

# On the Effectiveness of Migration-based Load Balancing Strategies in DHT Systems

Di Wu, Ye Tian and Kam-Wing Ng  
Department of Computer Science & Engineering  
The Chinese University of Hong Kong  
Shatin, N.T., Hong Kong  
Email: {dwu, ytian, kwng}@cse.cuhk.edu.hk

**Abstract**—As a fundamental problem in DHT-based P2P systems, load balancing is important to avoid performance degradation and guarantee system fairness. In this paper, to get a better understanding about the effectiveness of migration-based load balancing approaches in DHT systems, we analytically study two representative migration-based load balancing strategies: *Rendezvous Directory Strategy (RDS)* and *Independent Searching Strategy (ISS)*. They differ in load information management and decision making in the process of load balancing. We analyze their performance in terms of efficiency, scalability and robustness, and explore the impact of their parameter settings. Based on the analysis results, we also propose a *Gossip-Based Strategy (GBS)* for load balancing in DHT systems, which attempts to achieve the benefits of both *RDS* and *ISS*. Later, the effectiveness of *GBS* is evaluated by simulation under different workload and churn.

## I. INTRODUCTION

The importance of load balancing in DHT-based Peer-to-Peer systems has motivated a number of proposals, e.g., [1], [2], [3], [4], [5], etc. Many of them are based on the concept of “virtual server”. Similar to a physical peer, a virtual server is also responsible for a contiguous range of the DHT identifier space, but one physical peer can host multiple virtual servers. Due to the limitation of pure namespace balancing, virtual servers are allowed to migrate between nodes. By shedding an appropriate number of virtual servers from heavy nodes to light nodes, load balancing can be achieved. Based on the difference in load information management and decision making of load balancing, the existing migration-based approaches can be categorized into two representative strategies: (1) *Rendezvous Directory Strategy (RDS)* and (2) *Independent Searching Strategy (ISS)*.

In *RDS*, the load information of each peer is periodically published to the rendezvous directory, which can be a central entity, or organized in a distributed fashion (e.g., tree structure in [5]). The rendezvous directory is responsible for scheduling the load reassignment to achieve load balance. However, the system churn may cause the problem of stale information, and hurt the performance of *RDS*. In *ISS*, the node doesn’t publish its load information anywhere else, and only provides its load information upon request. To achieve load balancing, a node should perform searching (or sampling) independently to find other nodes with inverse load characteristics, and then migrate the load from the heavy node to the light node. To

date, most proposals of the above two strategies are validated by simulation. There lacks theoretical analysis to compare their effectiveness in P2P environments, which are highly dynamic, in large scale and with malicious peers.

In this paper, we analytically study the effectiveness of *RDS* and *ISS* in terms of their efficiency, scalability and robustness. We find that, in spite of the stale information caused by system churn, *RDS* is still more efficient than *ISS* in most situations. *RDS* can lead to a more balanced system state, except that its convergence time will become longer when the churn rate is high. The main drawbacks of *RDS* are its scalability and vulnerability. Contrary to *RDS*, *ISS* has good scalability and robustness.

Based on the analysis results, we also propose a new load balancing strategy for DHT-based P2P systems, called *Gossip-Based Strategy (GBS)*, which is scalable and achieves both the efficiency of *RDS* and the robustness of *ISS*. In *GBS*, the whole system is formed into groups, and the gossip protocol is used for load information dissemination within the group. After load information dissemination, every group member has the full load information of its own group and the utilization information of the whole system. Thus, each group member can act as the rendezvous directory to perform load reassignment within the group, but the position of the rendezvous directory is randomized to make *GBS* resistant to the node-targeted attacks. Besides intra-group load balancing, inter-group load balancing and emergent load balancing are also allowed in *GBS* to achieve a more balanced system state. Finally, simulation-based study is performed to validate the effectiveness of *GBS*.

The remainder of this paper is structured as follows. We first introduce the related work in Sec. II. The modeling and analysis are presented in Sec. III. In Sec. IV, we propose and analyze a new approach of load balancing - *GBS*. Finally, Sec. V summarizes the whole paper.

