Journal of Network and Computer Applications 36 (2013) 388–401

Contents lists available at SciVerse ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Replica-aided load balancing in overlay networks

Yuehua Wang[a,b], Zhong Zhou[a,b,*], Ling Liu[c], Wei Wu[a,b]

[a] State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing 100191, China
[b] School of Computer Science and Engineering, Beihang University, Beijing 100191, China
[c] College of Computing, Georgia Institute of Technology, Atlanta, GA30332, USA

ARTICLE INFO

Article history:
Received 18 June 2011
Received in revised form
15 February 2012
Accepted 11 May 2012
Available online 22 May 2012

Keywords:
Load balancing
Cost model
Load fluctuation
Overlay network
End-user communication

ABSTRACT

In recent years, there have been rapid advances in network infrastructure and technologies for end-user communication. However, because of network dynamics and resource limitation, providing scalable end-user communication services is challenging when the applications are utilized on a large-scale. To address this challenge, a replica-aided load balancing scheme (RALB) is proposed for enabling the nodes in an overlay networks to support the communication applications for a large number of users. This paper makes three unique contributions. First, we study the existing load balancing schemes and identify their weakness in handling time-varying workloads with frequent load fluctuations. Second, we introduce a sophisticated cost model for load balancing cost estimation, which captures the dependencies between the factors (e.g., the load, message number, and link latency). Third, we propose a performance tuning technique to minimize the load balancing cost. The extensive experiments show that RALB effectively reduces the load imbalance and eliminates the load balancing cost when compared to the existing load balancing schemes.

1. Introduction

With the development of communication technology and the rapidly growing popularity of hardware devices (e.g., PDAs, smart phones, game consoles and computers), the ways of accessing Internet services and applications (Kumar et al., 2003; Myles et al., 2003; ComScore, 2007; Technologyreview, 2010; Techcrunch, 2011) (e.g., e-commerce, web surfing, instant messaging, file fetching, online game, email and multimedia dissemination) have changed significantly. In many of these applications, a large number of hardware device users share the resources and communicate with each other anywhere and anytime. To meet the voluminous demand of the users, service providers usually construct overlay networks with a large number of nodes (servers) and spread applications over the overlay networks (Knoll et al., 2007; Cannataro and Talia, 2003; Delmastro et al., 2008; Avancha et al., 2002; INTRICE, 2009; Warcraft, 2010; Google, 2010). Typically, the users with hardware devices connect to the nearest nodes (servers), and fetch (publish) the data that are interesting. However, the nodes (servers) usually have are heterogeneous (homogeneous) and bounded capacities. If one of the nodes (servers) is overloaded, then it could become a bottleneck and degrade the system resource utilization and the service quality (e.g., the loss rate and response time). In such a case, load balancing is often used to alleviate the bottleneck so that the system performance in terms of utilization and service quality can be improved.

The primary goal of load balancing in a network is to balance the workload of the nodes in proportion to their capacities, which is measured in terms of their processor speed, storage capacity, and/or bandwidth. However, this task is inherently difficult because of the following: (1) the *network dynamics*, e.g., some nodes or network links could become unavailable because of a node crash or improper program termination; and (2) the *time-varying load distribution*, e.g., the load distribution can be changed as the popularity of files stored on the nodes changes. Usually, the nodes with popular files or data often have much heavier loads than the nodes with unpopular ones; and (3) the *frequent load fluctuations*, e.g., frequent load movements between nodes could be caused, which could further lead to expensive and problematic load balancing.

To solve those issues, a substantial amount of work has been performed recently on load balancing among the nodes in a dynamic network. Examples include Proximity-aware load balancing (Zhu and Hu, 2005), Address-space balancing (Karger and Ruhl, 2006), Multi-parameter load balancing (Werstein et al., 2006), Adaptive replication (Gopalakrishnan et al., 2004), Proportional replication (Tewari and Kleinrock, 2006), Cluster-based load balancing (Shen and Xu, 2008; Qiao and Bochmann, 2009; Kotoulas et al., 2010), and load balancer based load balancing (Ranjan et al., 2010). However, most of the existing load balancing approaches have some limitations in our opinion. They either have a high communication cost during the load balancing or they ignore the impact of the load fluctuation on the system performance. In fact, the frequent fluctuations in the load are

* Corresponding author. Tel.: +86 10 82313085; fax:+ 86 10 82339909.
E-mail addresses: yuehua.research@gmail.com (Y. Wang), zz@vrlab.buaa.edu.cn (Z. Zhou), lingliu@cc.gatech.edu (L. Liu), wuwei@vrlab.buaa.edu.cn (W. Wu).

1084-8045/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.jnca.2012.05.001

very common for Internet applications (e.g., online games, multimedia dissemination, and web surfing). Surprisingly, we find that most of the existing work to date (Byers et al., 2003; Gopalakrishnan et al., 2004; Werstein et al., 2006; Tewari and Kleinrock, 2006; Qiao and Bochmann, 2009; Kotoulas et al., 2010; Ranjan et al., 2010) addresses only the load balancing without considering the frequent load fluctuation. One intuitive approach, as mentioned in Kwon and Ryu (2004), is to keep track of the load status of the nodes during a time window in which the load movement is triggered when a predefined load threshold is violated. Although this approach makes the algorithm design easier, it could come at the cost of degraded performance in the network resource utilization and the service quality. We argue that in some cases, such approach can result in a burst of messages that are sent to the network system for maintenance, and a loss of the messages transmitted between the nodes from an unsuitable threshold setting.

In this paper, a novel load balancing scheme is proposed to achieve load balancing efficiency and to minimize the cost in the spite of a skewed load distribution and a frequent load fluctuation in the overlay networks. Such a scheme can empower the nodes to support the communication applications over the overlay networks for a large and growing number of end-users. Each end-user can access the services of an overlay network by connecting to one of the nodes in the network, usually through a wireless or wired network connection. Those nodes run the middleware and serve as proxies for the end-users. In general, this paper makes three unique contributions.

First, we study the existing load balancing schemes and identify their weaknesses in dealing with time-varying workloads with frequent load fluctuations. To overcome those weaknesses, a replica-aided load balancing scheme (RALB) is proposed, which deploys replica nodes for load sharing and intelligently avoids a large volume of load movements to achieve load balancing efficiency. In RALB, load movement is triggered only when there is no light-load replica node that can be used to process the extra load migrated from the others. This strategy greatly eliminates the influence of load fluctuation on the system performance.

Second, we introduce a new cost model for estimating the load balancing cost. This model captures the dependencies between the important performance-related factors (e.g., load, cost, message number, and link latency) in the load balancing.

Third, based on the cost model, we provide a performance tuning scheme to minimize the load balancing cost. This scheme empowers the nodes to explore different load balancing strategies and to choose the best strategy for the load balancing and cost minimization. The scheme that we presented in this paper, to the best of our knowledge, is the first method that provides a combination of techniques that allows nodes to adaptively make the best decisions by utilizing replicas to alleviate the impact of frequent load fluctuations and by introducing a sophisticated cost model to choose a strategy that minimizes the load movement cost.

Many communication applications could benefit from a system of nodes with our load balancing scheme, such as multimedia streaming, multiparty online games, interactive web surfing, and location-based advertising. For example, in the multimedia streaming, the streams with a higher popularity are often demanded by a large number of end-users. If only a few nodes with a limited capacity are employed to serve the voluminous demand of the end-users, the system performance would be significantly degraded in terms of resource utilization and service quality. To avoid this scenario, our scheme can be employed and can help the nodes to achieve load balancing while meeting the various demands of the end-users.

The rest of this paper is organized as follows. Section 2 surveys relevant literature and Section 3 defines the load balancing problem. In Section 4, we present the details of RALB. Section 5 describes our cost model and presents the performance tuning scheme. We introduce the experimental evaluation in Section 6 and conclude the paper in Section 7.

2. Related work

Many solutions have been proposed to tackle the load balancing issue in networks. Based on their strategies for load assignments, they are classified into two categories: centralized schemes and distributed schemes.

In a centralized scheme (Surana et al., 2006; Werstein et al., 2006; Zhu and Hu, 2005; Rao et al., 2003), head nodes are typically required for load information aggregation and load reassignment. At intervals, each node reports its latest load state to one head node; then, the head node performs load reassignment with respect to the load information of the nodes when there are nodes with heavy loads. To improve the robustness of the head node, a number of head nodes in Rao et al. (2003) are employed and the nodes in the network can randomly choose their head nodes to achieve better load balancing. The work conducted by Godfrey et al. (Surana et al., 2006) extends the method proposed by Rao et al. (2003) to provide an efficient load balancing method for the nodes in a dynamic environment, where data items could be continuously inserted or deleted, and nodes might join or leave the system at any point in time. The scheme in Zhu and Hu (2005) uses proximity information to guide load reassignment such that the loads are reassigned between physically close nodes. While this type of scheme has advantages in minimizing the difference between the nodes in resource utilization, it suffers from both scalability and reliability issues. In some cases, the overhead of information aggregation can be large, and the failures of the head nodes may lead to a complete failure of the load-balancing.

A distributed scheme addresses the load balancing issue in the overlay network from a different viewpoint, which strives to balance the load distribution among nodes intelligently based on the local knowledge of the nodes about the other nodes in the network. In Byers et al. (2003), a power of two choices approach is used to achieve load balance for which each data item is hashed into a small number of different IDs, which are then stored in the least loaded node among the nodes that are responsible for those IDs. Karger and Ruhl propose two load balancing protocols in Karger and Ruhl (2006). One protocol is to balance the loads by changing the distribution of the items among the nodes. It is useful when the distribution of items in the address space cannot be randomized. The other protocol is to balance the loads by changing the assignment of identifiers to the nodes. This strategy improves the consistency of the hash table. CAN (Ratnasamy et al., 2001) and GeoGrid (Zhang et al., 2007) attempt to address the load balancing issue through ID space reassignment. Steele et al. (2008) achieve load balancing with random peer selection. Hu et al. (2010) provide a scheduling strategy on virtual machine (VM) load balancing. According to historical data and the current state of the system, this strategy computes the influence it will have on the system after the deployment of the needed VM resources and then chooses the least-affective solution to reduce the dynamic migrations. In the above approaches, it is not clear how to deal with the frequent load fluctuations in a dynamic network. In fact, such load fluctuations could cause frequent and unnecessary load movements due to unpredictable traffic jitters and momentary fluctuations which are common in the underlay networks.

