, even with $RT = 80$ s. It indicts the effectiveness of the proposed replication scheme.

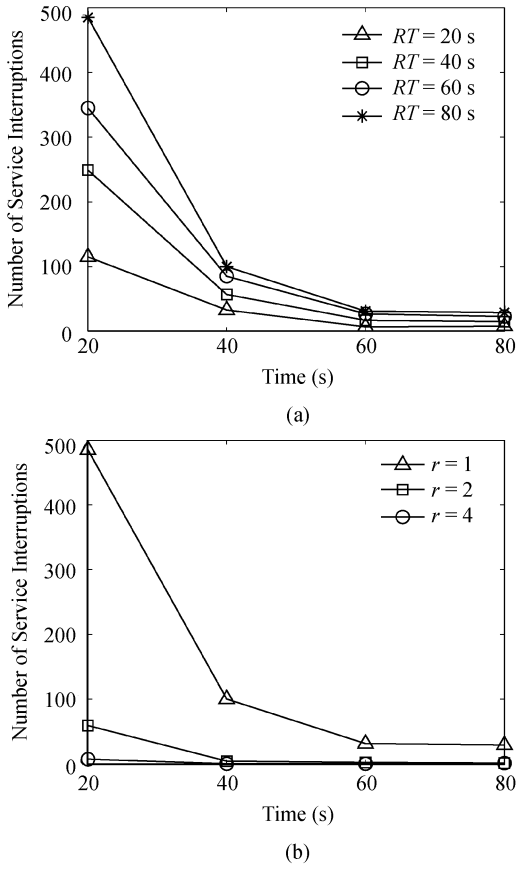

Fig.5. Service interruption during the service time. (a) $r = 1$ (FP = 0.6). (b) r (FP = 0.6 $RT = 80$ s).

To provide insight into the efficiency of the proposed replication scheme, we compare it to three schemes, the neighbor-based replication scheme (Neighbor)[8,32], the random replication scheme (Random)[33-34], and the

hybrid replication scheme (Neighbor & Random).

The major difference between those schemes is in the replica creation: 1) Neighbor: it favors to place data copies on the nodes located in adjacent area so that the replica creation and management can be proceeded while keeping overhead low; 2) Random: in the random replication, each node sends replica creation requests to a set of nodes in a random manner, and chooses the nodes with available capacities as replica nodes according to the responses; 3) Neighbor & Random: it is simply an extension of Neighbor and Random, which is exploited to improve the service reliability provided by Neighbor and reduce the high cost incurred by Random. In this scheme, neighbor and random replicas are created, where the exploring of random replicas allows the services provided by the unreliable nodes to have a higher probability of surviving than that of neighbor replicas.

Fig.6 shows the success rate for a varying system size. Success rate refers to the ratio of the number of *recoverable failure*s to the number of failures happened at runtime, where recoverable failure refers to the node failures that can be restored by one of its replicas when the node failure occurs. For a system with a certain number of node failures, the larger recoverable failure is, the higher success rate it has. The results in Fig.6 lead to two observations. First, Random yields a better performance than the other schemes. This benefits from the exploiting of random replicas. Those nodes have higher probability to remain alive even facing high churn rates. Second, compared to Neighbor & Random and NS, Neighbor performs poorly in terms of success rate. This is because the nodes in Neighbor are likely to place replicas on the neighbor nodes. It leads the services on the top of nodes to be vulnerable to the centralized failures. In such a situation, it is less likely to restore the services from interruptions by searching the living replica.