## II. RELATED WORK

In DHT-based P2P systems, much research work about load balancing is based on namespace balancing. Namespace balancing is trying to balance the load across nodes by ensuring that each node is responsible for a balanced namespace (e.g, [6], [2], etc). It is valid only under the assumption of uniform workload distribution and uniform node capacity. For

a better balancing effect, “*Virtual Server*” was introduced in Chord. By letting each node host multiple virtual servers, the  $O(\log N)$  imbalance factor between nodes can be mitigated. If considering the node heterogeneity, virtual servers can be allocated proportional to node capacity (e.g., CFS [7]). In case that a node is overloaded, the node simply removes some of its virtual servers. However, such simple deletion will cause the problem of “*load thrashing*”, for the removed virtual servers may make other nodes overloaded. Simply by pure namespace balancing, it is hard to handle workload skewness well.

A more general approach is the migration-based approach, which is applicable to various kinds of scenarios and able to handle workload skewness. A number of migration-based approaches have been proposed to date (e.g., [3], [4], [5], etc). Most of them are based on the concept of virtual servers. In [3], Rao et al. proposed three simple load balancing schemes: “*one-to-one*”, “*one-to-many*” and “*many-to-many*”. Among them, “*one-to-many*” and “*many-to-many*” belong to the *RDS* category, while “*one-to-one*” belongs to the *ISS* category. To enable emergent load balancing, Godfrey et al. [4] make a combination of “*one-to-many*” and “*many-to-many*” and use them in different scenarios. The scheme proposed in [5] also belongs to the *RDS* category, but its rendezvous directory is organized as a distributed  $k$ -ary tree embedded in the DHT.

Although load migration is well studied in the field of traditional distributed systems, there is few analytical work to study its effectiveness in the P2P system, which is different from traditional distributed systems in its dynamics, large scale and node characteristics. That is the motivation of our work. And our proposed *Gossip-Based Strategy (GBS)* differs from the previous work in that, we take robustness into account besides efficiency and scalability.

### III. MODELING AND ANALYSIS

#### A. System Model

Our analysis is conducted under the steady state. To better measure the effect of load balancing algorithms, it is assumed that there is no workload shift and the load on peers will not change unless being altered by the load balancing process. To model the system dynamics (also referred to as “*system churn*”), each peer is associated with a lifetime  $L$  when first joining the system. A peer leaves the system when its lifetime is exhausted. The peer lifetime  $L$  is assumed to follow a certain distribution. Based on the measurement results in [8], we adopt the Pareto lifetime distribution in our analysis, in which the CDF is given by  $F(x) = 1 - (1 + \frac{x}{\beta})^{-\alpha}$ .

Based on the ratio between the node utilization and the system utilization, the nodes can be classified into three types, i.e., light node, heavy node and normal node. During load balancing, only light nodes and heavy nodes participate the load reassignment. For simplicity, the balancing between one light node and one heavy node is assumed to result in two normal nodes. This assumption can also be relaxed by introducing a probability for different balancing results.

The percentage of imbalanced nodes (including both light and heavy nodes)  $p_{IB}$  is adopted as the metric to indicate the

system imbalance degree. Let  $h(t)$  and  $l(t)$  be the number of heavy nodes and light nodes alive in the system at time  $t$ , then the percentage of the imbalanced nodes at time  $t$  can be represented by  $p_{IB}(t) = \frac{h(t)+l(t)}{N}$ , where  $N$  is the system size under the steady state.

A good load balancing algorithm should be able to minimize  $p_{IB}$  as much as possible, and as quickly as possible. It is important for avoiding the performance loss due to the load imbalance. With the elapse of time, there exists a time point  $T_g$ , when  $t > T_g$ ,  $p_{IB}(t)$  can not be further minimized any more. Here,  $T_g$  is called the “*Convergence Time*”.  $T_g$  together with the percentage of imbalanced nodes at that time  $p_{IB}(T_g)$  reflect the efficiency of load balancing algorithms.

In the following sections, besides efficiency, we also analyze the scalability and robustness of *RDS* and *ISS*.

#### B. Rendezvous Directory Strategy (RDS)

In *RDS*, every node updates its load status in a period of  $T_u$  to the rendezvous directory. The process of load balancing is executed periodically, and the period is denoted as  $T_{lb}$ . Let  $t_i$  be the time of the  $i$ -th load balancing action,  $t_{i+1} = t_i + T_{lb}$ , ( $i = 0, 1, \dots$ ). Note that  $T_{lb}$  and  $T_u$  are not required to be the same, but we often set  $T_{lb} \geq T_u$  for it is unnecessary to perform multiple times of load balancing actions within one update period.