There are several methods that have been proposed to perform dynamic load balancing with clustering technology (Shen and Xu, 2008; Qiao and Bochmann, 2009; Kotoulas et al., 2010). Shen and Xu (2008) propose a hash based proximity clustering approach to address the load balancing problem in heterogeneous DHTs. Such an approach groups the low-capacity nodes and allocates them to the nodes in the system that have high capacities via consistent hashing of their physical proximity information on the Internet. In Qiao and Bochmann (2009), a global balance is achieved through balancing the neighborhoods of all of the clusters within the existing overlay network. Kotoulas et al. (2010) address the load balancing problem with a method for data distribution that is based on clustering in elastic regions. The data are not deterministically routed to one fixed peer; instead, the data are attracted to a region of peers. In contrast, GOGRID (2010) provides hardware load balancing services using state-of-the-art F5 load balancer hardware. The load balancer provides the round robin algorithm and least connect algorithm for routing application service requests.

In Gopalakrishnan et al. (2004), Yamamoto et al. (2006), Roussopoulos and Baker (2006), Xia et al. (2009), and Pitoura et al. (2010), replication-based schemes are used for load balancing. Their main concept is to place multiple object replicas to maximize the availability of the data (file) in the network instead of shifting the sole responsibility for the popular items to a more powerful node. This type of strategy is derived from the fact that highly skewed data access behaviors could lead to uneven load placements, and the popular data could fall into a weak node that is not capable of dealing with the high demands for the data. Although replication is a good solution in the case of recorded data (i.e., file music, movie), it would require complex algorithms to maintain data consistency in the case of real-time data (i.e., news, emergency alerts, location-based advertisements). It is common that the popular data objects could become unpopular or useless as time goes by. How to deal with the redundant data items is still an open issue in this literature. In addition, there is usually difficulty in determining a suitable replica number for the data items that have different popularity. In fact, a large replica number often leads to poor system performance in terms of resource utilization.

The work that we presented in this paper is somewhat similar to previous work (Gopalakrishnan et al., 2004; Yamamoto et al., 2006; Roussopoulos and Baker, 2006; Pitoura et al., 2010). For example, we use the concept of replication to achieve load balancing. However, two important features distinguish our approach from the prior proposed approaches. First, by making good use of replicas created for reliability purpose (Wang et al., 2011), our approach is proposed to balance the load of the nodes in an environment with time-varying load distributions and frequent load fluctuations. This approach is an extension of our previous work (Wang et al., 2011, 2010) in which the replication problems such as the number of replicas and the location of replicas were studied and addressed with a location-aware replication scheme. Second, in this paper, a new cost model is developed for load balancing cost estimation. By combining with the performance tuning scheme, our approach empowers the nodes to explore different load balancing strategies and to choose the best strategy for load balancing and cost minimization.

3. Problem formulation and motivation

We study the load balancing problem in a general overlay network S which can be described by a set of nodes N and a d-dimensional coordinate space G ($d \geq 1$) where $|N| = n$. This coordinate space can be logical, such as a geographical area of interest. At any point in time, the entire coordinate space G is dynamically partitioned among n nodes such that each node owns one individual rectangular region R within the space G, satisfying $G = \cup_{i=1}^{n} R_i$. Node E_i manages the region R_i, for all $0 < i \leq n$.

We assume that the network nodes are not mobile. Compared to mobile devices, desktop computers usually have more access to network bandwidth, more stable connections, and more storage space. We let c_i denote the capacity of node E_i. In our work, we use it to refer to two types of node resources, storage space and bandwidth, because those factors are the two main factors in applications such as content searching (BitTorrent, 2011; eMule, 2010) and media streaming dissemination (JOOST, 2008; Livestation, 2010; PPTV, 2011). For node E_i, c_i is denoted as a 2-d row vector in the form of $[c_i^s, c_i^d * b_0]$, where c_i^s is the maximum number of files that can be stored on node E_i, and c_i^d is the maximum number of nodes that can be connected with E_i simultaneously. For simplicity, we assume that each node E_j that has connected to E_i would consume the same bandwidth b_0 for data propagation. Let *utilization*$u_i(t)$ denote the fraction of its capacities that have been used at time t: $u_i(t) = lo_i(t)/c_i(t) = [lo_i^s(t)/c_i^s, lo_i^d(t)/c_i^d]$, where $lo_i(t)$ denotes the workload of node E_i at time t, which consists of two elements $lo_i^s(t)$ and $lo_i^d(t)$. The variable $lo_i^s(t)$ refers to the number of files stored on node E_i and $lo_i^d(t)$ refers to the number of connections linked to node E_i at time t.

We consider S to be a system in which the workload and available capacities of the nodes could change as the nodes continuously access or leave the services provided by the other nodes in the presence of frequent load fluctuations and time-varying load distributions. In such a dynamic network, the problem of load balancing is quite difficult to solve and, to the best of our knowledge, has not been addressed before. Formally, we define the load balancing problem as follows:

*Problem Statement*1: *Given n nodes with* $u_i(t)$, $i \in [1,n]$ *at time t, devise a load balancing algorithm that minimizes* $\sum_{i=1}^{n} |u_i(t) - \sum_{i=1}^{n} u_i(t)|$ *while minimizing the load balancing cost given the node capacity limitations.*

Solving this problem, however, is challenging, as it requires careful handling of the time-varying load among the nodes in a dynamic network that have unpredictable load fluctuations, and practical systems often contain a large number of nodes with heterogeneous capacities. Simple solutions such as applying modified space reassignment approaches either fail to minimize the load balancing cost or ignore the impact of load fluctuations on the system performance. We next present a motivating example to show the reason as well as some insights into the solution.

3.1. Relevance of load balancing

In this subsection, three basic questions are first proposed to provide a better understanding of the problem of load balancing discussed in our work.

What is the load balancing cost? As mentioned in many studies (Karger and Ruhl, 2006; Tewari and Kleinrock, 2006; Shen and Xu, 2008), it is desirable to allow the nodes in the system to have an amount of the load that is in proportion to their capacities. To achieve this goal, a fraction of the workload at the nodes that have a heavy load could need to be moved to the other nodes with light loads. In this procedure, two types of costs are involved: *the load information collection cost* and *the load movement cost*. We denote the *load information collection cost* to be the communication cost of acquiring load information about the nodes that are currently in the system. The *load movement cost* refers to the communication cost of load redirection and migration.

Is the absolute load balancing achievable? This question has been answered by many studies (Godfrey and Stoica, 2005; Surana et al., 2006; Chen et al., 2008), in which the problem of load balancing in a dynamic network has been classified as an NP-hard problem. Furthermore, for a system, it is impractical to trigger the load balancing mechanism once an imbalance of the loads occurs. Therefore, a system parameter δ, named the imbalance tolerance (IT) is introduced, which we define as the maximum ratio between the largest and smallest utilization. For node E_i in a load balanced system, it holds that $\max_{k \neq j}\{u_j(t)/u_k(t)\} < \delta, \forall E_k, E_j \in B_i(t), t \in R$, where $B_i(t)$ is a set of the nodes for which load information is collected by node E_i at time t. If such a condition is violated which we refer to as *load imbalance*, then node E_i notifies E_k of its state and performs load balancing.

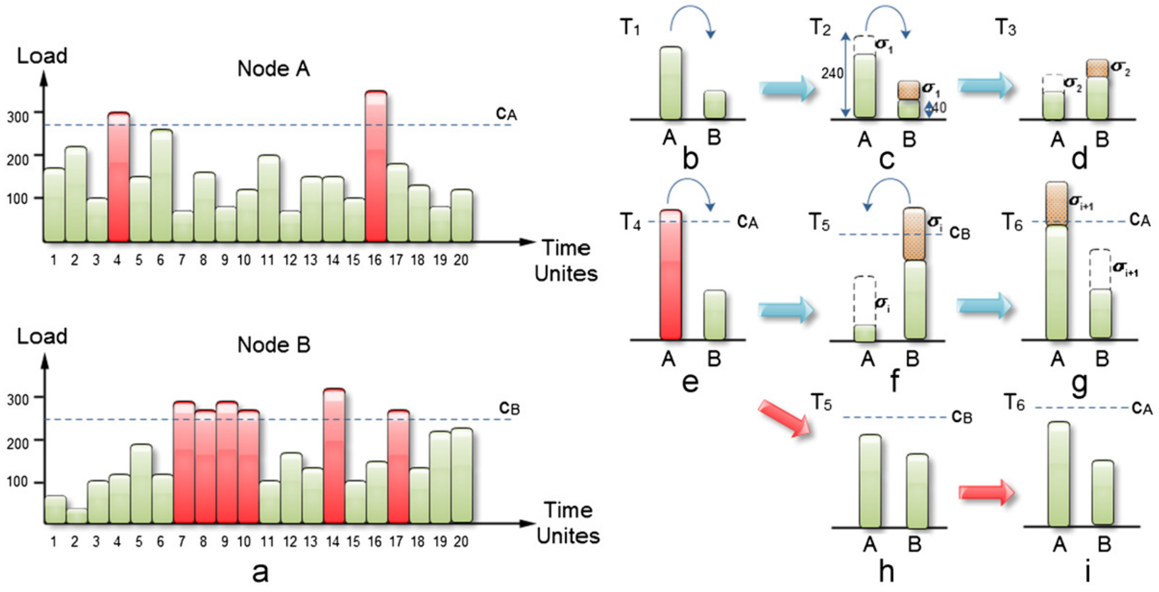

Fig. 1. A motivating example.

Is load balancing of space reassignment feasible? To answer this question, a motivating example is presented first.

3.2. System model

Figure 1 shows a snapshot of the load ($lo^d(t)$) of two nodes in a distributed system, where time is slotted into 5-s units. For simplicity, we set $\delta=1$, $dim(u_i(t))=1$, c_A=280 and c_B=240. In this example, nodes A and B are overloaded at time units 4, 16, and time units 7–10, 14, 17, respectively, as indicated by bars with the height exceeding c_A (c_B). Suppose that the procedure of load balancing is triggered when a load imbalance or an overload occurs, as shown in Fig. 1(b) and (e), respectively. To be specific, Fig. 1(b) depicts a scenario of nodes with different utilization. At time unit 2, node A moves a portion of its workload to node B according to the load information of node B at time unit 1. After the load migration, $lo_A^d(t2)$ changes to $lo_A^{d'}(t2)=240-\sigma_1$, where $\sigma_1=40$ is the amount of load moved from node A to node B. Correspondingly, $lo_B^{d'}(t2)=lo_B^d(t2)+\sigma_1$. Because $u_A(t2)>u_B(t2)$, the procedure of load balancing is triggered again, and then $lo_A^d(t3)$ and $lo_B^d(t3)$ update to $lo_A^{d'}(t3)$ and $lo_B^{d'}(t3)$ at time unit 3, respectively, as shown in Fig. 1(d). This procedure is repeatedly executed until the condition of $u_A(tn)=u_B(tn)$ is satisfied, $\forall lo_A^{d'}(tn)$, $lo_B^{d'}(tn)\leq$ min(c_A,c_B). Clearly, this method is not an efficient way for balancing the loads with setting of $\delta=1$ in terms of resource utilization. It could cause significant bandwidth and storage consumption at the nodes, which reduces the resource utilization and limits the scalability of the applications.