As the results shown in Fig.5, we find that the performance of the replication schemes with parameter heavily depends on the setting of parameter r. The larger r is, the more messages are generated. This reduces the network resource utilization. To avoid that, an efficient replication scheme should be able to employ "enough" replicas while keeping the cost low. In the following experiment, we study how the system size and replica number affect the replica creation cost for the replication schemes. In Fig.7, we observe that both

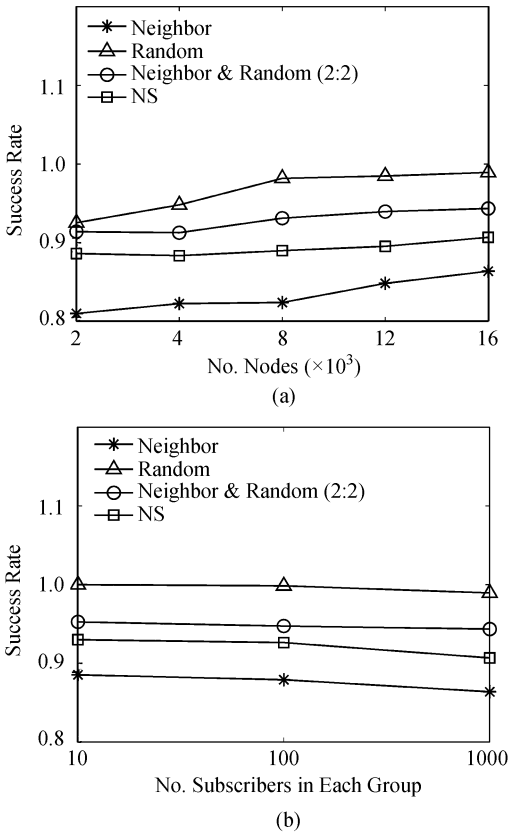

Fig.6. Success rate (FP = 0.6, RT = 80 s). (a) Different system size. (b) Different group size.

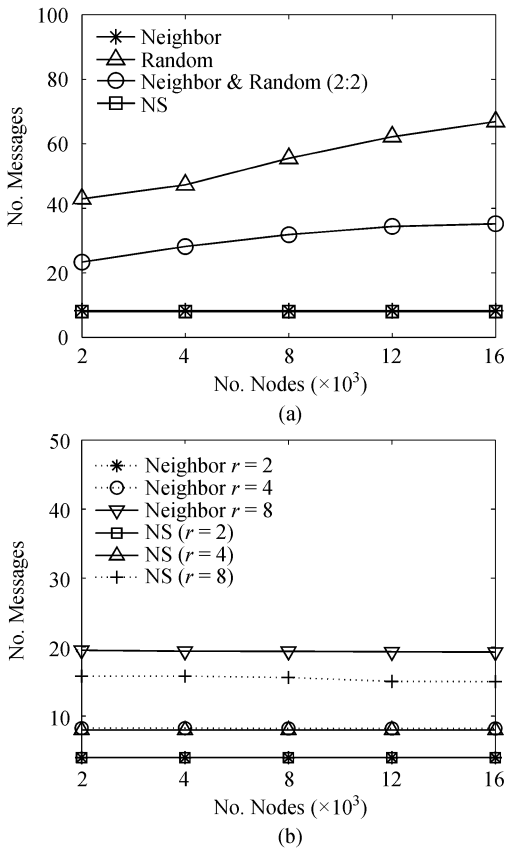

Fig.7. Overhead of replica creation. (a) Different scheme. (b) Different r.

Neighbor and NS achieve better performance than the others in overhead. This is because the majority of qualified replicas are discovered within 2 levels. But with the growth of r, the overhead of Neighbor increases. In such a case, more request messages are issued from the nodes for replica selection, which may lead to a poor application performance in scalability. Instead of sending more request messages, NS adaptively tunes its parameter and makes more replicas on the shortcut nodes. It not only reduces the replica selection cost, but also enhances the reliability of services in some sense. In addition, we notice that the number of the service interruptions is steadily small when setting r to 4, as shown in Fig.5. Thus, we set r to 4 in the following discussion.

