Peer-to-Peer Networking and Applications (2020) 13:368–387
https://doi.org/10.1007/s12083-019-00773-9

A balanced strategy to improve data invulnerability in structured P2P system

Xiaogang Qi[1] · **Min Qiang**[1] · **Lifang Liu**[2]

Received: 3 March 2018 / Accepted: 5 June 2019 / Published online: 19 August 2019

Abstract
With the rapid development of computer network and the fever of office automation, data storage gains more and more attentions. In recent years, P2P system-one of data storage networks, has been studied widely because of its high performances (e.g. in terms of distributed and extensibility). One of data storage challenges in it is how to improve data survivability under churn, which is caused by nodes' joining or leaving in the network. Increasing replicas is a comparatively good way to enhance the system fault tolerance, especially under high churn. According to existing references of the replication strategy which always focus on specific P2P systems, we propose a balanced replication method based on the zones partition-*BRBZs* which mainly consists of the copy distribution mechanism, the query mechanism and the consistency maintenance mechanism. It is a general way that *BRBZs* can be applied in different P2P networks. Meanwhile, we evaluate the performance of the *BRBZs* strategy by applying it in DHT systems with different routing protocols to compare with other replication strategies. Performance indexes are innovatively divided into dynamic and static indexes, making the evaluation of algorithm performance more comprehensive and accurate. Simulation results declare that the *BRBZs* has not only better data availability, higher query efficiency and lower searching fail-rate, but good scalability to be applied in systems with different sizes. Besides, the system applying *BRBZs* is convenient for management and maintenance as its user access interface bases on different protocols without centralized servers.

Keywords P2P system · Churn · Data invulnerability · Data redundancy

1 Introduction

P2P, one of the most important distributed computing platforms, has attracted great attention due to its high scalability and invalidity of useless of central nodes. P2P application is originally used for MP3 download. After that, it is adopted by many other areas such as file sharing, live streaming, instant messaging, network storage, grid computing and etc. Nowadays, P2P has been widely used in block chains [1], distributed online social networks [2], wireless networks [3], and the Internet of Things [4]. P2P offers a novel communication paradigm that allows users in the same network to share information and communicate with others [5, 6]. Data owners and requesters are two roles in P2P systems, the owners hold a set of data objects and share them to their corresponding requesters [7]. That characteristic eliminates the asymmetry of node function in the traditional client/server models. That is, users in the P2P system can be partners, contributors and controllers of their resources, and such feature indicates a great

✉ Min Qiang
catherine_mqiang@163.com

Xiaogang Qi
xgqi@xidian.edu.cn

Lifang Liu
lfliu@xidian.edu.cn

1 School of Mathematics and Statistics, Xidian University, Xi'an, Shaanxi, China

2 School of Computer Science and Technology, Xidian University, Xi'an, Shaanxi, China

potential use of P2P in many advanced decentralized applications [8]. [9–11] show that P2P outperforms the centralized servers in terms of autonomy, scalability, sustainable, and more importantly, P2P allows using dynamic resources of network edges, such as information storing/computing source.

P2P systems can be categorized two types: *structured* and *unstructured* types, which can be further subdivided into *full-distributed* and *semi-distributed* molds, based on whether the system has constraints in network topology and data placement [12]. Each node in structured P2P has a query table to aware a small subnet of nodes in systems ensuring the requester node search destination node and its data objective, efficiently and dispersedly. In recent years, structured P2P has been developed rapidly as the DHT protocol was proposed. Addressing and storing the entire DHT network can be achieved by distributing a small scale of routing addresses and a limited subset of data to each participating node, wherefore it is a distributed storage without servers, such as CAN [13], Pastry [14] and Chord [15]. DHT system can be highly dynamic, as nodes can freely join or leave the platform, which may lead to various issues such as the routing failure, the storing data loss and the inconsistency of information maintaining [16].

First, we review some important existing references of the replication strategy in DHT system, as nowadays replication is widely used to improve data availability, reliability and maintainability under churn [17–19]. Then, we propose a balanced replication strategy based on zones partition- *BRBZs*, which is composed of the distributing mechanism, the searching mechanism and the maintaining mechanism. After that, we apply our solution in DHT systems with Chord and Pastry protocols, respectively. Finally, by comparing our method with current state of the art, we find that our solution can achieve a better performance under churn and can be used in different DHT systems. The remainders of this paper are organized as follows:

Section 2 overviews related works and introduces three major categories of replication methods to improve fault-tolerance under churn in next section (Section 3). We details our replication technique based on three protocols: publication protocol, query protocol and maintenance protocol in Section 4. *BRBZs* is evaluated by three performance indexes in Section 5, followed by conclusion in Section 6.

The contributions of this study are:

1) We propose the distribution mechanism of *BRBZs*: first, partitioning the system space based on replicas to make the algorithm scalable to different network size; then, mapping replicas to proper nodes according to the searching table in selected routing protocol and corresponding zone information. Thus, it is convenient for maintenance of originating node as well as improving query efficiency;
2) Proposing a maintenance mechanism based on different triggering conditions can reduce the maintaining cost effectively, and the warning measure of departure nodes can improve system robustness even in high churn;
3) In addition to common evaluation indexes- Hops and searching fail-rate, we propose a static index-data availability, to estimate the data state under malicious and random attacks;
4) We extend the OMNet++ [20], for simulating and evaluating the replication algorithms;
5) We derive a calculation on maxHops、aveHops and minHops related to the network size and replica factor.

2 Related Work

Characteristics of dynamic and decentralized storage in structured P2P overlay network make nodes joining or leaving be a prime issue from the beginning. System Performances are improved by different replications, such as perfecting the system structure partially [21–24], gaining a better solution based on comparison or limit methods [25–27] and so on. Abraham, I., et al. [21] proposed a balanced network not only to maintain performances of infrastructure in overlay network such as smaller network diameter and effective routing mechanism, but also to improve scalability and resilience of system under a high churn. Li, S. and al [25] firstly observed the data invulnerability by comparing the full copy with block copy under the network size is a constant value and the failure nodes is a variable value. Then, they proposed a hybrid algorithm-DCDS algorithm, to acquire more useful data with the increasing number of failure nodes. Simulation results show the data storage strategy based on validation mechanism of dynamic center have good data availability under large-scale node failure or drop periodically. Besides, it can prevent useful data from reducing suddenly and substantially. Medrano-Chávez, A.G., et al. proposed an evaluating framework to estimate influences on various DHT systems caused by different parameters and to measure the churn rate quantifiably in [22]. Applying it in Chord and Kademliaz protocols showed storing data on nodes with lower churn rate was a good way to defend the data invalidation. Lam, S. and Liu, H. [26] analyzed how many failure nodes the system with different protocols could tolerate, then considering the impact of the node joining, proposed a service restoration strategy on the basis of k-neighborhood consistency. After researching various methods referring to the churn-tolerance and the resistant limitation under high churn, Liu, Z. and al [27] proposed a node survival strategy based on critical collapsing point-DARE. It is made up of three phases: the detection phase to help nodes find crash state spontaneously, the selection phase triggered by only one neighborhood node and the rejoining phase to reduce node workload. Aiming to the dynamic change of node set, Kuhn, F. et al. in [23] designed a novel hash table to gain the whole system state at any time without the influence of node joining or departing. It has characteristics of higher churn-tolerance, lower node degree and network diameter. Considering the data availability with user offline, Guidi, B. and al [24] proposed novel P2P online social network to ensure users control access by themselves and distinct from other strategies, its replica node set dynamically changed instead of statically defining. Besides, information diffusion and data availability are serviced by exploiting trust relationships. Silva, T. et al. [28] proposed a storage mechanism of global dictionary based on hash table and a query mechanism based on Hilbert Space Filling Curve (HSFC). Simulation resulted it can be applied in the environment with dynamic change of network scale, fail-node number or data size. Ohmata, H. and al [29] presented a maintenance protocol based on content distribution system to improve the churn-tolerance. It was mainly consisted of updating mechanism of multiple nodes at same time and data immigration mechanism triggered by failure nodes. Simulations under high churn indicated it could heighten data invulnerability and scalability without increasing the amount of network bandwidth.