In fact, invoking the load balancing for every load imbalance is not necessary. One could propose to allow $\delta<1$ to reduce the cost of load migration and to gain immunity to momentary load imbalance. However, this modified scheme performs poorly when frequent load fluctuations shown in Fig. 1(e) occur. Figure 1(e) depicts a scenario for one node that is overloaded temporarily, which satisfies $\{u_A(t4)/u_B(t4)\}>\delta$. To reduce the heavy load that is imposed on node A, load balancing is triggered and the operation of load migration is conducted in a similar fashion, as shown in Fig. 1(f). After taking some load from node A, $lo_B^{d'}(t5)$ changes to $lo_B^d(t5)+\sigma_i$. However, given the heavy load of node B at time unit 5, load balancing is triggered again, as shown in Fig. 1(g), which could cause frequent load migration and inevitably reduce the load balancing efficiency.

One solution is to use the concept of *redirecting*, for which some new incoming workload of the nodes is first redirected to node B, which has few loads, and load migration is triggered when an overload happens. When receiving the redirection load, node B checks its latest state and determines how to handle the load. If node B has available capacity, it processes the load. Otherwise, a reject message with node B's latest state is generated and sent to node A. Node A then restarts the procedure of load balancing based on the information included in the reject message and node A's current load state. This procedure is terminated when $\{u_A(t4)/u_B(t4)\}<\delta$ is satisfied. Node B handles the load in a similar manner. Figure 1(h) and (i) depicts a load balancing state of nodes A and B that is achieved by using the load redirection.

To make the approach practical, we also must need to answer the questions such as how to trigger the procedure of load balancing and how to optimize the load balancing operations to eliminate the load imbalance and reduce the load balancing cost? In consideration of these questions, we developed a replica-aided load balancing scheme, and we present its details in the following section.

4. Replica-aided Load Balancing scheme

A replica-aided load balancing scheme is designed to adaptively balance the loads among nodes in the presence of load fluctuations. In this section, we first introduce load monitoring at the nodes. Then the introduction of a replica placement strategy and the details of the proposed load balancing scheme follow.

4.1. Load monitoring

Each node E_i continuously monitors its workload based on two parameters, $lo_i^s(t)$ and $lo_i^d(t)$. Node E_i is marked as a heavily loaded node if its current load holds the relationship $T_r<\max\{lo_i^s(t)/c_i^s,lo_i^d(t)/c_i^d\}\leq 1$, where T_r is a parameter defined as the *phase* threshold of *region reshaping*. In such a case, node E_i triggers *region reshaping* and migrates some of its loads to the other nodes with a lower load.

If $T_i^s\leq\max\{lo_i^s(t)/c_i^s,lo_i^d(t)/c_i^d\}\leq T_i^r$, then E_i is lightly loaded, where T_s is a parameter defined as the *phase* threshold of *load sharing*. E_i triggers *load sharing*, and starts to redirect a part of the new incoming loads to its replicas for load balancing. It is worth noting that both a lightly loaded node and a heavily loaded node can receive loads migrated from the other nodes with heavier load to achieve load balancing.

If $\max\{lo_i^s(t)/c_i^s, lo_i^d(t)/c_i^d\} < T_i^s$, then E_i is idle and can receive loads issued by (or migrated from) the other nodes. The load balancing is terminated when there is no node that can receive the loads or the loads at the nodes are balanced within a ratio δ, which is expressed as $\max_{i \neq j}\{u_j(t)/u_i(t)\} < \delta, \forall E_j \in peernodelist_i(t), t \in R$, where $peernodelist_i(t)$ is a list of the nodes for whose the information is collected by node E_i at time t. It consists of a number of log entries, each of which includes a node ID, an IP address and a transmission port, a region's coordinate, and a set of node properties (e.g., capacity and load status). A region denotes a rectangular partition area in the coordinate space G, which is managed by a node (as mentioned in Section 3). Similar to CAN, each region in G is assigned to a node in the system. Two nodes are considered to be immediate neighbors when their rectangular areas' intersection is a line segment. In general, there are two types of nodes in $peernodelist_i(t)$: immediate neighbor nodes and shortcut nodes. The term shortcut node refers to the node that is an old neighbor node for a given node. As nodes arrive or depart, neighbor nodes could become shortcut nodes when they are not adjacent to the given node. Instead of removing old neighbors from the list, those nodes are kept in $peernodelist_i(t)$ and are used to speed up the procedure of message delivery. Detailed examples of the immediate neighbors and shortcut nodes in the system are presented in Wang et al. (2010). Note that the value of δ is highly related to the specific requirements of the applications. As suggested by Ganesan et al. (2004), we simply set δ to 2 and trigger the load balancing when the load balance is violated, i.e., $\max_{i \neq j}\{u_j(t)/u_i(t)\} > \delta$.

4.2. Replica placement

In Wang et al. (2011), replicas are placed to improve the reliability of the services offered by the unreliable nodes with consideration of the failure patterns and replication cost. Each node p in the multicast sessions deploys r nodes with a light load in the network as replica nodes by performing a two-phase procedure: neighbor selection and shortcut selection. In the first phase, $(1-\alpha)r$ neighbor nodes with a light load are employed as replica nodes for the purpose of low cost, where α is the importance parameter that is used to leverage the benefits of neighbor replica nodes and shortcut replica nodes and r is the number of replicas per node. In our system, the value of α is dynamically determined based on the nodes' state. Given the theoretical analysis and experimental results obtained through simulations, we set r to 4 to maximize the service reliability while minimizing the overhead caused by replica creation and maintenance.

Since the node p might not have "enough" light load nodes in its *peernodelist*, the node p will extend the search range once this condition occurs. In such a case, the nodes located within two hops from node p are considered. The shortcut selection is performed in a similar manner as in the first phase. It starts with the shortcut nodes in the *peernodelist* of p. Then this procedure is continuously executed at the next level until there are no αr capable shortcut nodes.

In the replica placement, only the nodes in the *peernodelists* are considered. The benefits are two-fold. First, this strategy reduces the replication cost by employing the nodes that are located in the same vicinity. Second, it alleviates the influence of network partitions on the service quality by deploying the data copies on shortcut nodes.

4.3. Load balancing

The goal of the replica-aided load balancing scheme is to avoid overloading by intelligently balancing the loads among nodes while keeping the cost low. The most distinct feature of the scheme is its use of replicas to reduce the heavy load imposed on the nodes and to avoid unnecessary load movements caused by the frequent load fluctuations. In our scheme, a part of the nodes' workload is first redirected to their replica nodes when they are in multicast sessions and then load movements are triggered when there is no replica node with few loads. Conceptually, the replica-aided load balancing scheme entails two major phases: *load sharing* and *region reshaping*.

4.3.1. Load sharing

Load sharing is invoked when node E_i's load is large than T_i^s. In *load sharing*, node E_i selects a set of replicas with available capacities and shares a part of the workload with them. The complete algorithm is sketched in Algorithm 1: E_i first examines which type of workload is slightly heavy at time t; then it initializes a set R'_{E_i} consisting of the replicas with a light load, and it assigns a weight to each node in the set R'_{E_i}; then by taking into account the utilization of the nodes, the strategies of load redirecting β and γ are determined. In the following paragraphs, we present the details of Algorithm 1.

Algorithm 1. Load sharing (E_i, R_{E_i}, T_i^s)

```
**Input:** $R_{E_i}$: the set of replicas of node $E_i$, $T_i^s$: the phase threshold parameter
**Output:** $\beta, \gamma$: two vectors that correspond to the strategies of load redirections
1  $flag \leftarrow 0; mu \leftarrow 0; mi \leftarrow 0; c \leftarrow 0; g \leftarrow 0; U \leftarrow u_i(t)$;
       /*check the load's type*/
2  **if** $U(1,1) \geq T_i^s$ **then**
3      flag =1; mu = U(1,1)-$T_i^s$;
4  **end if**
5  **if** $U(1,2) \geq T_i^s$ **then**
6      flag =(1≪1); mi = U(1,2)-$T_i^s$;
7  **end if**
       /*initialize the replicas set */
8  **for** $mi, mu > 0$ **do**
9      **for** each node e $\in R_{E_i}$ **do**
10         $V \leftarrow u_e(t)$;
11         **if** $\max\{\frac{lo_e^s(t)}{c_p^s}, \frac{lo_e^d(t)}{c_p^d}\} < 1$ **then**
12             $g \leftarrow g + V(flag)$;
13             **if** flag =1 **then**
14                 $av[j] = c_p^s - lo_e^s; c \leftarrow c + av[j]$;
15             **else**
16                 $av[j] = c_p^d - lo_e^d; c \leftarrow c + av[j]$;
17             **end if**
18             add node e to the list $R'_{E_i}$;
19         **end if**
20     **end for**
       /*initialize the replicas' weight*/
21     add node $E_i$ to $R'_{E_i}$
22     **for** each node e $\in R'_{E_i}$ **do**
23 $V \leftarrow u_e(t); d[i] = 1 - V[flag]/g; \omega[i]$ = av[i]/c;
24     **end for**
       /*allocate the load to the nodes in $R'_{E_i}$ */
25     sort $\omega$ and $R'_{E_i}$ in descending order of d;
26     **for** each node e $\in R'_{E_i}$ **do**
27         **if** flag%2= 1 **then**
28             $\beta_i = mu * \omega[i]$;
29         **else**
30             $\gamma_i = mi * \omega[i]$;
31         **end if**
32     **end for**
33     flag =2;
34 **end for**
```

(i) Node E_i examines its current workload $lo_i^s(t)$ and $lo_i^d(t)$ (lines 2–7). If $U(1,1) \geq T_i^s$ (i.e., $lo_i^s(t) \geq c_i^s(t) * T_i^s$), then E_i is currently lightly loaded, and E_i starts to redirect some of the file transfer requests to its replica nodes for load sharing, which we refer to as *file redirection*. Similarly, if $U(1,2) \geq T_i^s$ (i.e., $lo_i^d(t) \geq c_i^d(t) * T_i^s$), then a part of the new incoming connection requests will be selected and forwarded to E_i's replica nodes, which we refer to as *connection redirection*. Both the file redirection and connection redirection are implemented when the conditions $U(1,1) \geq T_i^s$ and $U(1,2) \geq T_i^s$ fare true.

(ii) In each round, for each replica node e in R_{E_i}, E_i checks whether $\max\{lo_e^s(t)/c_p^s, lo_e^d(t)/c_p^d\} < 1$. If so, it inserts node e into a new set R'_{E_i} (lines 9–20). The intuition behind this step is to share the loads with the replica nodes that have available capacity. Let $av[j]$ denote the available capacity vector of the node in the set R'_{E_i}, $\exists j \in [1, |R_{E_i}|]$. We measure $av[j]$ with the following equation:

$$av[j] = \begin{cases} c_e^s - lo_e^s & \text{if } \frac{lo_e^s(t)}{c_e^s(t)} \geq T_i^s,\ flag = 1 \\ c_e^d - lo_e^d & \text{if } \frac{lo_e^d(t)}{c_e^d(t)} \geq T_i^s,\ flag = 2 \end{cases} \tag{1}$$

If both $lo_e^s(t)/c_e^s(t) \geq T_i^s$ and $lo_e^d(t)/c_e^d(t) \geq T_i^s$ are satisfied, then the file redirection is first performed and then it is followed by a connection redirection.