7.2.3 *Effect of Failure Recovery Scheme*

Fig.8 depicts the recovery latency as a function of failure probability FP. Recovery latency refers to the interval between the time when a node detects a failure event to the time when the new responsible node offers to serve as the owner of island. We vary FP from 0.2 to 0.8 in a 16 000-node network to evaluate the performance of different recovery schemes in a dynamic environment. We use LS and NBS to denote the leaf-based and neighbor-based failure recovery scheme, respectively. In Fig.8, we observe that NBS steadily achieves better performance than LS. In all cases, NBS needs to take less time to recover the interrupted service than LS. With the growth of FP, the performance improvement of NBS is more pronounced. This indicts that NBS is more robust than its competitor in dealing with the node failures.

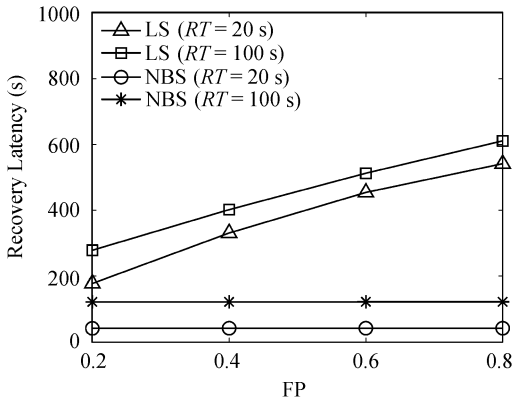

Fig.8. Effect of FP on region recovery latency.

Fig.9 provides a better understanding of the efficiency of NBS using three metrics: maximum, minimum, and average recovery latency. In particular, the maximum recovery latency refers to the time required for recovering services from interruption in the worst case. The results show that NBS greatly reduces the maximum recovery latency than LS. This is because of its two unique features. First, NBS conducts path recovery in parallel with region recovery, which reduces the recovery latency of services. Second, the mechanism of island owner selection in NBS enables the regions to be re-adopted within a small area, which limits the communication cost caused for searching qualified node and shorts the service recovery time, as shown in Table 1. RF and URF refer to the recoverable failure and unrecoverable failure, respectively. We can see that all node failures can be recovered within 3 hops in all cases. For LS, when we set FP = 0.8, 41.08% of the failures is unrecoverable at runtime. This is because that the failures cannot be recovered until the alternative nodes are fetched in LS.

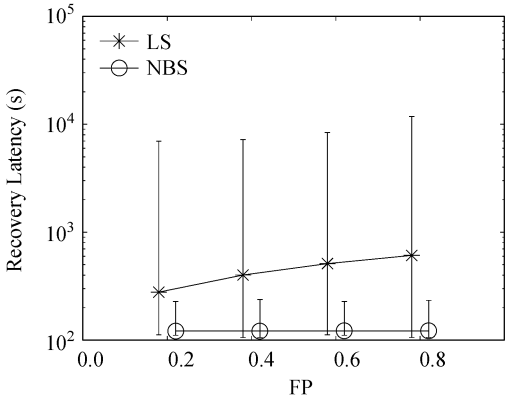

Fig.9. Max and Min region recovery latency ($RT = 20$ s).

Table 1. Success Ratio Comparison

			FP			
			0.2	0.4	0.6	0.8
NBS	Hop	1	86.70%	85.80%	85.70%	85.57%
		2	100%	100%	100%	99.99%
		3	100%	100%	100%	100%
		4	100%	100%	100%	100%
			0.2	0.4	0.6	0.8
LS	Type	RF	60.75%	59.90%	59.60%	58.92%
		URF	39.25%	40.10%	40.40%	41.08%