A peer makes its first update when joining the system, and sends updates periodically within its lifetime. The joining time of peers may be different, but their update period is the same. If there is no system churn, every peer will make exactly one update within any duration of  $T_u$ .

In the basic *RDS*, the peer set for balancing includes all the nodes that make updates within one load balancing period  $T_{lb}$ . However, there exist some peers that may leave after making updates within the period  $T_{lb}$ . For this portion of peers, we call them “*faked*” live peers. As from the view of the rendezvous directory, they are still being taken as live peers and paired with other peers. This will result in the failure of some assigned load transfers and impact the effectiveness of *RDS*. For distinction, the peers that are really alive at the time of load balancing are called “*real*” live peers.

By applying the theory of stochastic process, we have the following result:

*Theorem 1:* In the basic *RDS*, given the update period  $T_u$  and the load balancing period  $T_{lb}$ , the probability that a peer is a “*real*” live peer in the peer set for balancing at the rendezvous directory is  $p_{real} = \frac{1}{\frac{\alpha-1}{\beta}T_{lb} + \frac{\beta}{(2-\alpha)T_u}[(1+\frac{T_{lb}}{\beta})^{2-\alpha}-1]}$  for Pareto lifetime distribution with  $\alpha > 2, \beta > 0$ .

*Proof:* Please refer to our technical report [9]. ■

During load balancing, only when the load movement happens between two “*real*” live peers can the movement be successful. Let  $p_{succ}$  be the probability that a pair (*heavy node, light node*) can be a successful pair, then we have  $p_{succ} = P(\text{heavy node is alive, light node is alive}) = p_{real}^2$ .

Let  $h'(t_{i+1})$  and  $l'(t_{i+1})$  be the number of heavy nodes and light nodes in the peer set for balancing at time  $t_{i+1}$  (including both *faked* and *real* live peers). Denote  $N_p(t)$  as

the number of effective pairs that can be successfully executed during load balancing at time  $t$ . Then for the  $(i + 1)$ -th round of load balancing at time  $t_{i+1}$ , we have:  $N_p(t_{i+1}) = \min\{h'(t_{i+1}), l'(t_{i+1})\} \times p_{succ} = \min\{\frac{h(t_i)}{p_{real}}, \frac{l(t_i)}{p_{real}}\} \times p_{real}^2 = \min\{h(t_i), l(t_i)\} \times p_{real}$ .

After a load balancing action, the changes of heavy nodes and light nodes are provided by  $h(t_{i+1}) = h(t_i) - N_p(t_{i+1})$  and  $l(t_{i+1}) = l(t_i) - N_p(t_{i+1})$ .

Given the initial number of heavy nodes and light nodes, we can compute the convergence time  $T_g$  and the percentage of imbalanced nodes at the convergence time  $p_{IB}(T_g)$  via the above iterative relation. In Sec. III-D, we will perform numerical study on the two metrics.

To reduce the number of “faked” live peers further, we can make enhancement to the basic *RDS*. It is possible to only consider the peers that make updates within the last period of  $T_u$  before the load balancing action. If a peer is alive at  $t_{i+1}$ , it has surely made an update within the duration  $(t_{i+1} - T_u, t_{i+1}]$ . For the enhanced *RDS*, we have the following theorem.

**Theorem 2:** In the enhanced *RDS*, given the update period  $T_u$  and the load balancing period  $T_{lb}$ , the probability that a peer is a “real” live peer in the peer set for balancing at the rendezvous directory is  $p_{real} = \frac{1}{\frac{\alpha-1}{\beta} T_u + \frac{\beta}{(2-\alpha)T_u} [(1 + \frac{T_u}{\beta})^{2-\alpha} - 1]}$  for Pareto lifetime distribution with  $\alpha > 2, \beta > 0$ .

*Proof:* Please refer to our technical report [9]. ■

We proceed to consider the scalability of *RDS*, which mainly depends on the capacity of the rendezvous directory. In a system with  $N$  nodes, the traffic of update messages is  $W_{RDS} = O(\frac{N}{T_u})$ . It is hard for single rendezvous directory to support too many peers. To improve its scalability, a natural approach is to use multiple nodes to support the rendezvous directory service, e.g., to designate multiple peers by hashing [3] or using distributed structures (e.g., [5]).