(iii) In each round, for each node e in R'_{E_i}, E_i computes $d[j]$ and $\omega[j]$ (lines 22–24), where $d[j]$ and $\omega[j]$ represent the priority and importance of the node e at $R'_{E_i}(j)$ compared with the other nodes in R'_{E_i} in terms of $V[flag]$) and $av[j]$, respectively, where $flag$ is the symbol bit of the node. It satisfies that $V[flag] = lo_e^s(t)$ if $flag=1$; $V[flag] = lo_e^d(t)$ if $flag=2$. Then, $d[j]$ is computed as $d[j] = 1 - V[flag]/g$. The larger $V[flag]$ is, the smaller the value of $d[j]$ is, and the less priority the node e has. In our scheme, the use of $d[j]$ is to reduce the probability of the nodes with a high utilization that are imposed on a heavy load by the load redirections.

Intuitively, a node with a higher capacity should carry or process more loads than the nodes with lower capacity. Here, $\omega[j] = av[j]/c$ is introduced to measure the capacity of nodes taking more workload compared to the other nodes in the same set. For a given c, $\omega[j]$ increases with the growth of $av[j]$, which means that a higher workload can be carried by e without causing node overload. It is important to note that $d[j]$ and $\omega[j]$ are two different metrics. Given the definitions of those metrics, we can easily find that there is no direct correlation between those metrics: each is employed to describe the capacity of the node in a certain respect. In practice, it is common that the nodes with low utilization do not have high available capacities to serve the other nodes in the system.

(iv) Based on the sorted list R'_{E_i}, a load assignment is performed and the strategies of load redirection β and γ are then determined (lines 26–32). Specifically, in each strategy, each node in R'_{E_i} is assigned a portion of the loads mu or mi with respect to its available capacities. With these strategies, E_i forwards the loads to the replica nodes via messages. Upon receiving those messages, the nodes check their loads, and take or process the loads based on the information included in the messages if they have available capacities. Otherwise, the redirection messages will be abandoned because of the heavy loads on the nodes. Note that a large number of the file transfer and connection requests could cause a high cost in communication. A timer κ is introduced for message aggregation and elimination. This timer starts when the node issues a request message. It terminates when the node receives a response message or a data message with request information sent from the node that receive the request. If none of these requests are received in 120 s, then the request message is reissued.

Algorithm 2. Region reshaping (E_i,T_i^r,**K**)

```
Input: $R_{E_i}$: the set of replicas of node $E_i$,$T_i^r$: the phase threshold parameter,
       $K$: the maximum number of iterations
Output: None;
1  $flag \leftarrow 0$; $mu \leftarrow 0$; $mi \leftarrow 0$; $U \leftarrow u_i(t)$;
       /*node selection*/
2  if $U(1,1) \geq T_i^r$ then
3    flag =1; mu = U(1,1)-$T_i^r$;
4  end if
5  if $U(1,2) \geq T_i^r$ then
6    flag =2+flag; mi = U(1,2)-$T_i^r$;
7  end if
8  while $k < K$ do
9    for each node e in $peernodelist_i^j t$ do
10     $V \leftarrow u_p(t)$;
11     if flag =1 then
12       if $lo_e^s < lo_{E_i}^s$ && $e \notin \overline{Q}$ then;
13         d[i] = the distance between e and $E_i$;
14         add e into list $Q'$;
15       end if
16     else
17       if $lo_e^d < lo_{E_i}^d$ && $e \notin \overline{Q}$ then;
18         d[i] = the distance between e and $E_i$;
19         add e into list $Q'$;
20       end if
21     end if
22   end for
23   while $Q' \neq \Phi$ do
24     sort list $Q'$ in ascending order of d;
25     RRP = the first node of the list $Q'$;
       /*reshaping the regions */
26  if $LBC_{RRP} > LBC_R$ && $lo_e^s + lo_e^s < T_i^r * LBC_{RRP}$ &&
        RRP is the sibling of node $E_i$ then
27     do region merging;
28     jump to line 42;
29   end if
30   if $LBC_{RRP} > LBC_R$ && $lo_e^s < T_i^r * LBC_{RRP}$ &&
        RRP is not the sibling of node $E_i$ then
31     do region swapping;
32     jump to line 42;
33   end if
34   if $LBC_{RRP} = LBC_R$ && $lo_e^s + lo_e^s < T_i^r * LBC_{RRP}$ then
35     do region splitting;
36     jump to line 42;
37   end if
38   remove RRP from $Q'$ and add it to $\overline{Q}$;
39     end while
40 end while
41 EXIT with failure
42 EXIT with success
```

After moving off a fraction of the workload, E_i continues to receive requests from the nodes in the network. It randomly chooses some requests to process and forwards the rest to the node in R'_{E_i} by applying Algorithm 1. Although the forwarded workload is not handled by node E_i, E_i creates a new record with workload information (e.g., size and load holder), and then adds it into its load list. This method allows that E_i to be able to easily hand over all of its loads to others when it is overloaded. This procedure continues until $u_i(t)$ is less than T_i^s or increases up to T_i^r and there is no light load replica node around E_i.

4.3.2. Region reshaping

Node E_i initiates *region reshaping* whenever $T_r < \max\{lo_i^s(t)/c_i^s, lo_i^d(t)/c_i^d\}$. As sketched in Algorithm 2, node E_i first performs *node searching* within a search range in such a way that a node is found with few loads for load migration (lines 2–22), which we refer to as a *region reshaping partner* (*RRP*). Such a search range is defined by node E_i to be a region that covers the nodes in the set $peernodelist_i^{j+1}(t) = \{neighbors\ of\ the\ node\ in\ peernodelist_i^j(t)\}$, $\forall j \geq 1$. Initially, j is set to 1 and node E_i looks through the set $peernodelist_i(t)$ to check whether a node E_m with a lower load exists. If so, then E_i stops searching and reshapes the nodes' region to balance the loads between the nodes. Since there might be more than one node that has a lower load in the set, the node with the shortest geographical distance to the node E_i will be selected as RRP to reduce the cost of load migration in transmission. If there is no node with a lower load in the set $peernodelist_i(t)$, E_i increases j by 1 so that its search region is extended. Then, the *node searching* is conducted in a similar manner. This procedure is executed repeatedly until either an RRP node is detected or j reaches its maximum $K = 2$, where K is a system constant that is configured by default (Wang et al., 2011). We do not increase the value of K in the scheme of load balancing in such a way that the overhead of the nodes' information maintenance in the system can be restricted within a certain level.

Reshaping region is performed to adjust the nodes' load assignment. In terms of the RRP capacity and location, three scenarios are considered in our scheme.

Region merging (lines 26–29): We merge two regions to form a new region when an RRP node with a higher capacity is a sibling node of E_i, as illustrated in Fig. 2(a). In our work, two nodes are considered to be sibling nodes when they are neighbor nodes and have the regions of the same size (Ratnasamy et al., 2001; Wang et al., 2010). The newly formed region is assigned to the node that has a higher capacity. Then, the other node works as a backup node recorded in the owner node of the new region, where the owner node denotes the node which manages the new region and is responsible of collecting information about the region and replying the related requests issued by other nodes in the system. This scenario provides a new opportunity to the end system recovery such that the backup node could be used temporarily to take over the failed node's region when a node failure occurs. For brevity, we do not discuss this scenario in detail in this paper.

Region splitting (lines 34–37): This scenario is for an RRP node that has a capacity that is comparable to E_i. Before migrating into E_i's region, the RRP node first merges its load into its neighbors. Then, E_i splits its region in half and assigns it to the RRP. Consequently, the related load of that part is handed over to the RRP node. An example of such a scenario is given in Fig. 2(b).

Region swapping (lines 30–33): This scenario is performed when an RRP node with a higher capacity owns a smaller region. In such a case, E_i swaps its corresponding region with that of the RRP. This action allows the node RRP with more capacity to handle more workload than E_i (see Fig. 2(c)).

For consistency, related update messages are sent out to notify the changes after region reshaping. However, in some situations, the selected RRP node may not have sufficient capacity to carry (serve) the workload migrated from node E_i. If this scenario occurs, the Algorithm 2 is re-triggered at node E_i. This procedure is repeatedly executed until either an RRP node with sufficient capacity is detected or Algorithm 2 exits with failure.

Consider the example mentioned in Section 1. In the multimedia streaming application, the end users can subscribe to any published stream in the network by sending subscription requests. Such requests are continuously forward by the end users/nodes in the network towards the end users/nodes that have the required streams until the other subscriber nodes with similar interests are reached. The subscriber nodes refer to the nodes that have already subscribed to the streaming services of interest. They are responsible of receiving the subscription requests from the end users/nodes and disseminating stream content to them when new stream contents arrive. At any point in time, a node can unsubscribe to a multimedia streaming service by sending a un-subscription request to the upstream node along the service path.

Given the streams with a higher popularity are often demanded by a large number of end-users. If only a few nodes with a limited capacity are employed to serve the voluminous demand of the end-users, they might turn to bottlenecks and eventually degrade the system performance. To cope with this situation, in RALB, a number of replica nodes that are placed for reliability purpose as mentioned earlier are utilized to balance loads between the nodes. Once a lightly loaded node is detected, the node starts to redirect some of the multimedia streaming subscription requests to its replica nodes for load sharing. Upon receiving those redirected requests, the replica nodes check their loads, and take or process the loads based on the information included in the messages if they have available capacities. After moving off a fraction of the workload, the node continues to receive requests from the other subscriber nodes in the network. Once the node is heavily loaded or overload, the operations of region reshaping are performed to adjust the load assignment between nodes. This procedure is repeatedly executed until either the load balancing is achieved or it exits with a failure status if there are no RRP nodes that are located in the search area.

Compared to the existing schemes, our load balancing scheme has the following two advantages. First, the *node searching* mechanism enables our approach to distribute the workload across the nodes in a wide range, which gains immunity to the influence of the skewed load distribution. This scenario occurs essentially because the shortcut nodes maintained by the nodes are often far away from them. Second, our scheme takes into account the impact of load fluctuation on the system performance. Our replica-aided load balancing scheme empowers nodes to adaptively switch their load balancing strategies to cope with sudden and random load changes caused by the load fluctuations.

5. Performance tuning scheme

In this section, we study the problem of tuning local parameters for optimum resource utilization. We first introduce our cost model for estimating the load balancing costs of RALB.