8 Conclusions

In this paper, we present two schemes to support reliable and scalable group communication services in a distributed manner. They are derived from the two observations. The first one is that the location of replicas plays an important role in the strategies of replica placement. In general, the replicas located in adjacent area have advantages in reducing the replication overhead while the remote replicas are more likely to remain alive even facing high churn rates. The second one is that the efficiency of failure recovery dominates

the quality of the services throughout the whole group communication session. By taking into consideration replica location and the independence between service recovery and structure recovery in time domain, our proposed schemes improve the reliability of the services while keeping the overhead and recovery latency low. Finally, we demonstrate the effectiveness and efficiency of the purposed schemes using large-scale simulations. Our work on fault tolerance and recovery for group communication services in distributed networks continues along three directions. First, instead of deterministically deciding the value of the number of replicas, an adaptive threshold algorithm will be explored to improve the flexibility and practicability of the proposed replication scheme. Second, in this work, we focus on how to efficiently place data copies on the neighbor and shortcut nodes and use them to achieve quick failure recovery. One interesting problem for future research is to study the relationship of replica nodes in the distributed network for eliminating redundancy in replica placement and synchronization. Third, performing real time service failure recovery is another interesting yet challenging problem. To address this, the characteristics of real time services will be needed to be identified and intelligently utilized in the procedure of replica placement.

References

[1] Chu Y, Rao S G, Seshan S, Zhang H. A case for end system multicast. *IEEE Journal on Selected Areas in Communications*, 2002, 20(8): 1456-1471.

[2] Castro M, Druschel P, Kermarrec A M, Rowstron A I T. SCRIBE: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications*, 2002, 20(8): 1489-1499.

[3] Chawathe Y. Scattercast: An adaptable broadcast distribution framework. *Multimedia Systems*, 2003, 9(1): 104-118.

[4] Francis P. Yoid: Extending the internet multicast architecture. http://www.aciri.org/yoid/docs/index.html, 2000.

[5] Banerjee S, Bhattacharjee B, Kommareddy C. Scalable application layer multicast. In *Proc. SIGCOMM 2002*, Pittsburgh, USA, Aug. 19-23, 2002, pp.205-217.

[6] Banerjee S, Kommareddy C, Kar K, Bhattacharjee B, Khuller S. OMNI: An efficient overlay multicast infrastructure for real-time applications. *Computer Networks*, 2006, 50(6): 826-841.

[7] Jannotti J, Gifford D, Johnson K, Kaashoek M *et al.* Overcast: Reliable multicasting with on overlay network. In *Proc. OSDI 2000*, San Diego, USA, Oct. 23-25, 2000, pp.197-212.

[8] Zhang J, Liu L, Ramaswamy L, Pu C. PeerCast: Churn-resilient end system multicast on heterogeneous overlay networks. *Journal of Network and Computer Applications*, 2008, 31(4): 821-850.

[9] Castro M, Druschel P, Kermarrec A, Nandi A, Rowstron A, Singh A. SplitStream: High-bandwidth multicast in cooperative environments. In *Proc. SOSP 2003*, Bolton Landing, USA, Oct. 19-22, 2003, pp.298-313.

[10] Kostić D, Rodriguez A, Albrecht J, Vahdat A. Bullet: High bandwidth data dissemination using an overlay mesh. *ACM SIGOPS Operating Systems Review*, 2003, 37(5): 282-297.

[11] Zhang X, Liu J, Li B, Yum T. CoolStreaming/DONet: A data-driven overlay network for efficient live media streaming. In *Proc. INFOCOM 2005*, Miami, USA, Mar. 13-17, 2005, pp.13-17.

[12] Pai V, Kumar K, Tamilmani K, Sambamurthy V, Mohr A. Chainsaw: Eliminating trees from overlay multicast. *Peer-to-peer systems IV*, 2005, pp.127-140.

[13] Tran D A, Hua K A, Do T. Zigzag: An efficient peer-to-peer scheme for media streaming. In *Proc. INFOCOM 2003*, San Franciso, USA, Mar. 30-Apr. 3, 2003, pp.1283-1292.

[14] Gu X, Nahrstedt K, Yu B. SpiderNet: An integrated peer-to-peer service composition framework. In *Proc. HPDC 2004*, Honolulu, Hawaii, USA, Jun. 4-6, 2004, pp.110-119.