The robustness is another issue. We find that, *RDS* is vulnerable to the node-targeted attacks. In existing proposals (e.g., [4], [5]), the positions of rendezvous directories are static and public known to all the peers. By overwhelming the rendezvous directory, malicious peers can block update messages. What is more, in case that the rendezvous directories are occupied by attackers, the service of load balancing will be totally brought down.

### C. Independent Searching Strategy

In *ISS*, the nodes depend on searching to find the nodes with inverse load characteristics, and then form into pairs for load transfer. Either light or heavy nodes can initiate the searching. Here we consider the scenario where light nodes are the search initiators, in order to avoid the phenomenon of “thrashing”.

Searching can be seen as a kind of sampling here. Within each time interval, we assume that each peer can sample  $k$  addresses. In case that multiple peers perform sampling simultaneously, the sample sets of different peers may be overlapped. Under the worst case, they have the same sample set. Here, we only consider the best situation and suppose that there exists an ideal sampling service that could make the

sample sets of peers as independent as possible. For *ISS*, our results are as follows:

**Theorem 3:** For *ISS*, supposing that each peer can sample  $k$  addresses per time interval, then the number of load balancing pairs formed within  $[t, t + 1)$  under the best situation is  $N_p(t) = \min\{h(t), [1 - (1 - \frac{h(t)}{N-1})^{\frac{k\beta}{\beta+s(\alpha-2)}}] \times l(t)\}$ , where  $s$  is the average time for node replacement in case of failure.

*Proof:* Please refer to our technical report [9]. ■

Based on the above theorem, we have a similar iterative relation as Sec. III-B, with which we can compute the convergence time  $T_g$  and  $p_{IB}(T_g)$  of *ISS*:  $h(t + 1) = h(t) - N_p(t)$  and  $l(t + 1) = l(t) - N_p(t)$ . The numerical results are presented in Sec. III-D.

For *ISS*, the scalability depends on whether the participating peer can tolerate the message traffic towards it. If DHT lookup is used as the sampling service, based on the results from [10] and assuming the underlying DHT to be Chord, the message traffic that a peer experiences per interval is given as  $W_{ISS} \leq N \frac{\frac{N}{2} \log_2 N - N + 1}{N^2} \approx O(\log_2 N)$ ; if  $k$ -random walk is adopted as the sampling method instead and assuming the traffic is distributed equally to all nodes, the message traffic for a peer per interval  $W_{ISS} \leq k$ . Under both situations, *ISS* exhibits good scalability.

Compared with *RDS*, the robustness of *ISS* is rather good. There is no server-like entity in the system and the responsibility is dispersed to all the peers, thus *ISS* is less vulnerable to the node-targeted attacks.

### D. Numerical Study

In this section, we perform numerical study about the efficiency of *RDS* and *ISS* under churn based on the theorems in the previous section.

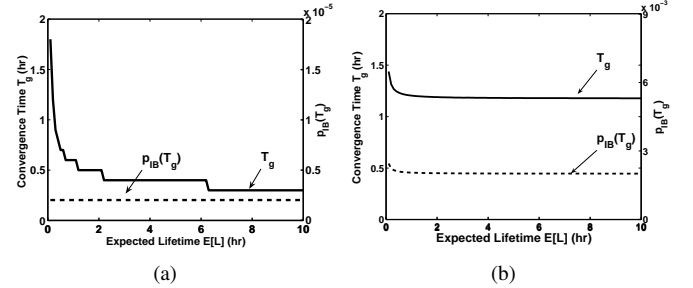


Fig. 1. Impact of System Churn on Load Balancing Strategies under Pareto Lifetime Distribution: (a) Impact to *RDS*; (b) Impact to *ISS*

In Fig. 1, we study the impact of system churn on both load balancing strategies. From Fig. 1, we find that, regardless of the system churn level, the percentage of imbalanced nodes at the convergence time  $p_{IB}(T_g)$  in *RDS* is always much smaller than *ISS*. It means that, *RDS* can lead to a more balanced state compared with *ISS*. The underlying reason is that, it becomes hard to find a heavy node in *ISS* when there are only a few heavy nodes left in the system. The convergence time  $T_g$  of *RDS* is rather good except under very high-churn environment