Replication is the relatively good way to deal with churn problem. Therefore, we focus on three major categories about it in next section.

3 Replication techniques in the decentralized storage system

Problems of decentralized storage in structured P2P system are solved by hash computing. The core of this solution can be implemented by three steps: 1) transform the object characteristic (keyword) to the key value through hash function, namely, mapping all objects to a range of values; 2) distribute different and specific storage scope to each participating node; 3) choose proper and corresponding nodes to store resources according to the key value. Besides, it is used to locate the correct and nearest node when the resource is needed to find.

One character of P2P system is dynamic, which is caused by nodes joining, departing or failing without controlled. It severely impacts the data invulnerability, making the necessity to design a churn-defended storage method, namely, to design a replication strategy to protect data from churn. Various replication methods aiming to decrease searching fail-rate, aveHops and improve data availability need a node set to store data and its replicas. Enlarging the node set is good for improving the data availability, but the reverse is true in decreasing the cost and bandwidth. Existing replication strategies are classified into three types: relevant –physics features replication, models replication and application backgrounds replication, according to [30–41].

3.1 Replication related to physics features

Replications based on physic features are the traditional replication algorithms, such as replication based on neighborhood nodes and replication based on paths. Every node in DHT system has a searching table, which contains information about other m nodes. While different routing protocols determine diverse intervals between nodes of query table. For examples, it is $2i$ (i is the array subscript in table) in Chord protocol and achieves the binary search, while in leaf set of query table of Pastry, it stores information of $|L|$ nodes which are closest to the current node in the key space. Deriving from different protocols and aiming to reduce query complexity, some methods called relevant-neighbor nodes replication are proposed. Its replica node set consists of nodes which are near to the file owner in searching table. Distributing replicas to successor or predecessor nodes of originating node with query table, is named successor/predecessor replication (*SR/PR*) [30, 31], while delivering data to predecessor and successor nodes is known as leaf replication(*LR*) [32, 33]. Data invulnerability can be improved in that way, because successor/predecessor node is able to replace failing source node.

In replication on path, node will not copy the data object in the path until it reaches the final destination [34], such as Tapestry [35]. In the CPU protocol of controlled updating propagation [36], data is copied to all nodes in searching path and index caches are created asynchronously when to response the search query. In conclusion, it can reach a high expected value of data invulnerability when data requesters distributed in non-intersect dense regions, while in system where data requesters are scattered across it, this way would degrade system performances.

3.2 Replication related to models

Model replication is to construct selection models of replica nodes, according to node property (interdependency, identifier correlation), node attribute (centrality) and evaluating indexes. Ghodsi, A. at al. [37] presented a replication strategy based on identifiers, which copied data of i node to corresponding r identifier nodes and associated each identifier with other r identifiers (r is a replica factor) based on the function m: $I \times F \rightarrow I$, defined as:

$$m\ (i, f_r) = i \oplus (f_r - 1)N/f_r \tag{1}$$

where i denotes the logical identifier of each peer received from an identifier space, f_r denotes the identifier i is associated to f other identifiers to achieve replication degree r and N expresses the identifier space.

Mohammad, A.K. et al. [38] designed a measure method of node centrality- EasyRank to make sure that lower-degree nodes can gain higher scores, especially one-degree node. Base on that, a replication strategy considering node location and storage is proposed:

$$Rind = w\_c*centrality + w\_s*storage \tag{2}$$

where w_c and w_s are weights related to centrality and storage, respectively. *Rind* is a score and storage availability can be improved by giving priority to lower *Rind*, when to select replica nodes.

Huang C. Q et.al [39] built reliability model of data nodes based on CPU of data node and I/O of network, and then, built reliability model of network links by choosing data amount of communication event and data rate as variants. After that, they proposed a method to judge reliability degree of replica service, combining the access intensity of data nodes. Besides, selection algorithm of replica nodes, distribution initialized algorithm of replicas, distribution adjustment algorithm of replicas and deletion algorithm of redundant copies are proposed to improve data invulnerability and decrease storage cost. Kermarrec, A.M. et al. [40] proposed a method to improve data availability under churn by allocating replicas to pairs of nodes with lowest relevance.

3.3 Replication based on application backgrounds

Different applied environments make different network topologies in P2P and different emphases. The topology in social network is related to the social relationships of participating users, Mohammad, A.K. et al. [38] use centrality measurement to avoid bias of storing availability without global information. Node importance of social network is related to not only the number of neighborhood nodes, but their quality. The problem of replica nodes selection can be turned to work out the optimal solution of double objective problem with storage constraint and node importance constraint, after sorting nodes importance by EasyRank. Nakashima, T. et al. [41] proposed a replication strategy which focuses on consistency maintenance of file-sharing systems in P2P-SMART (Scalable Consistency Maintenance of File Replicas Based on Trees), and it allows end users to update sharing files. The main idea of SMART is to construct static trees containing a longest path from the root node to each file node, and updating information can be initialized by sending message to the root node. Besides, each node needs to maintain a constant number of its neighborhood nodes, ensuring that updating messages can be propagated to its all replica nodes effectively.

3.4 A contrastive study

After a brief introduction of different replication strategies used to protect system from the churn, we can argue that:

Every replication has advantages and disadvantages, Replications related to physics features not only increase the probability of data to be searched but also decrease look-up hops, which unfortunately, can be only used in specified DHT systems. For example, replication in successors can be used in Chord because each node maintains a list of its successor nodes in Chord. Reverse is true in replications based on models, but it needs to handle complicated data structures for assuring the replica factor r or the weight factor w. Even though the path replication improves the data availability under churn, it can not guarantee a certain number of replicas, which may result the load imbalance between nodes. Aiming to improve data availability and decrease load expense, we propose a replication strategy for most DHT systems based on the zones partition- *BRBZs*, mapping replicas to corresponding nodes, uniformly and efficiently.

4 Replication strategy based on zone partition

Churn, associating with the change of the size of node set, may cause the degradation of system performance, the inconsistency of updating or searching data, even the loss of storing data. Some replication methods should be considered to avoid above problems. As many replication algorithms are only suitable for given systems under the premise of the analysis of 3.4, *BRBZs*, containing the distributing protocol, the searching protocol and the maintaining protocol, is designed to widely apply in different DHT systems. Distribution mechanism is responsible to uniformly map replicas to corresponding nodes after partitioning zones; query mechanism is mainly related to the specific routing protocol; while maintenance mechanism is in charge of the consistency of data under update or churn environment. Systems using the *BRBZs*, are not only high resilient under churn, but convenient for management because its user access interfaces are suitable for different DHT protocols. These systems can be divided into the application layer, middle layer and the routing layer, as shown in Fig. 1.