[15] Wang Y, Liu L, Pu C, Zhang G. GeoCast: An efficient overlay system for multicast applications. Technical Report, Georgia Institute of Technology, 2009, http://www.cercs.gatech.edu/tech-reports/tr2009/git-cercs-09-16.pdf.

[16] Wen C, Wu C, Yang M. Hybrid tree based explicit routed multicast for QoS supported IPTV service. In *Proc. GLOBECOM 2009*, Honolulu, Hawaii, USA, Nov. 30-Dec. 4, 2009, pp.1-6.

[17] Fei A, Cui J, Gerla M, Cavendish D. A "dual-tree" scheme for fault-tolerant multicast. In *Proc. ICC 2001*, Helsinki, Finland, Jun. 11-14, 2001, pp.690-694.

[18] Banerjee S, Lee S, Bhattacharjee B, Srinivasan A. Resilient multicast using overlays. *ACM SIGMETRICS Performance Evaluation Review*, 2003, 31(1): 102-113.

[19] Gopalakrishnan V, Silaghi B, Bhattacharjee B, Keleher P. Adaptive replication in peer-to-peer systems. In *Proc. ICDCS 2004*, Tokyo, Japan, Mar. 23-26, 2004, pp.360-369.

[20] Yoshinaga H, Tsuchiya T, Sawano H, Koyanagi K. A study on scalable object replication method for the distributed cooperative storage system. In *Proc. ICDT 2009*, Colmar, France, Jul. 20-25, 2009, pp.96-101.

[21] Sandhu H S, Zhou S. Cluster-based file replication in large-scale distributed systems. *ACM SIGMETRICS Performance Evaluation Review*, 1992, 20(1): 91-102.

[22] Shen H, Zhu Y. A proactive low-overhead file replication scheme for structured P2P content delivery networks. *Journal of Parallel and Distributed Computing*, 2009, 69(5): 429-440.

[23] Tirado J M, Higuero D, Isaila F, Carretero J, Iamnitchi A. Affinity P2P: A self-organizing content-based locality-aware collaborative peer-to-peer network. *Computer Networks*, 2010, 54(12): 2056-2070.

[24] Ho C, Lee S, Yu J. Cluster-based replication for P2P-based video-on-demand service. In *Proc. ICEIE 2010*, Kyoto, Japan, Aug. 1-3, 2010, pp.49-53.

[25] Zhao K, Niu Z, Zhao Y, Yang J. Search with index replication in power-law like peer-to-peer networks. In *Proc. ICCET 2010*, Chengdu, China, Apr. 16-18, 2010, pp.334-338.

[26] Zhang J, Liu L, Pu C, Ammar M. Reliable peer-to-peer end system multicasting through replication. In *Proc. P2P 2004*, Zurich, Switzerland, Aug. 25-27, 2004, pp.235-242.

[27] Ratnasamy S, Francis P, Handley M, Karp R, Schenker S. A scalable content-addressable network. In *Proc. SIGCOMM 2001*, San Diego, USA, Aug. 27-31, 2001, pp.161-172.

[28] Yamamoto H, Maruta D, Oie Y. Replication methods for load balancing on distributed storages in P2P networks. *IEICE Transactions*, 2006, E89-D(1): 171-180.

[29] Kalogeraki V, Gunopulos D, Zeinalipour-Yazti D. A local search mechanism for peer-to-peer networks. In *Proc. CIKM 2002*, Nov. 5-8, 2002, pp.300-307.

[30] Ganesan P, Bawa M, Garcia-Molina H. Online balancing of range-partitioned data with applications to peer-to-peer systems. In *Proc. VLDB 2004*, Toronto, Canada, Aug. 31-Sep. 3, 2004, pp.444-455.