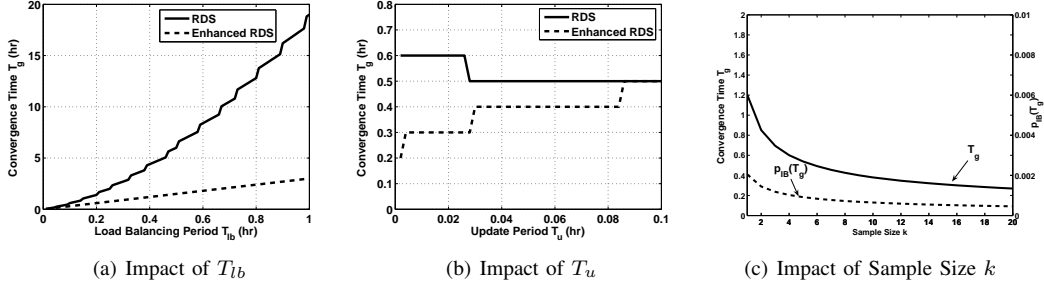


Fig. 2. Impact of System Parameters: (a) Impact of  $T_{lb}$  to RDS; (b) Impact of  $T_u$  to RDS; (c) Impact of Sample Size  $k$  on ISS

(e.g.,  $E[L] < 15$  mins). However, such high churn hasn't been observed in the existing deployed P2P systems.

In Fig. 2, we plot the impact of system parameters. Fig. 2(a) shows the impact of load balancing period  $T_{lb}$  to RDS and its enhancement. With the increasing of  $T_{lb}$ , the convergence time  $T_g$  increases almost linearly. Compared with the basic RDS, the enhanced RDS has a smaller convergence time. It is because that, the number of “faked” live peers is reduced in the enhanced RDS. For different  $T_{lb}$ , we also find that the percentage of imbalanced nodes at the convergence time  $p_{IB}(T_g)$  is almost the same.

The impact of the update period  $T_u$  is illustrated in Fig. 2(b). Compared with  $T_{lb}$ , the impact of  $T_u$  adjustment is not obvious. For the basic RDS, it is interesting to observe that the convergence time  $T_g$  becomes longer when  $T_u$  uses a smaller value. When  $T_u$  is small, it is easier for one peer to make an update within  $T_{lb}$  before leaving. For the enhanced RDS, we can see that  $T_g$  becomes smaller when the update period  $T_u$  is smaller. The reason is that only the updates in the last update period will be considered. With a smaller  $T_u$ , less “faked” live peers will be taken into account. In our result,  $p_{IB}(T_g)$  is also observed to be almost the same for different  $T_u$ .

For ISS, the only tuning parameter is the sample size  $k$  and its effect is illustrated in Fig. 2(c). With the increasing of  $k$ , the convergence time  $T_g$  and the percentage of imbalanced nodes at the convergence time  $p_{IB}(T_g)$  will decrease accordingly. However, the decreasing speed will become slower when  $k$  is bigger. We also find that, even by using a big sample size ( $k = 20$ ),  $p_{IB}(T_g)$  is still much higher than what RDS can achieve. The efficiency of ISS can only be improved in a limited degree by increasing  $k$ .

Overall, even considering the impact of system churn, RDS is still more efficient than ISS in most situations. The main problems of RDS are its scalability and robustness.

#### IV. GOSSIP-BASED LOAD BALANCING STRATEGY

In this section, we propose a *Gossip-Based Strategy (GBS)* for load balancing in DHT systems. The objective of our design is to exploit the efficiency of RDS while improving its scalability and robustness at the same time.

#### A. Gossip-based Load Information Dissemination and Aggregation

GBS is built on top of ring-like DHTs, such as Chord, etc. Based on the system size, one or more load balancing groups are formed among peers. For systems with only a few thousands of peers, only one group is formed; but for systems with hundreds of thousands of peers, the peers form into multiple groups, each of which corresponds to a continuous region with equal size in the ring. All the peers within the same group share the same prefix, which is referred to as the “GroupID”.