4.1 Distribution mechanism

4.1.1 Algorithm overview

The main idea of *BRBZs* is to divide nodes into different zones according to the replica factor and ensure that each zone have data or its copy. The Distribution mechanism is presented as follows:

Firstly, we map IP of participating nodes to the liner space by hash function; secondly, we derive the total number of nodes in each zone (*numberNode_zones*) based on the the function *f*:

$$f : numberNode\_zones = Number\_nodes/r \qquad (3)$$

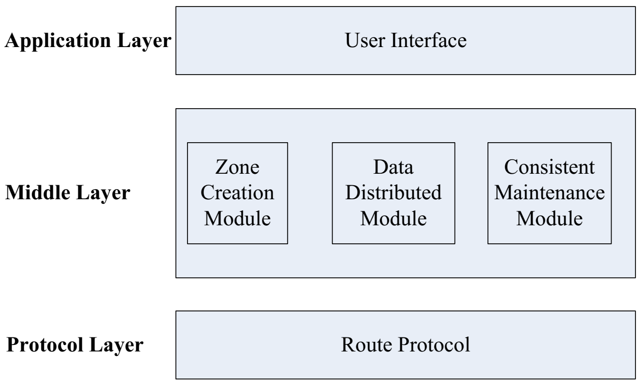

Fig. 1 Structure diagram of system

where *Number_nodes* represents the network size of storage systems and r is the replica factor.

then partitions nodes to different zones after creating a list of zones ($List\_zone_j(N_i)$) to record information of specific nodes(N_i) in each area; lastly, so as to minimize look-up hops, maps data and its copies to corresponding nodes according to the zone information and the searching table of the given routing protocol, after establishing two list recording specific node IDs and its corresponding zones of replication set, which are represented by $List\_replica\_set(N_i)$ and $List\_replica\_zoneset(Z_{Ni})$. Besides, considering the SR has a relatively smaller advantage of aveHops when network size is small, which is caused by the property of finger table of Chord, we introduces a threshold (θ) to judge whether choose a successor node as a replica node. The θ is related to the network size. The Pseudo code is shown in Algorithm 1.

4.1.2 Algorithm description

As shown in Algorithm 1, inputs of Distributing Algorithm are the replica denoted by *r* and identifier length denoted by *m*. The *r* is related to the number of zones and the *m* determines the network size. Firstly, we can gain the number of nodes in each zone from inputs by Eq. (3).

We need a list to record area adscription information of nodes ($List\_zone_j(N_i)$), after gaining the number of nodes contained in each zones and mapping nodes into logical space by hash function (Algorithm 1, Line 1). The *count* is a temporary variable to hold the results of an expression($count = N/r$) and also one of conditional values to judge the current node should belong to the present zone or the next zone. When nodes contained in a zone reached the critical value, subscript of $List\_zone_j(N_i)$ j should be added one (Algorithm 1, Lines 2–13). Every node in system has a search table and in order to improve system performances, we prefer the node which can be found in search table of the data source node as the first replica node. Meantime, it cannot be in the zone with the data source node to make sure each zone have a copy after distributing finished. Then, add the first replica node to the $List\_replica\_set(N_i)$, a list created to record node IDs, and add its zone number to the $List\_replica\_zoneset(Z_{Ni})$ to easily find the zone containing data when we needed. Regarding the first replica node as the node of data source and repeating above steps, the second replica node can be selected (Algorithm 1, Lines 17–21). And so on, we can finally gain a set of replica nodes, the output of Algorithm 1. Besides, in simulation stage, we notice the SR has a relatively small advantage in Hops performance with Chord protocol when the network size is small, which may be caused by the nature of finger table of Chord. Thus, we set a threshold θ to judge the replica node selecting principal follows *SR* or *BRBZS* (Algorithm 1, Line 14–16). Simulation results show *BRBZs* achieve low expected value of the Hops performance when increasing the network size to 1024, so we set θ to 1024.

Algorithm1Distributing Algorithm

```
Input: replica $r$, identifier length $m$;
Output: the proper set of replica nodes.
/*Initiate the Node identifiers*/
1 $N_i$=hash(IPnode$_i$);
/*Assume a Boolean to judge whether the zone number $j$ needs to plus one or not*/
2 Boolean flag=0;
/*Associate node to a specific zone*/
3 count =0;
4 For $i$=1 to $N$ do
5       If(i %(N / r)<N/r && count <=N/r )
         /*Assign nodes to the corresponding zone*/
6       List_zone$_j$($N_i$);
7      count++;
        /*Assure the proper and fixed size of nodes contained in a zone */
8      if(count %(N/r)==0)
9          flag=1;
10           j++;
11          count=0;
12           if(flag==1)
13               flag=0;
             End if
        End if
      End if
End For
14 if($\theta$< $N$)
15   List_replica_set($N_{i+1}$);
16   List_replica_zoneset($Z_{Ni+1}$);
End if
17 For $i$=0 to $r$, do
18   For $j$=1 to $N$, do
         /*Find the node is not only in the search_table, but distinguished from the current zone */
19        If( $N_j \in$ Search_table_i && !$N_j \in$ Zone($N_i$))
             /*Save the list of the replica nodes*/
20           List_replica_set($N_j$);
21           List_replica_zoneset($Z_{Nj}$);
        End if
     End For
End For
```

Figure 2 illustrates the function of the distributing protocol, where copies are stored in proper nodes in different zones on the basis of replica factor $r = 4$ and the identifier length $m = 4$. Indeed, the fact that we create r replicas of each file and distribute each of them in different zones improves the data availability and reduces the searching time.

4.2 Query and maintenance mechanism

As it can be applied in different DHT systems, *BRBZs* needs an efficient query mechanism to reduce searching hops and improve its generality. The query mechanism can be divided into the coarse localization of zone based on the key value of requesting node and the accurate searching on the basis of routing protocol of the system. After confirming the nearest zone from the requester, it searches data by half in accordance with finger table of chord protocol; while in pastry, it will not search by neighborhood set until queries based on leaf set and routing table both fail.

Because each replica node is in the finger table of the last replica node, we make the source node maintain the first replica node, the first maintain the next,..., and the last maintain the source node, which can reduce maintenance cost. Besides, it can periodically detect replica factor, file availability and source consistency. Then, considering the data migration, we take into account two maintenance strategies: the basic maintenance and the periodic maintenance of the data consistency (Algorithm 2).

4.2.1 Algorithm overview

Basic maintenance is triggered when peers join or leave the system. In case of the node i joining, pointer information in node i-1 and node $i+1$ are need to modify according to the routing protocol of the given system, which are presented by *Pointer_table*[i-1,i] and *Pointer_table*[i,$i+1$] respectively. When node i leaves the system, it informs node i-1 and node $i+1$ to change their pointer tables and gain data in node i by delivering a warning message to node i-1 or node $i+1$. We

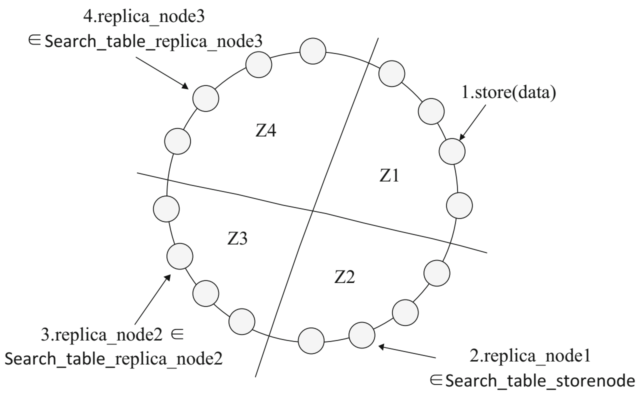

Fig. 2 An example of distributing protocol with 4 zones using chord

have to stress here that migrating data to the predecessor or the successor node depends on if the node i is the last node in a zone. Besides, *One-to-One maintaining method* is in charge of the data modification when originate node updates.

Periodic maintenance is launched by the given time interval and can improve churn-defended of DHT systems. The target is twofold: first, creating a list(*nodeList(source_node(replicaKeys*[i]*)*)) to store the replica node information of each source node makes each source node can periodically contacts its all replicating nodes to ensure the data consistency (*maintainReplicas*); second, each node needs insuring that it only maintains data itself (*verifyReplicas*). The Pseudo code is shown in Algorithm 2.