[31] Sato H, Matsuoka S, Endo T, Maruyama N. Access-pattern and bandwidth aware file replication algorithm in a grid environment. In *Proc. Grid 2008*, Tsukuba, Japan, Sep. 29-Oct. 1, 2008, pp.250-257.
[32] Chang T, Ahamad M. Improving service performance through object replication in middleware: A peer-to-peer approach. In *Proc. P2P 2005*, Konstanz, Germany, Aug. 31-Sep. 2, 2005, pp.245-252.
[33] Lv Q, Cao P, Cohen E, Li K, Shenker S. Search and replication in unstructured peer-to-peer networks. In *Proc. ICS 2002*, New York, USA, Jun. 22-26, 2002, pp.84-95.
[34] Liu Y, Liu X, Xiao L, Ni L, Zhang X. Location-aware topology matching in P2P systems. In *Proc. INFOCOM 2004*, Hong Kong, China, Mar. 7-11, 2004, pp.2220-2230.
[35] Falchi F, Gennaro C, Zezula P. A content-addressable network for similarity search in metric spaces. In *Proc. DBISP2P 2005/2006*, Trondheim, Norway, Aug. 28-29, 2005, pp.98-110.
[36] Sahin O D, Gupta A, Agrawal D, El Abbadi A. A peer-to-peer framework for caching range queries. In *Proc. ICDE 2004*, Boston, USA, Mar. 30-Apr. 2, 2004, pp.165-176.

Yue-Hua Wang is a Ph.D. candidate in School of Computer Science and Engineering at Beihang University, China. She received the B.S. and M.S. degrees in electrical engineering from North University of China in 2003 and 2006 respectively. Her research interests currently lie in availability, reliability, and scalability issues in large-scale distributed systems such as P2P networks and Cloud datacenters. She is a student member of IEEE.

Zhong Zhou is an associate professor at State Key Lab of Virtual Reality Technology and Systems, Beihang University, Beijing, China. He received his B.S. degree from Nanjing University and Ph.D. degree from Beihang University in 1999 and 2005 respectively. His main research interests include tele-immersion, natural phenomena simulation, distributed virtual environment and Internet-based VR technologies. He is member of CCF, ACM and IEEE.

Ling Liu is a full professor in the School of Computer Science at Georgia Institute of Technology. There she directs the research programs in Distributed Data Intensive Systems Lab (DiSL), examining various aspects of data intensive systems with the focus on performance, availability, security, privacy, and energy efficiency. Prof. Liu and her students have released a number of open source software tools, including GTMobiSim, PeerCrawl, WebCQ, XWRAPElite. Prof. Liu has published over 300 International journal and conference articles in the areas of databases, distributed systems, and Internet computing. She is a recipient of the Best Paper Award of ICDCS 2003, WWW 2004, the 2005 Pat Goldberg Memorial Best Paper Award, and 2008 Int. Conf. Software Engineering and Data Engineering. Prof. Liu has served as general chair and PC chair of numerous IEEE and ACM conferences in data engineering, distributed computing, service computing and cloud computing fields and is a co-editor-in-chief of the 5th volume Encyclopedia of Database Systems (Springer). She is currently on the editorial board of several international journals, such as Distributed and Parallel Databases (DAPD, Springer), Journal of Parallel and Distributed Computing (JPDC), IEEE Transactions on Service Computing (TSC), ACM Transactions on Web (TWEB) and Wireless Network (WINET, Springer). Dr. Liu's current research is primarily sponsored by NSF, IBM, and Intel. She is a senior member of IEEE.

Wei Wu is a professor in School of Computer Science and Engineering at Beihang University, currently the chair of TCVRV (Technical Committee on Virtual Reality and Visualization) of CCF (China Computer Federation). He received the Ph.D. degree from Harbin Institute of Technology, China, in 1995. He has published more than 90 papers, 33 issued patents and 1 book. His current research interests involve real-time 3D reconstruction, remote immersive system and augmented reality.