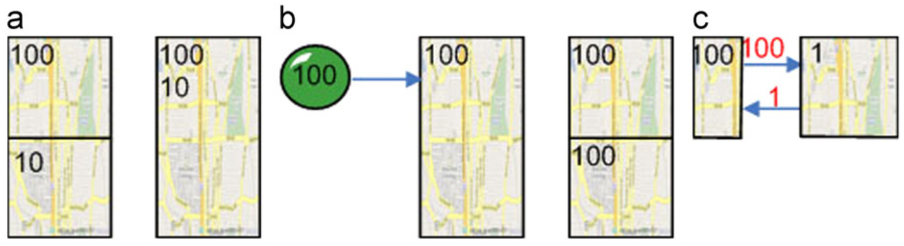

Fig. 2. Region reshaping mechanism. (a) Region merging. (b) Region spliting. (c) Region swapping.

5.1. Cost model

Load balancing in the RALB algorithm consists of load sharing and region reshaping. We use lbc_{ij} and lo_{ij}^{R} to denote the communication cost of redirecting a subscription (un-subscription) request from node i to node j and the amount of workload redirected from node i to node j, where the subscription (un-subscription) requests are the messages issued to subscribe to (unsubscribe from) the data delivery services provided by the nodes. Given the definition mentioned in Section 3, lo_{ij}^{R} can be computed as $lo_{ij}^{R} = lo_{ij}^{s} + lo_{ij}^{d}$, where lo_{ij}^{s} and lo_{ij}^{d} represent the number of files and the link connections moved from node i to node j, respectively. Given that the subscription requests are often of fixed size and the link latency between nodes is of great importance, we set lbc_{ij} to l_{ij}, where l_{ij} refers to the latency required for a message to transmit from node i to node j on average.

Estimating the cost of region reshaping LBC_R is relatively complicated as it depends on the specific state of the nodes' workload. Therefore, the three scenarios mentioned in Section 4 are further studied below.

Region merging: Let node p be a node that triggers its load balancing procedure and finds an RRP node q with the ability to take extra loads from the other nodes in the system. In the scenario of region merging, an RRP node with a higher capacity is detected and employed as an owner node whose responsible region is merged from two regions with the same size. For load balancing, the RRP node absorbs all of the workload of node p and p's replicas and handles them correspondingly as the subscription (or un-subscription) requests arrive. Thus, we can compute the load balancing cost of region merging LBC_{Rme} as

$$\begin{aligned} LBC_{Rme} &= \frac{lbc_{pq} * (lo_p + \sum_{i=1}^{r} lo_{pi}^{R})}{ML} \\ &= \frac{lbc_{pq} * (lo_p^d + lo_p^e + \sum_{i=1}^{r} (lo_{pi}^d + lo_{pi}^e))}{ML} \end{aligned} \tag{2}$$

where lo_p and lo_{pi}^{R} are the amount of the workload on node p and the workload redirected from node p to node i, respectively. ML is the number of workload information units contained in a transition message. The heavier the load that node p and its replicas have, the higher the load balancing cost is.

Region splitting: Once an RRP node q with a similar capacity to node p is detected, region splitting is invoked. In this scenario, the RRP node q will shed its load to one of its neighbors and gain half of node p's workload for load balancing. Thus, we have

$$LBC_{Rsp} = \frac{lbc_{pq} * (lo_p + \sum_{i=1}^{r} lo_{pi}^{R})}{2ML} + \frac{LBC_{qk} * (lo_p + \sum_{j=1}^{r} lo_{qj}^{R})}{ML} \tag{3}$$

where $LBC_{qk} * (lo_p + \sum_{j=1}^{r} lo_{qj}^{R})/ML$ is the cost of load migration from node q to node k, which is a neighbor of node q. Given the fact that the RRP node q has more available capacity than node p, we obtain

$$\frac{LBC_{qk} * (lo_p + \sum_{j=1}^{r} lo_{qj}^{R})}{ML} \le \frac{lbc_{pq} * (lo_p + \sum_{i=1}^{r} lo_{pi}^{R})}{ML} \tag{4}$$

Thus, we have

$$LBC_{Rsp} \le \frac{3}{2} * LBC_{Rme} \tag{5}$$

Region swapping: Region swapping is triggered when an RRP node with a higher capacity but a smaller region is found. In this scenario, nodes q and p will switch their regions and workloads. Thus, the cost of region swapping can be calculated as

$$LBC_{Rsw} = \frac{LBC_{pq} * (lo_p + \sum_{i=1}^{r} lo_{pi}^{R})}{ML} + \frac{LBC_{pq} * (lo_p + \sum_{j=1}^{r} lo_{qj}^{R})}{ML} \tag{6}$$

Given the observation that it is likely that a node with a bigger region has a heavier load than the node with a smaller region in the system, we have

$$LBC_{Rsw} \ge 2 * LBC_{Rme} \tag{7}$$

Based on the above analysis, the cost of load balancing LBC can be computed as follows:

$$LBC = P_p^{sh}(1 - P_p^{lo}) * \left(\sum_{i=1}^{r} LBC_{pi} * lo_{pi} \right) + (1 - P_p^{sh}) P_p^{lo} LBC_R \tag{8}$$

where P_p^{sh} is the probability of triggering load sharing on node p. $P_p^{sh} = 1 \Leftrightarrow (1 - P_p^{lo}) * (\sum_{i=1}^{r} LBC_{pi} * lo_{pi}) < P_p^{lo} C_R$; otherwise $P_p^{sh} = 0$. P_p^{lo} is the probability of $lo_p > T_p^r$ and no region reshaping occurs at t_i. Note that two independent workload factors lo_p^d and lo_p^s are contained in the vector lo_p. We have

$$\begin{aligned} P_p^{lo} &= P(lo_p(t_{i+1}) > T_p^r \,|\, lo_p(t) = lo_p(t_i)) \\ &= 1 - P(lo_p^d(t_{i+1}) \le T_p^r \,|\, lo_p^d(t) \\ &= lo_p^d(t_i)) * P(lo_p^e(t_{i+1}) \le T_p^r \,|\, lo_p^e(t) = lo_p^e(t_i)) \end{aligned} \tag{9}$$

To compute $P(lo_p^d(t_{i+1}) \le T_p^r \,|\, lo_p^d(t) = lo_p^d(t_i))$ and $P(lo_p^e(t_{i+1}) \le T_p^r \,|\, lo_p^e(t) = lo_p^e(t_i))$, we choose discrete-time stochastic processes for modeling the dependent workload values, since it is simple and has been proven to be applicable to various real-world stream data. Under this model, the values of future $lo_p(t_i + 1)$ and past $lo_p(t_i)$ are independent, given the sequence of workload values. Formally,

$$\begin{aligned} P(lo_p^d(t_{i+1}) \le T_p^r \,|\, lo_p^d(t) = lo_p^d(t_i)) &= P(lo_p^d(t_{i+1}) \le T_p^r) \\ &= 1 - \frac{\sum_{i=0}^{t_i} a_j lo_p^d(j)}{\sum_{j=0}^{t_i} lo_p^d(j)} \end{aligned} \tag{10}$$

where a_j is an influence coefficient. If $lo_p^d(j) \ge T_p^r$, $a_j = 1$. Otherwise, $a_j = 0$. Similarly, $P\{lo_p^e(t_{i+1}) \le T_p^r \,|\, lo_p^e(t) = lo_p^e(t_i)\}$ can be computed as

$$P(lo_p^e(t_{i+1}) \le T_p^r \,|\, lo_p^e(t) = lo_p^e(t_i)) = P(lo_p^e(t_{i+1}) \le T_p^r) = 1 - \frac{\sum_{j=0}^{t_i} b_j lo_p^e(j)}{\sum_{j=0}^{ti} lo_p^e(j)} \tag{11}$$

Based on the above equations, we have

$$LBC \in [LBC_{min}, LBC_{max}] \tag{12}$$

where

$$LBC_{min} = P_p^{sh}(1 - P_p^{lo}) * \left(\sum_{i=1}^{r} lbc_{pi} * lo_{pi} \right) + (1 - P_p^{sh}) P_p^{lo} LBC_{Rme}$$

$$LBC_{max} = P_p^{sh}(1 - P_p^{lo}) * \left(\sum_{i=1}^{r} lbc_{pi} * lo_{pi} \right) + (1 - P_p^{sh}) P_p^{lo} LBC_{Rsw}$$

Then we can define our parameter tuning problem as follows:

Problem statement 2: *Given the workload of node p and p's r replicas, determine the values of T_p^r so that the maximum load balancing cost LBC_{max} is minimized.*

5.2. Performance tuning scheme

To determine the best values for T_p^r, we adopt a block search scheme that divides a range of values into range partitions, and we sequentially look for values until the value that minimizes LBC_{max} is found in each range partition. To be specific, there are ρ range partitions, with partition boundaries at $R_0 < R_1 < R_2 < \cdots < R_i < \cdots < R_\rho$, where $R_0 = 0, R_{i+1} = R_i + (1/\rho)$ for $0 \le i \le \rho - 1$, $i \in R$. In each range partition $(R_i, R_{i+1}]$, this scheme starts with the value of $R_i + (1/\rho^2)$ and calculates its LBC_{max} value with the above equation. Then it increases R_i by $1/\rho^2$ and repeats this process. Such process is executed until R_{i+1} is reached. From making a comparison between the values of LBC_{max}, the one with minimum value of LBC_{max} is selected in the range of $(R_i, R_{i+1}]$. Finally, the best T_p^r is found by comparing the values of LBC_{max} of the range partitions.

The computational complexity of this block search scheme is $O(\rho^2)$. The larger the ρ is, the more comparisons the search scheme needs to conduct, the higher accuracy it has, and the better performance it could achieve. To reduce the computational complexity, a small value of ρ is preferred in the block search scheme, which could come at the cost of system performance. However, from the experimental results, we observe that there is no apparent system performance improvement in node utilization after ρ reaches 10. Given that, in this work, it is simply set to 10. We plan to optimize this block search scheme in our next study.

Thus far, a parameter tuning scheme has been introduced. It runs at the nodes and determines the best parameter settings. However, the question arises as to how to set parameter T_p^s to optimize the performance of the proposed approach in terms of resource utilization and cost. Recall that the parameters T_p^s and T_p^r are used to minimize the cost caused by load balancing. Rather than deterministically trigger massive load migration whenever a load imbalance or a node overload occurs, the use of T_p^s and T_p^r allows the nodes to handle the loads in a gradual fashion such that with a growth of a load on the nodes, the replicate nodes are first used to relieve the high burden of the nodes, and the load migrations are triggered only when the nodes are heavily loaded. In such a way, the number of load migrations caused by the frequent load fluctuations and slightly load imbalances is greatly reduced, which leads to a better system performance in terms of the resource utilization and cost.