4.2.2 Algorithm description

For the maintenance algorithm, the first thing is to make sure the type of changes triggered it, which can be easily achieved by calculating the number of current nodes in system and comparing it with that of before (Algorithm 2, Lines 1,4). Whatever the case, the node i-1 and node $i+1$ need to modify their pointer tables and in the latter case, there exists a judgment for the data migrating. The node location can be gained through *List_zone*$_j$(N_i) from Algorithm 1, and if it is the last node in its zone, migrating the data to its predecessor, otherwise to its successor (Algorithm 2, Lines 2–3, 5–10). *One-to-One maintaining method* means each replica node is maintained by one node in basic maintenance caused by distributing protocol. For example, node j is in charge of node i, j will send message to compare key value of i with itself when it is triggered. If Boolean expression is false, the data in i would be changed (Algorithm 2, Lines 11–15). We have to stress there is an assumption that the data only be updated by the data source node.

In periodic maintenance, the *maintainReplicas* is mainly in charge of the data consistency by sending message to nodes in *nodeList(source_node(replicaKeys*[i]*))* through the data source node and creating two sets to record data loss and data inconsistency information, respectively (Algorithm 2, Lines 16–22). Aiming to decrease maintenance cost, the *verifyReplicas* is proposed. If the *replicaKeys.length*(i) is not equal to that of source node, Lines 25–26 of Algorithm 2 would be worked, otherwise, Lines 23-26would be worked.

5 Evaluation

In order to prove the *BRBZs* strategy is a general and better method of replicas placement in the DHT system with different route protocols, we compare it with other replication strategies by using the same parameters to evaluate their performances. In view of the fact that the path replication can not

```
Algorithm2 Maintenance Algorithm
i. basicMaintainReplicas
/*judge the type of changes*/
/*a node is joining in the system*/
1 if($\sum{}'node_i = \sum node_i + 1$)
   /*Modify the corresponding nodes pointer_table*/
2 Pointer_table[i+1]←Pointer_table[i,i+1];
3 Pointer_table[i-1]←Pointer_table[i-1,i];
   /* a node is departing from the system */
4 Else if ($\sum{}'node_i = \sum node_i - 1$)
5          Aware (Node[i+1]);
              /*Modify the corresponding nodes pointer_table */
6          Pointer_table[i+1]←Pointer_table[i,i+1];
7          Pointer_table[i-1]←Pointer_table[i-1,i];
              /*Migrate data in Node[i] to the Node[i-1] or the Node[i+1]*/
8          Judge(List_zone_j(N_i));
9          replica = replicaList[i];
10         replicaNode[i+1].addReplica(i) or replicaNode[i-1].addReplica(i);
           Else
           /*Executing the One-to-One maintaining method */
11      For n=0 to r, do
12         For i=1 to r,do
13               replicaNode[n]=replicaList[i];
14               if (!replicaNode[n].getKey_Value. equals(replicaNode[n-1].getKey_Value))
15                     replicaNode.Replica(myKeys(i))= replicaNode (myKeys(i-1));
                  End if
              End For
           End For
       End if
End if
n.matainReplicas
16 For i=0 to replicaKeys.length do
17    root =nodeList( source_node(replicaKeys[i]));
18    While(i.next()!=0)
       /*Judge whether there exists a data loss in each storage node or not and create a
       set(recordData_Loss_nodeList) to record data loss information*/
19      if(root(i).getKey_Value==0)
20            recordData_Loss_List.put(N_key(i));
        /*Judge whether there exists an inconsistency between storage nodes or not and create a
        set(recordData_Inconsistency_nodeList) to record data inconsistency information */
       Else
21         if(hash'(keys(i))!=hash(source_keys))
22               recordData_ Inconsistency _nodeList.put(N_key(i));
           End if
       End if
    End While
n.verifReplicas
/*Judge whether there exists a repeated maintenance in each storage node or not and create a
set(recordRepeated_Maintenance_nodeList) to record repeated node information and a
set(recordRepeated_Maintenance_keyList) to record repeated information */
23   For i=0 to replicaKeys.length, do
24     if(replicaNode[n].getValue(replicaKeys(i))!=sourceNode.getValue(replicaKeys(i)))
25         recordRepeated_Maintenance_nodeList. put(replicaNode[n]);
26         recordRepeated_Maintenance_keyList. put(replicaKey (i));
       End if
   End for
```

guarantee the minimum degree of replicas and has a relative high expense in storage, and meanwhile, the feature replication needs to handle complexity data structure, we estimate the BRBZs performance in different systems by using the *PR* and the *SR* as contrasts under Chord protocol, while selecting the *LR* as the contrast under Pastry protocol because it has a better performance in Pastry protocol.

Assuming that the system consists of fixed size (*N* nodes), the identifier length *m* is 9 both in Chord and Pastry protocols, parameter *b* in Pastry protocol is 2, threshold θ related the networks size is 1024 and the distribution mechanism has been done in initial stage. The dataset is about a special network with 98742KB. It contains 370700 data that consist of 36 attributes such as location information, object name, event impact, net name, alarm title and so on. We use its subset about 2223 KB, containing 19472 data that consist of 7 attributes (location, net name, alarm start time, alarm end time and so on) by deleting data on empty information of important fields and incomplete information of fields.Each experiment in simulator platform of OMNet++ starts from the initial loading stage and the number of look-up message is set by 2000. Besides, the evaluations both in Chord and Pastry are done based on the following metrics: the data availability (the static performance of algorithms), the number of Hops and the searching fail-rate (the dynamic performance of algorithms).

Average Hops(aveHops) is a common parameter to evaluate performances of P2P overlay network. It relates to the number of nodes between the requester node and the destination node. Besides, we add maxHops and minHops to evaluate its performance, which are calculated by the mean maximum or minimum hops of multiplicating repeated experiments, to make simulation results more comprehensive and more accurate.

Data availability is divided into availability under random attack and availability under malicious attack. It can be regarded as a random attack when part of nodes in network drop or fail owing to some unavoidable external factors, such as power failure and line fault. We observe that the performance of data availability of different algorithms by choosing failure nodes randomly to mimic the random attack. While the malicious attack is caused by selecting failure nodes based on some feature of network such as topology structure of network, and we regard the remaining storage space as the feature of the network in this paper. Readers can directly observe *BRBZs* performances by coordinate the coordinates under different conditions, namely, the number of failure nodes is replaced with churn rate in our simulations because it can be thought of as a special churn.

We add a packet loss rate following a uniform distribution in [1,200] to make the simulation results more realistic and set a searching fail signal when the query message returns to the requester node to estimate the searching fail-rate under churn:

$$fail\text{-}rate = the\ number\ of\ fail\ messages/ \\ the\ number\ of\ look\text{-}up\ messages \tag{4}$$

5.1 Effect of the number of replicas

We evaluate the effect on performances of algorithms caused by the transformation of the number of replicas under fixed churn rate (0.5), based on Hops, data availability and searching fail-rate metrics. It is generally known that the number of replicas is proportional to the robustness of system under churn and the reverse is true in storage space and maintaining expenses. Therefore, we set the number of replicas to 2, 4, 6 and 8, respectively.

Firstly, we evaluate the effect of number of Hops (minHops, aveHops, maxHops) with *r* replicas. From Fig. 3, we can notice that the number of Hops decreases slightly with the increase of the number of replicas in both replications techniques. We can observe that the *BRBZs* achieves a smaller number of aveHops in Pastry protocol than in Chord protocol, namely, the differences are 1.0141 hop and 0.703 hop in worst and best conditions, respectively. This is due to the fact that the searching table of Pastry additionally introduces conceptions of the leaf set and the neighborhood set compared with the routing protocol of Chord, which reduces the query paths and increases the searching speed. Besides, the performance of Hops decreases hardly when replica reaches 8 and its minHops has an obvious advantage compared with other replications. From Fig. 3.a and 3.b, it can be observed aveHops and maxHops performances of BRBZS are more sensitive with the change of the number of replicas under Chord protocol, while its minHops is more suspectible under Pastry protocol. We use the calculation (5) to drive more intuitionistic conclusions about *BRBZs* performances:

$$X\_decreaseRate = [(X_m)max-(X_m)min]/(X_m)max \\ -[(X_n)max-(X_n)min]/(X_n)max \tag{5}$$

where *X* represents aveHops, maxHops or minHops, *m* represents *BRBZs*, while *n* represents *PR*, *SR* or *LR*. *BRBZs* decreases more quickly than other replication strategies from Fig. 3, so we use *X_decreaseRate* represent the difference of decreasing rate between *BRBZs* and other replications. The calculation results are shown in Table 1.