In GBS, instead of reporting to a fixed rendezvous directory, each node disseminates its load information within the group by gossip protocol. Here, to reduce the message traffic, a gossip “tree” embedded in the DHT is used for information dissemination. This tree doesn't require explicit maintenance and just expands based on local information. If a node wants to disseminate its load information, it will send the information to all the peers with the same GroupID in its routing table. Next, when an intermediate node  $N_i$  receives the message from the node  $N_{i-1}$ , the one who has already received the information,  $N_i$  only forwards the message to every node  $N_{i+1}$  in its routing table satisfying  $prefix(N_i, N_{i+1}) > prefix(N_i, N_{i-1})$ . Here,  $prefix(x, y)$  is defined as the maximum common prefix between the node  $x$  and  $y$ .

The load information of a node  $N$  to be published includes: (1) the node's capacity; (2) the load of all the virtual servers hosted by the node; (3) the node's ip address. For a small system with only one group, the load information of each peer can be delivered quickly to all members within a short period. After that, each peer has the load information of all other peers.

In case of a large system, there exist multiple groups. Each node  $N$  should run at least two virtual servers. Among them, one virtual server  $N_1$  is with the identifier  $id_1$  generated by hashing the node's IP; and another virtual server  $N_2$  is with the identifier  $id_2$  generated by hashing  $id_1$ .  $N_1$  is called the *primary server* of node  $N$ ; and  $N_2$  is called the *secondary server* of node  $N$ . During load reassignment, these two virtual servers can't be moved to other nodes, but their size can be changed. Given two arbitrary groups, the probability that there exists at least one node  $N$  whose primary server  $N_1$  is in one

group and secondary server  $N_2$  belongs to another group is given by  $1 - e^{-c}$  [11], where  $c$  is the ratio between the group size and the number of groups. With  $c = 5$ , the probability is as high as 0.993. It implies that, for any group, it is with high probability that there exists a secondary server of its group members in any other group. It provides us with an opportunity to aggregate load information of other groups and calculate the system utilization, which is important to achieve the system-wide load balance.

Through gossiping in each group, for a given node  $N$ , it can have the full load information of at least two groups: the group  $S$ , where its primary server  $N_1$  is in; and the group  $T$ , where its secondary server  $N_2$  is in. To let other group members in the group  $S$  learn the load status of the group  $T$ , the node  $N$  also propagates the information about the average load level and the node number of the group  $T$  within the group  $S$  by piggybacking with its own load information. For the secondary servers of the group members in the group  $S$  exist in almost all other groups, after gossiping, every member in the group  $S$  learns the load status of all other groups, and is able to estimate the system-wide utilization independently.

### B. Intra-Group Load Balancing

After load information dissemination, every member within a group holds the load information of the full group and the system-wide utilization information. Each group member has the capability to act as the rendezvous directory to classify the nodes and schedule load reassignment in the group.

However, in order to be robust to the node-targeted attacks, we should randomize the position of the rendezvous directory for each load-balancing round. Although some sophisticated randomization algorithms and security mechanisms can be used to make the position selection be more secure, we adopt a simple but effective approach. It is based on the assumption that, the node-targeted attack is often costly and the probability to compromise a node within a short period is small.

The idea of our approach is as follows: we associate the round of load balancing with a sequence number  $seq_{lb}$ , which is known by all group members.  $seq_{lb}$  will be increased by 1 every load balancing round. In each round, every node locally generates a random key by hashing  $seq_{lb}$  with a common hashing function. The prefix of a generated key should be replaced by *GroupID* to guarantee the node responsible for the key exists within the group. The node that hosts that key is selected as the rendezvous directory. After the completion of load balancing, the node increases  $seq_{lb}$  by one and disseminates the value to all members for consistency checking of  $seq_{lb}$ . In the next round, another node will be chosen as the rendezvous directory due to the randomness of hashing function. So even if the current rendezvous directory is compromised, the service of load balancing can quickly be recovered. As there is no fixed rendezvous directory in the group, *GBS* is more robust than *RDS* to the node-targeted attack.

As the problem of computing an optimal reassignment of virtual servers between heavy nodes and light nodes is NP-

complete, a simple greedy algorithm similar to [4] is adopted here to perform the virtual server reassignments.

### C. Inter-Group and Emergent Load Balancing

Inter-group load balancing is performed only when there exist multiple groups in the system. For the distribution of light nodes and heavy nodes may be various in different groups, it is possible that even after intra-group load balancing, some groups still have many light nodes, while other groups have many heavy nodes. To handle this situation, inter-group load balancing is allowed.