In contrast to the parameter T_p^r, the setting of parameter T_p^s mainly focuses on how well the loads of the nodes are balanced. For each node in the system, it requires $\max_{p \neq j}\{u_j(t)/u_p(t)\} < \delta, \forall E_j \in peernodelist_p(t), t \in R$. Thus, we have $T_p^s = \max\{u_j(t)\}/\delta$. Periodically, the nodes in $peernodelist_p(t)$ exchange heartbeat messages and update the values of utilization. Then, node p computes T_p^s and invokes the procedure of load balancing when the current workload exceeds T_p^s.

6. Performance evaluation

In this section, we perform simulations to study the efficiency of the schemes proposed in this paper. First, we describe the simulation environment in our experiments, and we then present the simulation results.

6.1. Experimental environment

We use the GT-ITM topology generator to generate 10 internet-like physical networks, and we then build GeoCast on top of these physical networks. Each network consists of 8080 routers, and a number of end system nodes with heterogeneous capabilities. Similar to Zhang et al. (2007), it enables 5% of the nodes to have 1000 units of capacity, 15% of the nodes to have 100 units of capacity, 30% of the nodes to have 10 units of capacity, and the remainder of the nodes to have 1 unit of capacity. The higher the capacity the node has, the more powerful it is. Each unit of resource capacity allows a nodes to maintain 10 files in its local memory, and keep 1 connection with another node in the network.

In each simulation, the multicast groups are built first. Both the publisher nodes and the subscriber nodes sequentially participate in the groups, following an independent and identical uniform distribution. For simplicity, we do not consider the case when the subscribers in the same group have different interests in the publisher's content since it would be handled in a similar manner.

Note that there are no obvious sources of real-world trace data to drive the load fluctuations; thus, we adopt a random distribution to simulate it. At intervals, 10 existing nodes in the system are randomly chosen as the center of hot spots. Each hot spot is a circular area with a random initial radius between 1 hop and 3 hops, and it has the highest workload that are subscription requests issued by the nodes in the network. Once a request is accepted, node that the request was submitted to will continuously receive multicast information from the publisher node. The workload of the nodes within a circular area is determined by a formula $1-d/r$, where d is the distance of a node to the center of the hot spot in terms of the hop count and r is the radius of the hot spot. The nodes on its border have a workload of 0.

In the following section, we compare the performance of our load balancing scheme to two types of schemes: replication-based load balancing (RBLB) (Yamamoto et al., 2006; Roussopoulos and Baker, 2006; Gopalakrishnan et al., 2004) and ID space reassignment load balancing (IDSRLB) (Zhang et al., 2007; Karger and Ruhl, 2006; Ratnasamy et al., 2001). In GeoCast, RBLB is employed to balance the load by sharing the workload with replica nodes. If an overload occurs, then RBLB redirects the new incoming workload to the replicas that have with available capacities. IDSRLB achieves load balancing through ID space reassignment (i.e., massive load migration). Although it is a good solution in the case of the nodes with a heavy load, it decreases resource utilization and service quality because of the high cost caused by load migration.

6.2. Results

In the following experiments, we vary the system size from 1000 to 16,000 nodes, and we generate a service subscribing sequence to examine how well the load balancing schemes deal with the time-varying load distribution. Such a sequence consists of service subscribing requests issued by nodes in GeoCast at runtime, each of which sends 100 requests to the multicast sessions by following a uniform distribution on [0, 60 T] and continuously receives multicast information from the sessions.

6.2.1. Impact of the system size and multicast session

Figure 3 presents node utilization (NU) as a function of the system size, where

$$NU = \frac{1}{N}\sum_{i=1}^{N} \frac{lo_i^s(t)}{c_i^s} + \frac{lo_i^d(t)}{c_i^d}$$

refers to the average utilization of the nodes in GeoCast. The smaller NU is, the more lightly the node is loaded. Given that very similar results have been observed under various settings, we simply set the group number and group size to 10 and 10 (10,10). The results in Fig. 3(a) show that without load balancing, the nodes in GeoCast have heavier loads than the nodes with load balancing schemes, which indicates the necessity of load balancing. We also note that, compared to RBLB, the schemes of IDSRLB and RALB achieve better performance in terms of node utilization. This scenario shows benefits from the technique of load migration, where parts of the load at heavily loaded nodes are moved to the nodes with light loads.

Figure 3(b) depicts the standard deviation of node utilization in GeoCast with different schemes. Similar to Fig. 3(a), RBLB performs more poorly than other schemes because it only employs replicas with available capacities as load-receiving nodes, and it redirects the workload to them. As mentioned

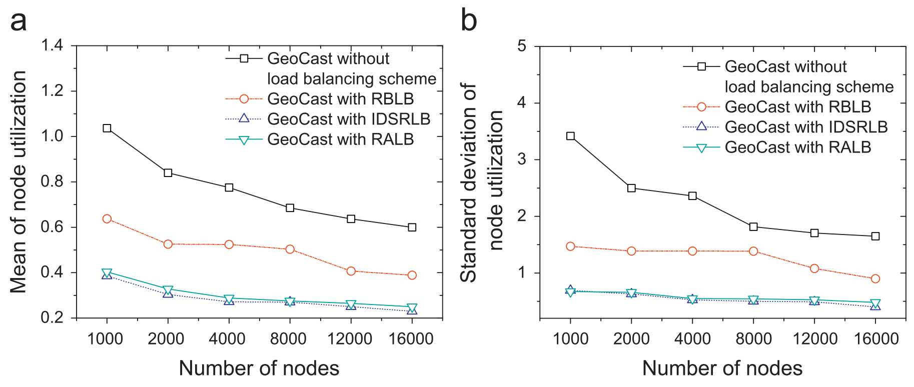

Fig. 3. Node utilization versus network size for group number=10 and group size=10: (a) mean and (b) standard deviation.

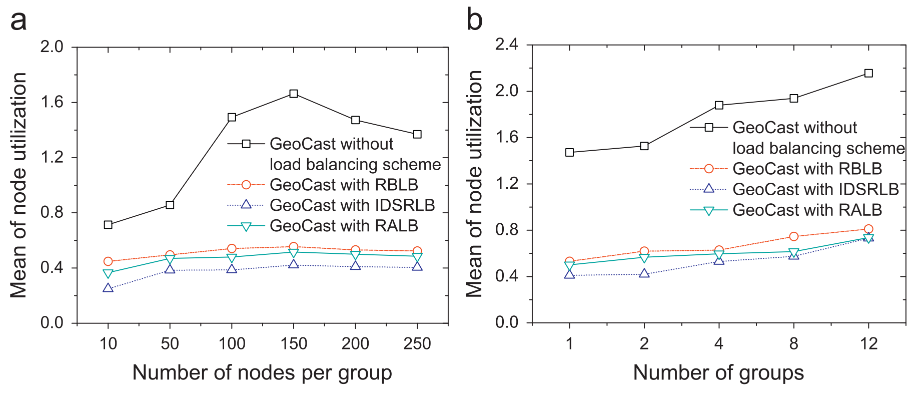

Fig. 4. Node utilization versus group number and size for network size=16,000: (a) group number=10 and (b) group size=200.

earlier, this scheme has difficulty in dealing with the workload when the nodes are heavily loaded. The majority of loads at nodes could be discarded for relieving hot spots and could prevent services from single point failures. To avoid that outcome, it is straightforward to increase the replica number r so that the performance of RBLB can be improved. However, the disadvantage of this enhancement is the increased creation and maintenance cost required by replica nodes, which limits the scalability of the system. We will discuss these factors in detail below.

In Fig. 4, we display how the group size and group number affect the node utilization in a network of 16,000 nodes. From the results depicted in Fig. 4(a) and (b), we clearly notice that the nodes in the system can greatly benefit from the operations of load balancing. All of the load balancing schemes could ensure that the utilization of the nodes is within a certain level. This is because all of the schemes will migrate part of a nodes' workload to lightly loaded nodes if a nodes becomes heavily loaded. In contrast, the nodes in GeoCast could have a high load even if there are only a few subscribers in a group because the number of nodes in the system that are passively involved into the multicast sessions is large. In such cases, the majority of node resources are consumed by the services such as message forwarding. This scenario reduces the efficiency of the nodes' resources. Further, in Fig. 4(a), it is interesting to note that there is a peak on the first curve when the group size is 150. This peak occurs basically because some of the member nodes in the multicast sessions that change their roles (in terms of serving as a publisher or a subscriber or a member or a combination of the three) as the group size becomes larger. In such a case, no additional message is issued as a notification of joining. On receiving the data from the root node of a multicast session, those nodes only need to deliver the data to the upper application in addition to handing down to the children nodes.

6.2.2. *Effect of load balancing*

In the following experiments, we provide further insights into how the load balancing schemes perform in a network of 16,000 nodes, and we investigate the influence of the parameter r on the system performance for different load balancing schemes. We set the group parameters to 10:10, and we use the technique of nonlinear regression to help us analyze the performance of the load balancing schemes RALB, RBLB and IDSRLB, which leads to a better understanding of the results obtained from the simulation.

Figure 5 depicts the *node utilization* and *request ratio* as a function of time for the load balancing schemes. We measured the *request ratio* as the number of subscribing requests received by the nodes in each time slot divided by the total number of requests in the service subscribing sequence. As shown in Fig. 5, in the first 5 min, approximately 6% of the requests have been issued by nodes for the service subscription. At the beginning, the nodes' workload is relatively high because of the uneven workload imposed by subscribing. As time elapses, the workload drops. This drop occurs because the operations of load balancing are performed, which release the hot spots in the network. Compared to RBLB and RALB, IDSRLB achieves its best performance in terms of node utilization. By reassigning the region distribution, it enables nodes with a high capacity to be likely to have larger responsible regions.

6.2.3. *Impact of load balancing overhead*

However, the scheme of IDSRLB is costly, as shown in Fig. 6(a) and (b). It is observed that, with IDSRLB, the nodes in

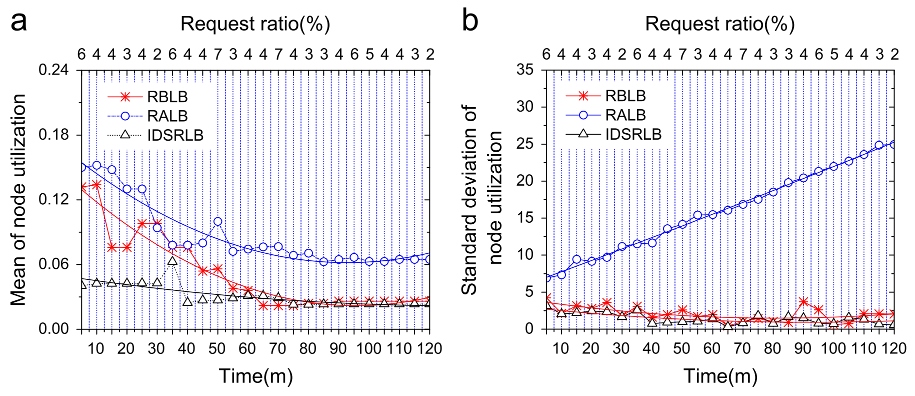

Fig. 5. Performance of the load balancing schemes over time in terms of node utilization: (a) mean and (b) standard deviation.