Figure 4 illustrates the effect of the number of replicas on the data availability with respect to the random attack and the malicious attack, respectively. It mainly estimates the effect on the probability of retrievable files of system, which is primarily

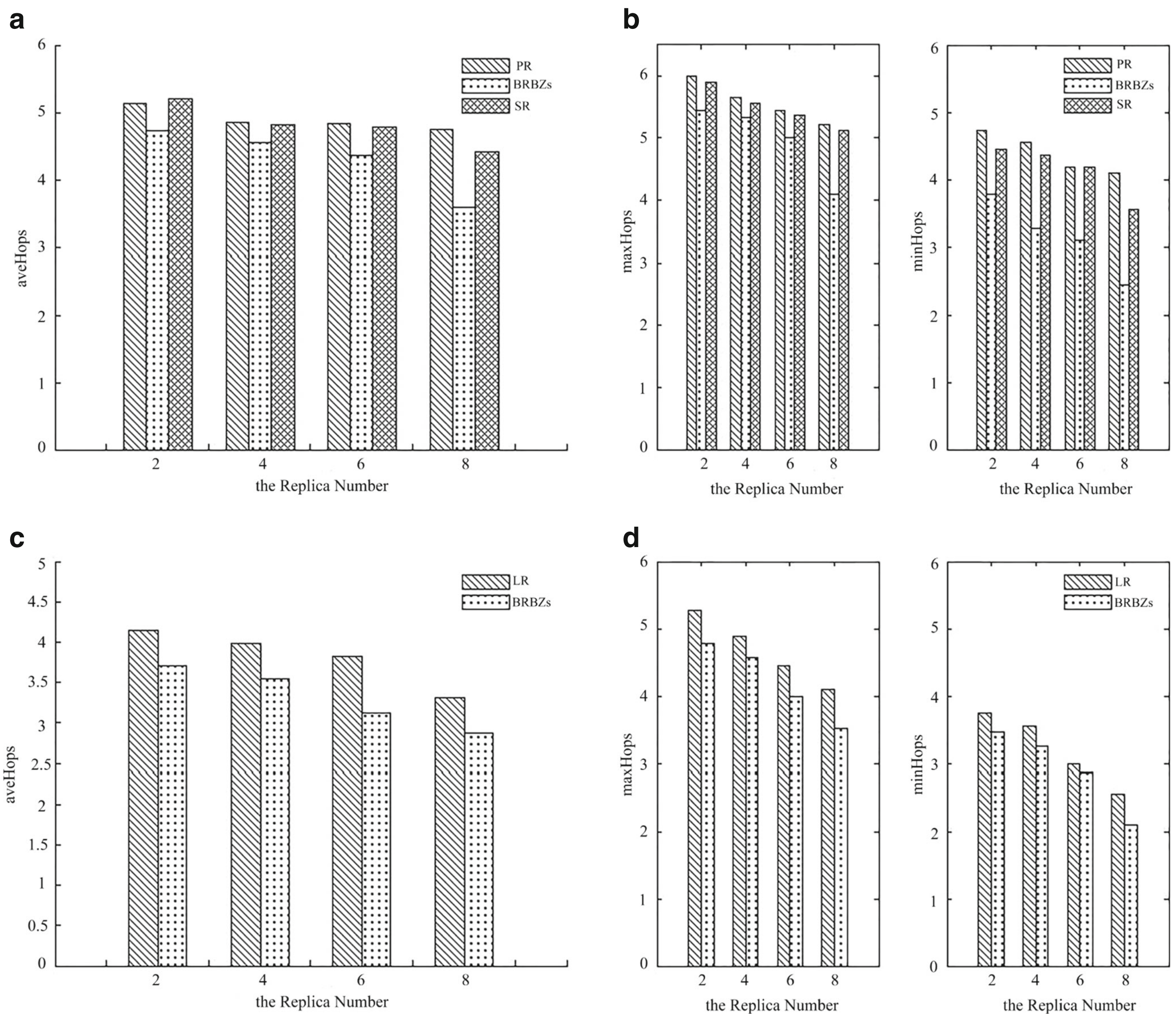

Fig. 3 **a** Effects of the number of replicas on the aveHops under Chord Protocol. **b** Effects of the number of replicas on the maxHops and minHops under Chord Protocol. **c** Effects of the number of replicas on the aveHops under Pastry Protocol. **d** Effects of the number of replicas on the maxHops and minHops under Pastry Protocol

Table 1 the decreasing difference between BRBZs and other replications

X_decreaseRate	X	n
+17.87%	aveHops	PR
+10.2%		SR
+2.44%		LR
+11.65%	maxHops	PR
+11.43%		SR
+4.16%		LF
+13.77%	minHops	PR
+6.82%		SR
+7.43%		LF

caused by the distribution mechanism of different replication algorithms. Therefore, we directly simulate those replications without considering any protocol of DHT system. As the space has been divided based on the number of replicas and each zone maintains one copy, the data availability of *BRBZs* algorithm is highest in both random and malicious attack. Superiority of replication algorithms under random attacks is *BRBZs* > *LR* > *PR* > *SR*, when to consider the performance of data availability combined with the storage expense. We have to stress the *LR* algorithm has better performances in both storage spending and data availability when it is under the malicious attack. In malicious attacks, replica 4 is a watershed. Before it, the superiority of replication algorithms shows the same with that of random attacks, and after it, the superiority of replication algorithms is

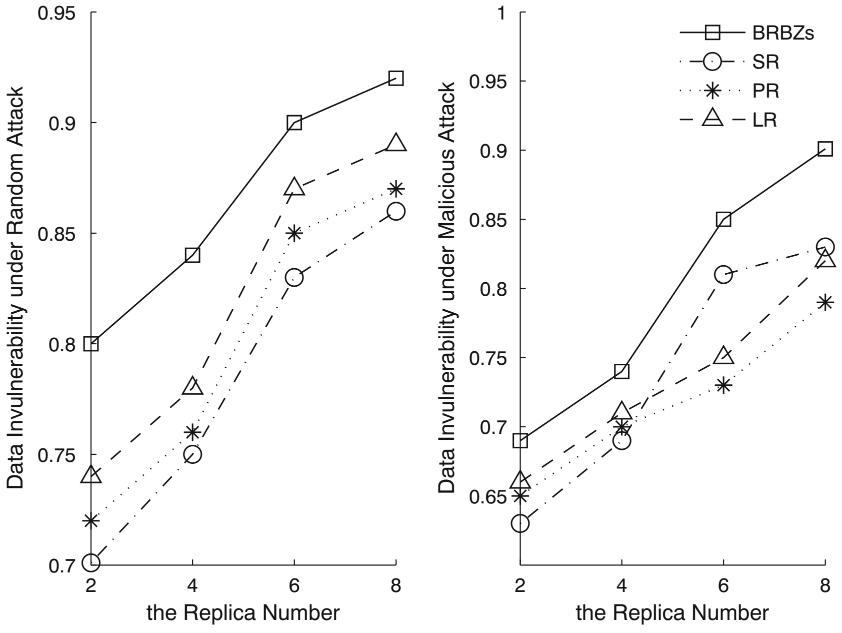

Fig. 4 Effects of the number of replicas on the data availability under random and malicious attack, respectively

BRBZs > *SR* > *LR* > *PR*. With the increasing number of replication factor, *SR* gradually shows its advantages and becomes the best storage strategy except *BRBZs*, if we just focus on data invulnerability.