In a group  $S$  with many heavy nodes, its average utilization is higher than the average utilization of the whole system. Suppose that the node  $N$  is the current rendezvous directory of the group  $S$ , due to load information dissemination, the node  $N$  has the full load information of its own group and the load status of all other groups with high probability. The node  $N$  can select one group with the most amount of free capacity to perform inter-group load balancing.

The process is as follows: Given the inter-group load balancing to be performed between the group  $S$  and the group  $T$ , the rendezvous directory  $N$  of the group  $S$  first finds another node  $N'$  within the group  $S$  whose secondary server exists in the group  $T$ , and then transfers the responsibility to  $N'$ . Since  $N'$  has the full load information of both the group  $S$  and the group  $T$ , it can perform the best-fit load reassignment among the nodes of the two groups. With inter-group load balancing, a system-wide balanced state can be achieved.

*GBS* can also provide emergent load balancing. In case that the utilization of one heavy node is beyond a threshold, it can directly contact the lightest node within its group; if there exist no light nodes in its own group, based on collected load information of other groups, it selects the group with the most amount of free capacity and sends a request to one random node in that group, which will return a light node to relieve its load.

### D. Performance Analysis

*GBS* is conceptually similar to *RDS*, therefore, their efficiency should be comparable.

The main concern is the scalability of *GBS*. Let  $g$  be the group size. In small or medium-size systems,  $g = O(n)$ ; in large systems, the ratio between the group size and the group number  $c$  is adjusted to make  $g = O(\sqrt{n})$ , however, the value of  $c$  cannot be too small in order to guarantee efficient information dissemination between groups. In the evolution process of the system, the groups are split only when the size of the current group is over a threshold.

The update traffic that a peer experiences per interval is  $W_{GBS} \approx O(g)$ . Given  $T_u = 30$  seconds,  $n = 250,000$ ,  $c = 4$  and message size = 30 bytes,  $W_{GBS}$  is about 5k bytes/sec for normal peers. The traffic is rather low and tolerable even for the low-bandwidth users.

The robustness of *GBS* is greatly improved compared with *RDS*. By gossip-based information dissemination, it is hard for

malicious peers to block the update messages. The randomization of rendezvous directories enables *GBS* resistant to the node-targeted attacks.

### E. Experimental Evaluation

We evaluate the performance of *GBS* on top of a Chord simulator, which is enhanced to support virtual servers. Synthetic traces are generated to simulate the system churn. In the experiment, the node capacity follows a Gnutella-like distribution, and the average node capacity is able to serve 100 requests per second. Initially, each node is assigned 5 virtual servers. Similar to [5], we also assume the size of a virtual server satisfies exponential distribution. The update period and the load balancing period is set as 30 and 180 seconds respectively.

The workload is generated by the requests from all the nodes uniformly, but the destinations are chosen from either uniform or Zipf distribution. To serve a request will put one unit of load on the destination node. In case that a request is served by an overloaded node, the request is called as the “*ill request*”. We run the simulation for a rather long period, and take the average percentage of “*ill requests*” among all the requests as the metric to evaluate the effectiveness of different load balancing algorithms.

For comparison, we also simulate another four strategies: (1) Rendezvous Directory Strategy (*RDS*); (2) Independent Searching Strategy (*ISS*); (3) Proportion strategy: the nodes remove or add virtual servers to make the load proportional to their capacity; (4) No load balancing for the system.

Fig. 3 plots the effects of load balancing strategies under different degrees of workload. Under the uniform workload, *Proportion*, *ISS*, *RDS* and *GBS* can all guarantee a low percentage of ill requests even with a high request rate. *GBS* is slightly worse than *RDS*, but outperforms *ISS* and *Proportion*. Under the skewed workload, all the strategies perform worse, but *GBS* and *RDS* still can retain a low percentage of ill requests. The reason is that, both *GBS* and *RDS* can utilize the global information to achieve a more balanced system state.

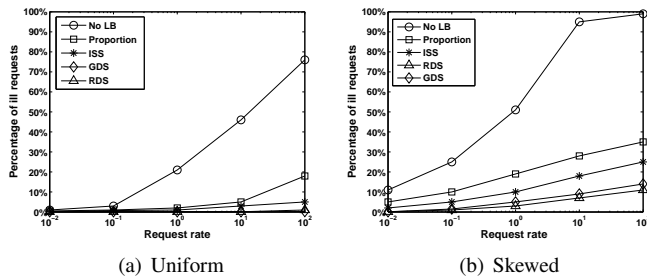


Fig. 3. Percentage of ill requests under different degrees of workload (a) Uniform workload; (b) Skewed workload (zipf distribution with parameter 1.2).