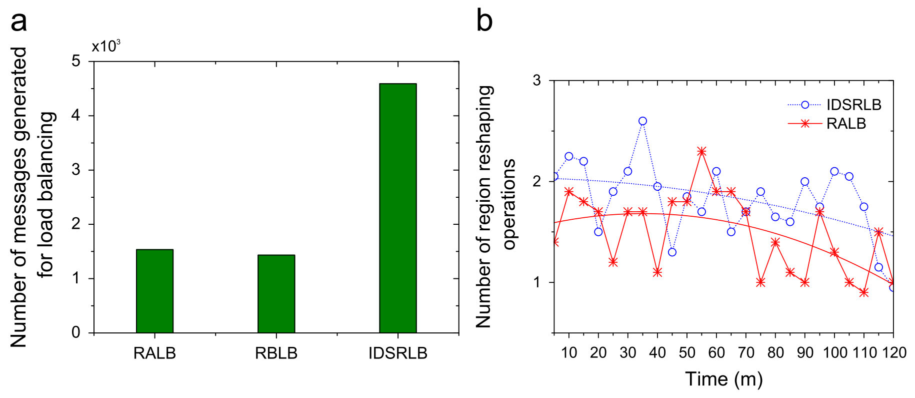

Fig. 6. Comparison of load balancing efficiency in terms of message volume and load reshaping operations: (a) message volume and (b) load reshaping operations.

GeoCast generate more messages than the nodes of the others schemes because of a larger volume of load migrations and region reshaping operations. In IDSRLB, those messages could impose heavy loads on the network links and some node in the network could become temporarily overload because of their low bandwidth. In such a scenario, a number of unnecessarily load movements and ID space reassignments could be caused, which reduces the efficiency of network resource usage and eventually degrades the performance of applications based on that. In contrast to IDSRLB, RALB handles the workload at nodes in a different way with consideration of frequent load fluctuations. As mentioned in Section 4, rather than migrating the loads to the nodes with a lower load, RALB makes full use of the replicas created by replication, and shares the workload with them, which reduces the amount of workload moved among the nodes, and consequently improves the load balancing efficiency. For this reason, RALB exhibits a performance that is comparable to IDSRLB at runtime, as shown in Fig. 5. We can see that the differences between IDSRLB and RALB become negligible as time goes by, which indicates the efficiency of RALB. For RBLB, it performs poorly in terms of node utilization. This observation can be explained by the fact that only replica nodes are taken into account to achieve load balancing. The performance of RBLB depends strongly on the parameter r.

6.2.4. *Impact of replication degree*

In Fig. 7, we show the impact of parameter r on the system performance in terms of two metrics: the message volume and the node utilization. We vary r from 2 to 16. As expected, as r increases, more the hot spots in the system are released, and the utilization of the nodes decreases. However, by comparing to RALB, we observe that the performance of RBLB can be slightly improved by increasing r. Even with $r = 16$, RBLB achieves almost same performance to RALB with $r = 4$ after the time reaches 65. In practice, it is unacceptable for a network to continuously increase r to improve the performance of the load balancing, since the maintenance cost is dramatically increased with r, as shown in Fig. 7(b). Given this scenario, we restrict r to 4, and we use RALB to balance the workload among the nodes. The benefits are two-fold. First, it avoids the additional cost caused for replica placement. Second, it makes the system have the ability to deal with the cases of nodes with a heavy load on a large scale.

Note that the number of messages generated for region reshaping could be far less than the overhead consumed for replica maintenance in the system with a low churn rate. In such a case, RALB might not be a good load balancing. However, the facts prove the contrary. In GeoCast, replica maintenance can be negligible because the majority of the state information of the replica nodes are piggybacked in the data or in updated messages communicated among nodes. With respect to the length of the data or update messages, the length of the information is quite small.

6.2.5. *Effect of the performance tuning scheme*

Figure 8 compares the effect of the proposed performance tuning scheme (TTS), the periodic threshold scheme (PTS) (Surana et al., 2006), and the deterministic threshold scheme (DTS) (Gopalakrishnan et al., 2004). The PTS and DTS are representatives of threshold selection schemes(Surana et al., 2006; Gopalakrishnan et al., 2004; Marques et al., 2009; Gao et al., 2011). In PTS, the load balancing threshold k_p is periodically computed based on the average utilization $\overline{U}$ of the nodes that report to a directory, setting $k_p = (1+\overline{U})/2$. In contrast to PTS, DTS deterministically sets the thresholds to fixed values, where $T_p^s = 0.3$ and $T_p^s = 0.75$.

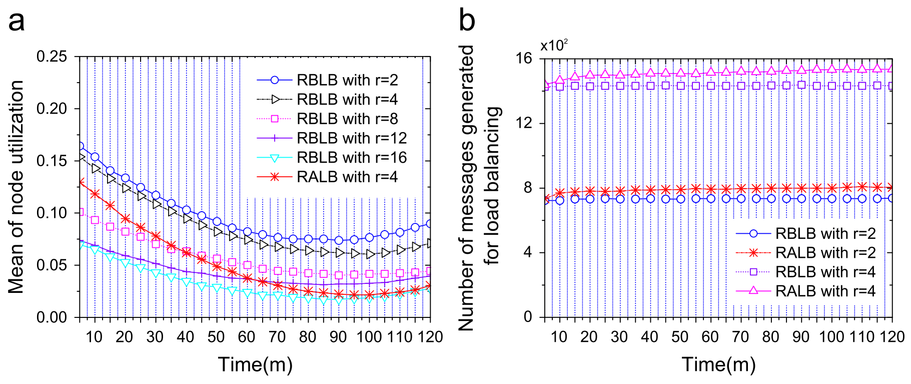

Fig. 7. Impact of parameter *r*: (a) node utilization and (b) message volume.

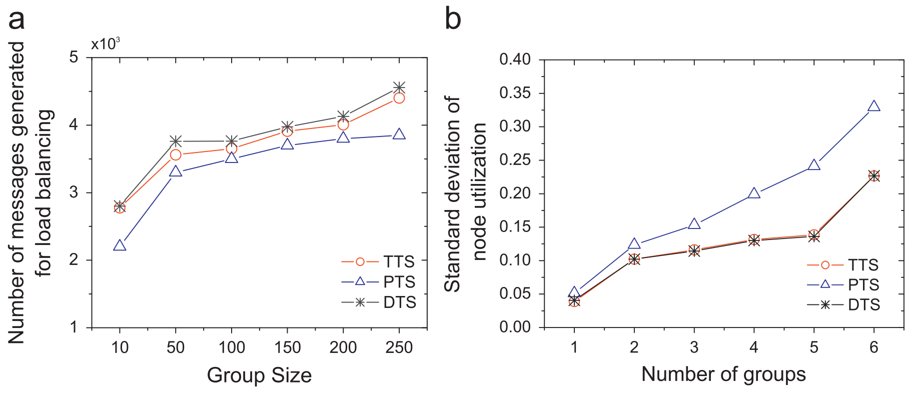

Fig. 8. Effect of performance tuning scheme: (a) node utilization and (b) message volume.

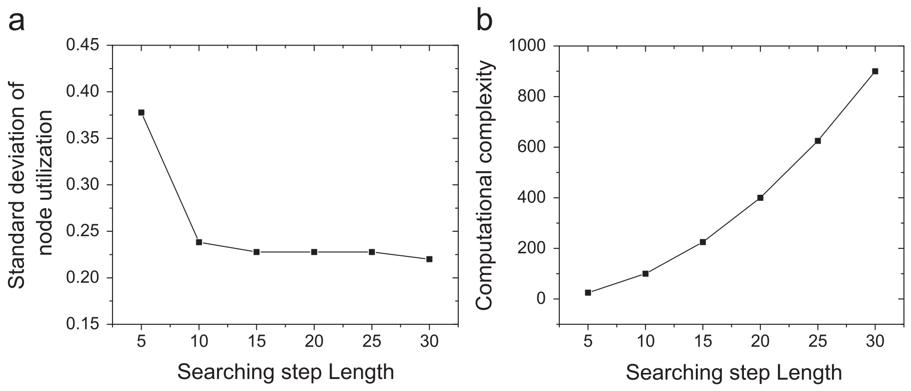

Fig. 9. Impact of searching step length: (a) node utilization and (b) computational complexity.

Assume that there are 4000 nodes and 10 groups in the system. We vary the group size from 10 to 250 nodes. In Fig. 8(a), we can see that, compared to TTS and DTS, PTS generates fewer messages for achieving load balancing in the presence of frequent load fluctuations and time-varying load distributions. However, this advantage comes at the expense of load imbalance, as shown in Fig. 8(b). In PTS, the load movements are triggered only when the nodes' utilization is larger than k_p. However, it suffers from the difficulty of dealing with load fluctuations. In contrast, DTS uses two threshold parameters to adapt the nodes' load to the changes of the load distributions. However, more messages are required to balance the loads among the nodes than in TTS, where the nodes dynamically adjust the threshold parameters to accommodate for the changes in load distribution. Thus, it is impractical to set the threshold parameters in a deterministic manner.

The results in Fig. 8(b) show that all of the schemes enable the nodes to adapt to changes in the system; however, in PTS, the load utilization becomes higher when the group size is larger. This is because when the group size is larger, the nodes in the system with TTS (DTS) will have a high probability of having replicas help with the short-time load fluctuations. The new incoming loads are first selectively redirected to those replicas with available capacities before the overload occurs.

Figure 9(a) illustrates the impact of the searching step length on the performance of load balancing based on the standard deviation of node utilization. The smaller the standard deviation, the smaller the differences among all of the nodes and the better the load balancing scheme. We set the group parameter to 10:10, and we vary the searching step length from 5 to 30. The results show that in a system of 4000 nodes, the standard deviation of node utilization decreases with the growth of the searching step length. However, after the value of the searching step length reaches 10, there is no pronounced decrease in the standard deviation of the node utilization. Potentially, this indicates that the performance tuning scheme does not rely strongly on the length of the searching step. Fig. 9(b) shows the computational complexity of the performance tuning scheme as a function of the searching step length. We can see that as the searching step length increases, the computational complexity dramatically increases. Given the results in Fig. 9(a), we find that it is practical to set the searching step length to a small value to eliminate the load imbalance among the nodes while keeping the cost low.