For the searching fail-rate, it varies inversely with the number of replicas from Fig. 5. Initiating 2000 look-up messages and assuming the distribution mechanism exists loss rate with uniform probability, we notice the *BRBZs* has lowest searching fail-rate both in Chord and Pastry protocols, since its distribution and consistency maintenance mechanisms. Besides, the searching fail-rate of the *BRBZs* decreases sharply when the replica is 4, after that, it transforms to the gradual decrease and finally ends by the slight decline. In other words, the replica 4 is the best solution in condition of a fixed churn rate (0.5), if we only regard searching fail-rate and storage cost as considerations. Besides, it can improve fail-rate performance by 13.2% compared with the best performance of other replications when the replica is 4.

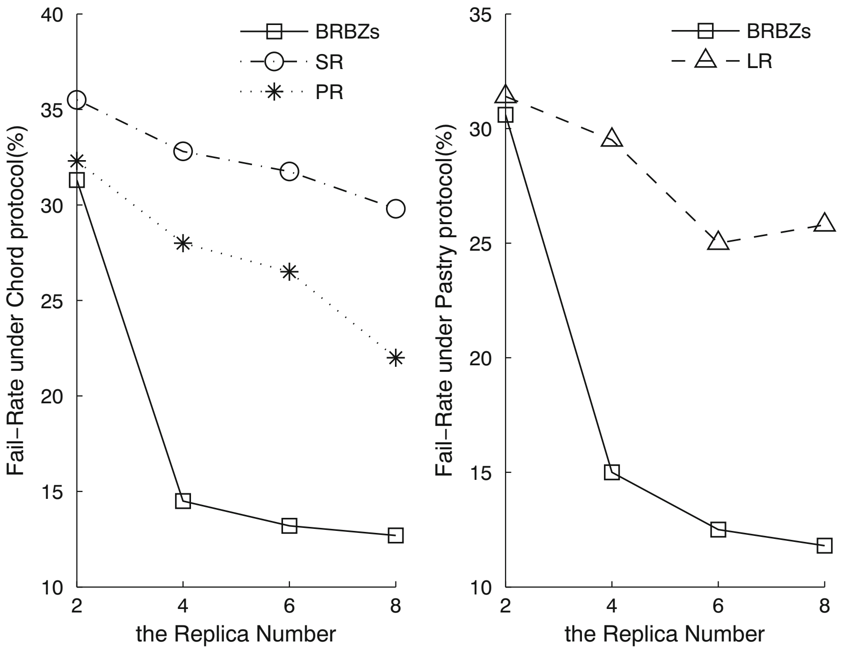

Fig. 5 Effects of the number of replicas on the searching fail-rate

5.2 Effect of churn rate

After that, we analyze the performance of algorithms in Chord and Pastry protocols with a fixed replica and a variable churn rate. Increasing the churn rate can damage the performance of DHT systems, while the replication techniques weaken these effects. Accordingly, we set the number of replicas at 4, which performs good performance with respect to the fail-rate and storage expense from Fig. 5 and the churn rate at 0.1, 0.2, 0.3, 0.4 and 0.5, respectively.

In Fig. 6, the number of Hops increases with the decreasing of participating nodes. *SR* and *PR* have even difference of hops, such as aveHops and minHops, while *BRBZs* has a greater performance about them from Fig. 6.a and Fig. 6.b. Besides, we can notice that the aveHops under Chord protocol is more sensitive to the variety of churn rate, compared its mean change of the slop of Fig. 3.a is 0.1902 with that of Fig. 6.a is 0.208. While in Pastry protocol, it is more susceptible to the changes of the number of replicas, compared its mean change of the slop of Fig. 3.c is 0.1383 with that of Fig. 6.c is 0.1125.The maxHops of *BRBZs* increases significantly under Chord protocol and the reverse is true in minHops under Pastry protocol, because it can be calculated the average increment of maxHops under Chord is 0.5137 and that is 0.225 under Pastry. For minHops, it is 0.1494 under Chord while 0.325 under Pastry.

In a stable network and for different replication algorithms, more than 69% of data are usable (Fig. 7). Raising the number of failure zones makes data availability of different replications show various falling trends and *BRBZs* has greatest data availability both in two conditions of the attack. In addition, we notice the *LR*

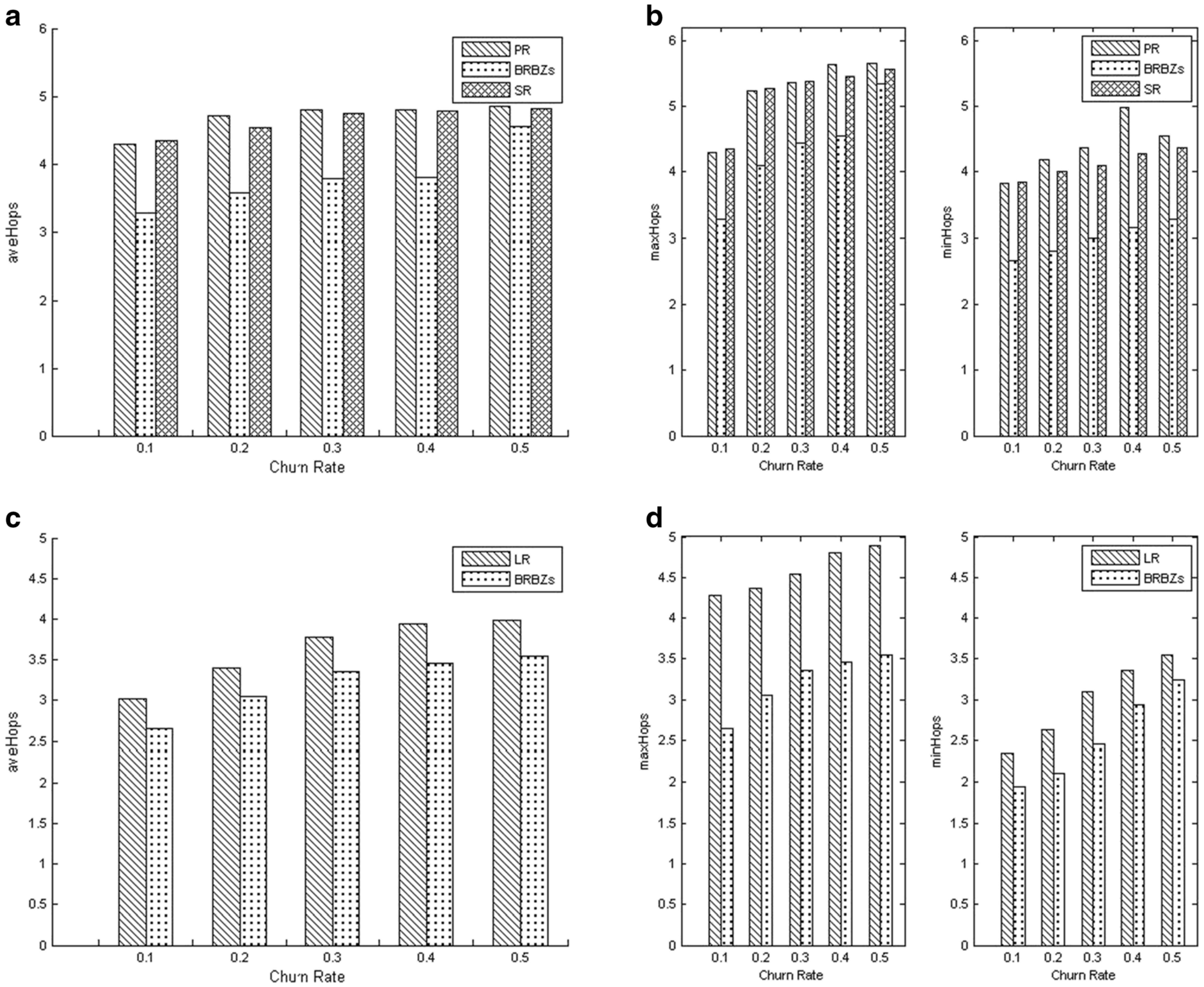

Fig. 6 **a** Effects of churn rate on the aveHops under Chord Protocol. **b** Effects of churn rate on the maxHops and minHops under Chord Protocol. **c** Effects of churn rate on the aveHops under Pastry Protocol. **d** Effects of churn rate on the maxHops and minHops under Pastry Protocol