In Fig. 4, we examine how the churn impacts the load balancing strategies by varying the average peer lifetime. It can be observed that, under both uniform and skewed workload, compared with *ISS* and *Proportion*, *GBS* and *RDS*

can always lead to a more balanced system state and minimize the percentage of ill requests. But *GBS* and *RDS* perform slightly worse when the churn rate is higher. The reason is that, a high churn rate will increase the convergence time to the balanced state, which will cause the increase of ill requests accordingly. However, the impact is slight to the efficiency *GBS* and *RDS*. The above results also correlate well with our analytical results in Sec. III.

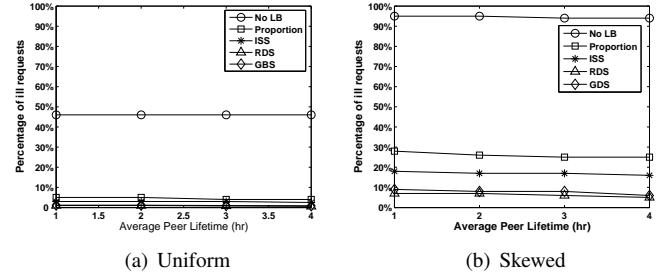


Fig. 4. Percentage of ill requests under different churn rates (a) Uniform workload; (b) Skewed workload (zipf distribution with parameter 1.2).

In summary, we find that *GBS* can achieve almost the same efficiency as *RDS*. But different from *RDS*, *GBS* also achieves scalability and robustness at the same time.

## V. CONCLUSION

In this paper, we have studied the effectiveness of two representative load balancing strategies in DHT-based P2P systems: *Rendezvous Directory Strategy (RDS)* and *Independent Searching Strategy (ISS)*. Through a simple analytical model, we analyzed their efficiency, scalability and robustness. Later, we proposed a *Gossip-Based Strategy (GBS)* for load balancing in DHT systems, which achieves both the efficiency of *RDS* and the robustness of *ISS*.

## REFERENCES

- [1] J. Byers, J. Considine, and M. Mitzenmacher, “Simple load balancing for distributed hash tables,” in *Proc. IPTPS’03*, 2003.
- [2] M. Naor and U. Wieder, “Novel architectures for p2p applications: the continuous-discrete approach,” in *Proc. of ACM SPAA*, 2003.
- [3] A. Rao, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica, “Load balancing in structured p2p systems,” in *Proc. IPTPS’03*, Feb. 2003.
- [4] B. Godfrey, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica, “Load balancing in dynamic structured p2p systems,” in *Proc. IEEE INFOCOM’04*, Hong Kong, Mar. 2004.
- [5] Y. Zhu and Y. Hu, “Efficient, proximity-aware load balancing for dht-based p2p systems,” *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 16, Apr. 2005.
- [6] X. Wang, Y. Zhang, X. Li, and D. Loguinov, “On zone-balancing of peer-to-peer networks: Analysis of random node join,” in *Proc. of ACM SIGMETRICS*, 2004.
- [7] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica., “Wide-area cooperative storage with cfs,” in *Proc. of 18th ACM SOSP*, 2001.
- [8] S. Saroiu, “Measurement and analysis of internet content delivery systems,” *Doctoral Dissertation, University of Washington*, Dec. 2004.
- [9] D. Wu, Y. Tian, and K.-W. Ng, “On the effectiveness of migration-based load balancing strategies in dht systems,” in *CUHK Technical Report*, <http://www.cse.cuhk.edu.hk/~dwu/files/lb-tr.pdf>, 2006.
- [10] N. Christin and J. Chuang, “A cost-based analysis of overlay routing geometries,” in *Proc. IEEE INFOCOM’05*, 2005.
- [11] C. Tang, M. J. Buco, R. N. Chang, S. Dwarkadas, L. Z. Luan, E. So, and C. Ward, “Low traffic overlay networks with large routing tables,” in *Proc. ACM SIGMETRICS’05*, 2005.