7. Conclusion

We have presented RALB, a replica-aided load balancing scheme for enabling the nodes of the overlay network to support communication applications for a large number of users with hardware devices in an environment of time-varying load distributions and unpredictable load fluctuations. To demonstrate the efficiency of RALB, we apply it to GeoCast and evaluate it through extensive experiments based on a realistic network topology model and a dynamic workload model. The results show that RALB enables an application to scale to a large number of nodes, a larger number of groups, and a larger group size. In the presence of dynamic load movements and load fluctuations, it can efficiently balance the load among nodes while minimizing the overhead required for large-scale load migration and node communication, when compared to the existing load balancing schemes.

Table 1
Definition of variables.

Notation	Definition
G	*d*-Dimensional coordinate space
$\{E_i\}$	The set of nodes in the G, $\forall i \in [0,n]$; n is the number of nodes in the G
R_i	The individual range managed by E_i, $\forall G = \cup_{i=1}^{n} R_i$
c_i	The capacity of E_i;a 2-d row vector
c_i^s	The storage capacity of E_i
c_i^d	The highest connection degree of E_i
b_0	The bandwidth unit
$u_i(t)$	The utilization of E_i
lo_i^s	The number of files stored on E_i
lo_i^d	The number of connections linked to E_i
δ	Imbalance tolerance parameter
$B_i(t)$	The set of nodes whose load information is collected by E_i
T_r	The threshold of load sharing phase
T_s	The threshold of region reshaping phase
R_{E_i}	The replicas set of E_i
r	The number of replicas per node
α	The importance parameter
$av[j]$	The available capacity vector of un-overloaded node j in R_{E_i}
$d[j]$	The priority of node j
$\omega[j]$	The importance of node j
$\omega[j]$	The importance of node j
flag	The symbol bit of node
$V[flag]$	The reference of node's utilization $u[flag]$
U	The reference of node's utilization
LBC	The load balancing cost
LBC_R	The cost of region reshaping
LBC_{Rme}	The cost of region merging
LBC_{Rsp}	The cost of region splitting
LBC_{Rsw}	The cost of region swapping
lbc_{ij}	The communication cost of redirecting a subscription request from node i to node j
lo_{ij}^R	The amount of workload redirected from node i to node j
l_{ij}	The latency required for a message to transmit from node i to node j
ML	The number of workload information units per message
p_p^{sh}	The probability of triggering load sharing on node p
p_p^{lo}	The probability of $lo_p(t_{i+1}) > T_p^r$ and no region reshaping occurs at t_i

As part of our future work, we plan to develop additional parameter tuning mechanisms and apply them to different types of applications and environments to evaluate their efficiency. These applications could include multimedia streaming applications, pervasive applications, voice-ip applications, and location-based advertising applications.

Acknowledgments

This paper is supported by 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 third author is partially supported by grants from NSF CISE NetSE program, Cyber Trust program, an IBM faculty award and an Intel ISTC on Cloud Computing.

Appendix A

The definition of the variables used in this paper is given in Table 1.

References

Avancha S, D'Souza P, Perich F, Joshi A, Yesha Y. P2P Mcommerce in pervasive environments. ACM SIGecom Exchanges 2002;3(4):1–9.
BitTorrent. Available at 〈http://www.bittorrent.com/〉; 2011.
Byers J, Considine J, Mitzenmacher M. Simple load balancing for distributed hash tables. Peer-to-Peer Systems II 2003;2735:80–7.
Cannataro M, Talia D. Towards the next-generation grid: a pervasive environment for knowledge-based computing. In: Proceedings of the international conference on information technology: computers and communications (ITCC'03); 2003. p. 437–441.
Chen J, Liao G, Hsie Jr. S, Liao C. A study of the contribution made by evolutionary learning on dynamic load-balancing problems in distributed computing systems. Expert Systems with Applications 2008;34(1):357–65.
ComScore. Number of U.S. computers accessing the internet via mobile broadband soars 154 percent in 2007. Available at 〈http://www.comscore.com/dut/Press_Events/Press_Releases/2008/03/Mobile_Broadband_Usage_Increases_in_US〉; 2010.
Delmastro F, Passarella A, Conti M. P2P multicast for pervasive ad hoc networks. Pervasive and Mobile Computing 2008;4(1):62–91.
eMule. Available at 〈http://www.emule-projec.tnet/〉; 2010.
Ganesan P, Bawa M, Garcia-Molina H. Online balancing of range-partitioned data with applications to peer-to-peer systems. In: Proceedings of the 30nd international conference on very large data bases (VLDB'04); 2004. p. 444–455.
Gao Q, Tang P, Deng T, Wo T. Virtualrank: a prediction based load balancing technique in virtual computing environment. In: IEEE world congress on services; 2011. p. 247–256.
Godfrey P, Stoica I. Heterogeneity and load balance in distributed hash tables. In: Proceedings of the 24th annual joint conference of the IEEE computer and communications societies (INFOCOM'05); 2005. p. 596–606.
GOGRID. (F5) load balancer. Available at 〈https://wiki.gogrid.com/wiki/index.php/(F5)_Load_Balancer〉; 2010.
Google. Googledocs. Available at 〈https://docs.google.com/〉; 2010.
Gopalakrishnan V, Silaghi B, Bhattacharjee B, Keleher P. Adaptive replication in peer-to-peer systems. In: Proceedings of the 24th IEEE international conference on distributed computing systems (ICDCS'04); 2004. p. 360–369.
Hu J, Gu J, Sun G, Zhao T. A scheduling strategy on load balancing of virtual machine resources in cloud computing environment. In: Proceedings of the 3rd international symposium on parallel architectures, algorithms and programming (PAAP'10); 2010. p. 89–96.
INTRICE. Available at 〈http://www.intrice.com/〉; 2009.
JOOST. Available at 〈http://www.joost.com/〉; 2008.
Karger D, Ruhl M. Simple efficient load-balancing algorithms for peer-to-peer systems. Theory of Computing Systems 2006;39(6):787–804.
Knoll M, Wacker A, Schiele G, Weis T. Decentralized Bootstrapping in Pervasive Applications. In: Proceedings of the 5th IEEE international conference on pervasive computing and communications workshops (PerCom Workshops'07); 2007. p. 589–592.
Kotoulas S, Oren E, Van Harmelen F. Mind the data skew: distributed inferencing by speeddating in elastic regions. In: Proceedings of the 19th international conference on World Wide Web (WWW'10); 2010. p. 531–540.

Kumar M, Shirazi B, Das S, Sung B, Levine D, Singhal M. PICO: a middleware framework for pervasive computing. IEEE Pervasive Computing 2003;2(3):72–9.

Kwon G, Ryu K. Bypass: topology-aware lookup overlay for DHT-based p2p file locating services. In: Proceedings of the 10th international conference on parallel and distributed systems (ICPADS'04); 2004. p. 297–304.

Livestation. Available at 〈http://www.livestation.com/〉; 2010.

Marques C, Ilarri S, Barroso G. Darc: a dynamic architecture for reconfiguration of web servers clusters using multiagent systems. In: Proceedings of the 5th international conference on networking and services (ICNS'09); 2009. p. 169–174.

Myles G, Friday A, Davies N. Preserving privacy in environments with location-based applications. IEEE Pervasive Computing 2003;2(1):56–64.

Pitoura T, Ntarmos N, Triantafillou P. Saturn: Range queries, load balancing and fault tolerance in DHT data systems. IEEE Transactions on Knowledge and Data Engineering 2010;PP(99), 1–14.

PPTV. Available at 〈http://www.pptv.com/〉; 2011.

Qiao Y, Bochmann G. A diffusive load balancing scheme for clustered peer-to-peer systems. In: Proceeding of 15th international conference on parallel and distributed systems (ICPADS'09); 2009. p. 842–847.

Ranjan R, Zhao L, Wu X, Liu A, Quiroz A, Parashar M. Peer-to-peer cloud provisioning: service discovery and load balancing. Cloud Computing: Principles, Systems and Applications 2010:195–217.

Rao A, Lakshminarayanan K, Surana S, Karp R, Stoica I. Load balancing in structured P2P systems. Peer-to-Peer Systems II 2003;2735:68–79.

Ratnasamy S, Francis P, Handley M, Karp R, Schenker S. A scalable content-addressable network. ACM SIGCOMM Computer Communication Review 2001;31(4):161–72.

Roussopoulos M, Baker M. Practical load balancing for content requests in peer-to-peer networks. Distributed Computing 2006;18(6):421–34.

Shen H, Xu C. Hash-based proximity clustering for efficient load balancing in heterogeneous DHT networks. Journal of Parallel and Distributed Computing 2008;68(5):686–702.

Steele T, Vishnumurthy V, Francis P. A parameter-free load balancing mechanism for P2P networks. In: Proceedings of the 7th international conference on peer-to-peer systems (IPTPS'08); 2008. p. 21–26.

Surana S, Godfrey B, Lakshminarayanan K, Karp R, Stoica I. Load balancing in dynamic structured peer-to-peer systems. Performance Evaluation 2006;63(3):217–40.

Techcrunch. Boingo wireless partners with gogo for in-flight internet access. Available at 〈http://techcrunchcom/2011/06/27/boingo-wireless-partners-with-gogo-for-in-flight-internet-access/〉; 2011.

Technologyreview. A new way for drivers to access the internet that, the automaker says, is safe. Available at 〈http://wwwtechnologyreviewcom/blog/editors/24634/〉; 2010.

Tewari S, Kleinrock L. Proportional replication in peer-to-peer networks. In: Proceedings of the 25th annual IEEE international conference on computer communications (INFOCOM'06); 2006. p. 1–12.

Wang Y, Liu L, Pu C, Zhang G. An Utility-driven routing scheme for scaling multicast applications. In: Proceedings of the 7th international conference on collaborative computing (CollaborateCom'10); 2010. p. 65–74.

Wang Y, Zhong Z, Liu L, Wu W. Efficient location-aware replication scheme for reliable group communication applications. In: Proceedings of the 10th international conference on networks (ICN'11); 2011. p. 394–400.

Warcraft. Available at 〈http://us.blizzard.com/en-us/〉; 2010.

Werstein P, Situ H, Huang Z. Load balancing in a cluster computer. In: Proceedings of the 7th international conference on parallel and distributed computing, applications and technologies (PDCAT'06); 2006. p. 569–577.

Xia Y, Chen S, Cho C, Korgaonkar V. Algorithms and performance of load-balancing with multiple hash functions in massive content distribution. Computer Networks 2009;53(1):110–25.

Yamamoto H, Maruta D, Oie Y. Replication methods for load balancing on distributed storages in P2P networks. IEICE Transactions on Information and Systems 2006;89(1):171–80.

Zhang J, Zhang G, Liu L. GeoGrid: A scalable location service network. In: Proceedings of the 27th international conference on distributed computing systems (ICDCS'07); 2007. p. 60–67.

Zhu Y, Hu Y. Efficient, proximity-aware load balancing for DHT-based P2P systems. IEEE Transactions on Parallel and Distributed Systems 2005; 16(4):349–61.