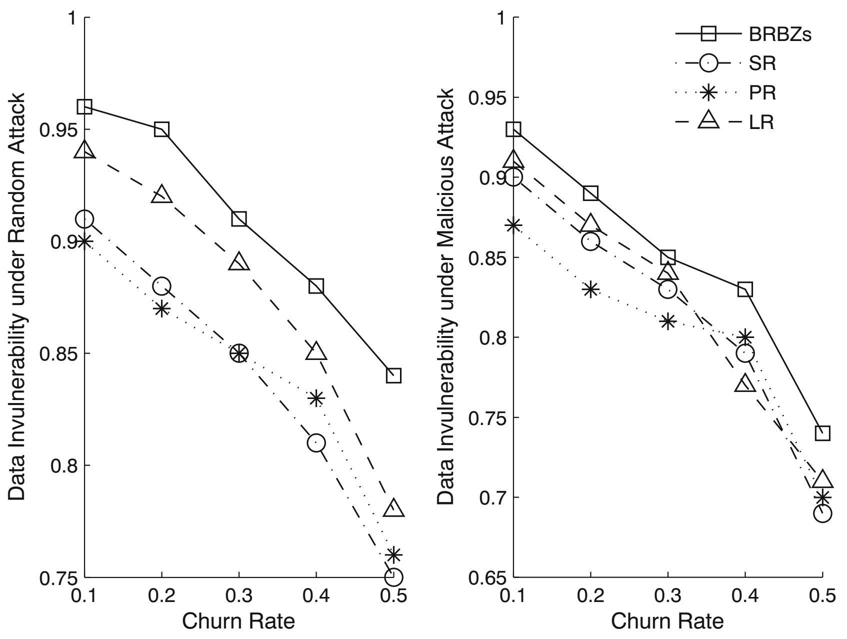

Fig. 7 Effects of the churn rate on the data availability under random and malicious attack, respectively

algorithm and the *SR* algorithm can not maintain a comparatively good performance of data availability under malicious attack, especially when the churn rate is greater than or equal to 0.4. But the reverse is true for the *PR* algorithm after the churn rate reaches 0.4 (Fig. 7). Besides, we can notice the *SR* is superior to *PR* with smaller churn rate(churn rate < =0.3) and a fixed replica (r = 4) under random attacks from Fig. 7, while *PR* is superior to *SR* with a fixed churn rate (churn rate = 0.5) and various of replicas from Fig. 5.

From Fig. 8, we can see that the churn rate also affects the searching fail-rate in both overlays. As the overlay size decreases, the searching fail-rate increases. We notice that *BRBZs* achieves a relatively low fail-rate, which is due to the fact that such technique has an even distribution mechanism and a maintenance mechanism taking into

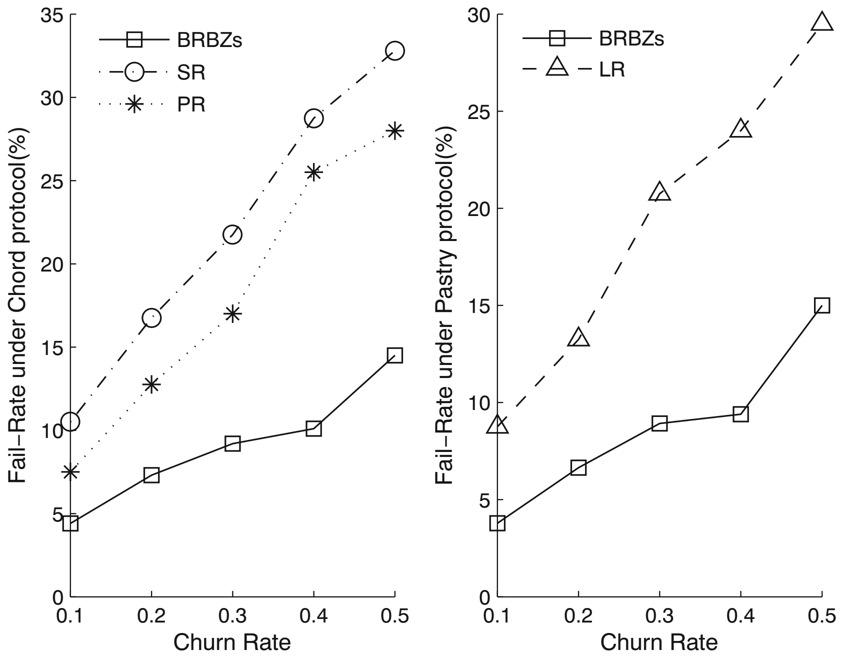

Fig. 8 Effects of the churn rate on the searching fail-rate

account the data immigration. Noting that the fail-rate of the *BRBZs* algorithm sharply increases after the churn rate reaching the 0.4, while other replications sharply increase before the churn rate reaching the 0.4, meaning the *BRBZs* also has a good churn-tolerance. Besides, we gain the mean changes of the slope angle of *BRBZs* from Fig. 8, which are 3.14° and 6.84°, under Chord protocol and Pastry protocol respectively. Meantime, we plan to put the designing of churn warning mechanism as one of our future works, the warning mechanism can make the churn rate drop to a certain value that is less than some threshold, after it triggered.

5.3 Performance of the scalability

We determine to present the detailing behaviors of the scalability of the *BRBZs* strategy under Chord protocol and compare it to the *PR* strategy and the *SR* strategy. We fix the number of replica to 4 cause the *BRBZs* has a relative balance between replicas and fail rate from Fig. 5, and set the churn rate at 0.5, considering some references set vales of the churn rate up to 0.5, then, vary the network scale (512, 1024, 2048, 4096 and 8192 peers).

From Fig. 9, we can conclude that not only the aveHops performance, but maxHops and minHops, the *BRBZs* strategy is superior to that of other replication strategies. Hops performances are proportional to the network size if we fixed the replica factor and the churn rate. From Figs. 10 and 12, we can observe the *BRBZs* has obvious advantages of searching fail-rate and data availability under malicious attacks. The change trends of data availability of replication strategies curve slightly from Fig. 11, because the random attack is one of gentle attacks. We have stress that the worst performance of the *BRBZs* in that condition improves data availability by 5.1% compared with the best performance of other replications. Above analysis prove the proposed strategy can effectively improve data invulnerability and query performances (searching fail-rate and aveHops), which further leads to a better churn-defended performance in different DHT systems. The simulation results disclose that for a fixed replication and a certain churn rate, the performance of the *BRBZs* algorithm is independent of the network size. Besides, the scalability analysis confirms that the results obtained in the 512-node scenarios are equally effective with larger system scale Fig. 12.

5.4 Analysis of the *BRBZs* strategy

5.4.1 Performance of the *BRBZs* strategy

First, we study the relationship on hops, network size and replica number; then research a dual objective optimization problem based on aveHops and fail-rate as metrics, namely study the replica optimization of BRBZs on the basis of the analysis of network size and churn rate.

The DHT system maps physical nodes into logical nodes in a ring space through hash functions and according to different routing protocols, the hops in worst condition may be different. That is, the maxHops, which is related to the query table of routing protocols in the DHT system, can be expressed by $\log(\delta N)$. δ is a parameter related to the function of the routing protocol $f(c_r, c_d)$. c_r and c_d are the request node and the destination node, respectively. The distribution mechanism of BRBZs, relying on zones partition related to replica factor r, guarantees each zone have one data, that is, there exists a mirror image between each zone and the maxHops drops from $\log(\delta N)$ to $\log(\delta N/r)$. The relationship on the maxHops(h_m), the network size (N) and the replica number(r) is f_m:$N \times r \rightarrow h_m$, defined as:

$$f(h_m, N, r) = m - log_2(r) + c \tag{6}$$

where h_m presents the number of hops, N presents the network size and r is the number of the replicas. Besides, m is identifier length and c is a constant related to the routing protocol.

Based on the f_m, the maxHops is proportional to the m when the replica number is fixed, and the reverse is true when the m is fixed, it decreases with the increasing number of replicas. When $r \in 2^i (i = 0,1,2,3,\ldots)$时, maxHops = m-i + c, while r∈Z-2^i,maxHops∈([$\log_2(N/r)$ + c]-1,[$\log_2(N/r)$ + c]) and the [$\log_2(N/r + c)$] can be gained by the *floor*(x). The ideal condition is the request node and the destination node are in the same search table, whose hop is 1, namely the minHops is 1. Therefore, the relationship on the aveHops(h_a), the network size (N)and the replica number(r) is f_a:$N \times r \rightarrow h_a$, defined as:

$$f(h_a, N, r) = (m - log_2(r))/2 + const \tag{7}$$

where h_a presents the number of hops, N presents the network size and r is the number of the replicas. Besides, m is identifier length and $const = (c + 1)/2$, c is a constant related to the routing protocol.

The performance of storage system is proportional to the number of replicas, meaning that increase replicas can improve system robustness and users' query quality, but produce higher costs of storage and consistency maintenance. It is necessary to balance them by calculate relatively good replica number related to the churn rate and the network size. From Fig. 13, we can see that there exists an obvious trend between 4 and 6 replicas with the increasing number

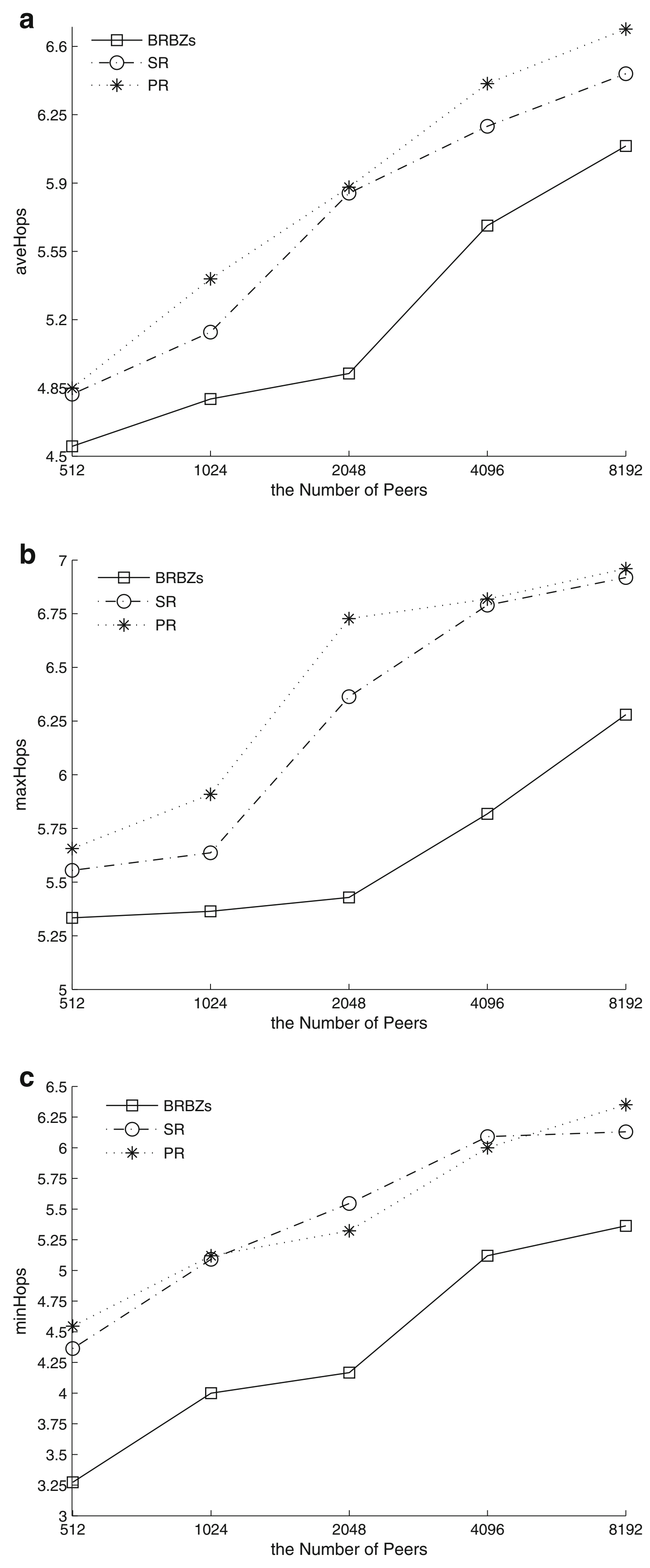

Fig. 9 **a** Evolution of the aveHops. **b** Evolution of the maxHops. **c** Evolution of the minHops

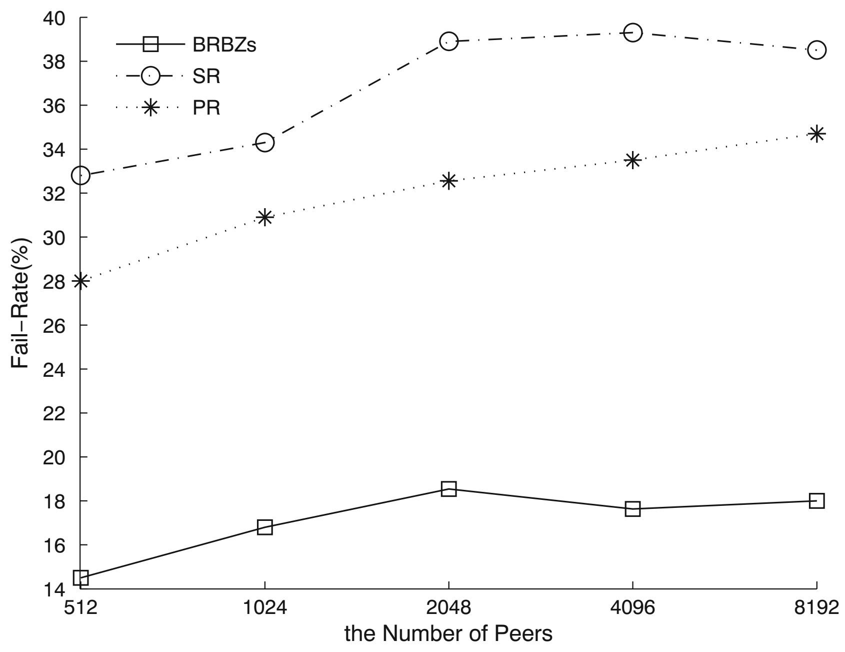

Fig. 10 Evolution of the searching fail-rate

of network size and the churn rate, especially when the network size reaches 2048, increasing replicas from 4 to 6 can decrease aveHops obviously. The trend of the fail-rate gradually pace down with respect to the overlay size from Fig. 14 and the difference of the impacts of the 2 replicas and 4 replicas decreases with the increasing of the overlay size. Different systems focus on different performances makes the weight allocation between fail-rate and aveHops vary. For example, data storage network of document system may focus on the lower fail-rate, the better solution of replica number is 4 in the condition of the smaller network size and lower churn rate, and the replica is 8 when the network size is larger and the churn rate is higher from Fig. 14. While the browser system may mainly

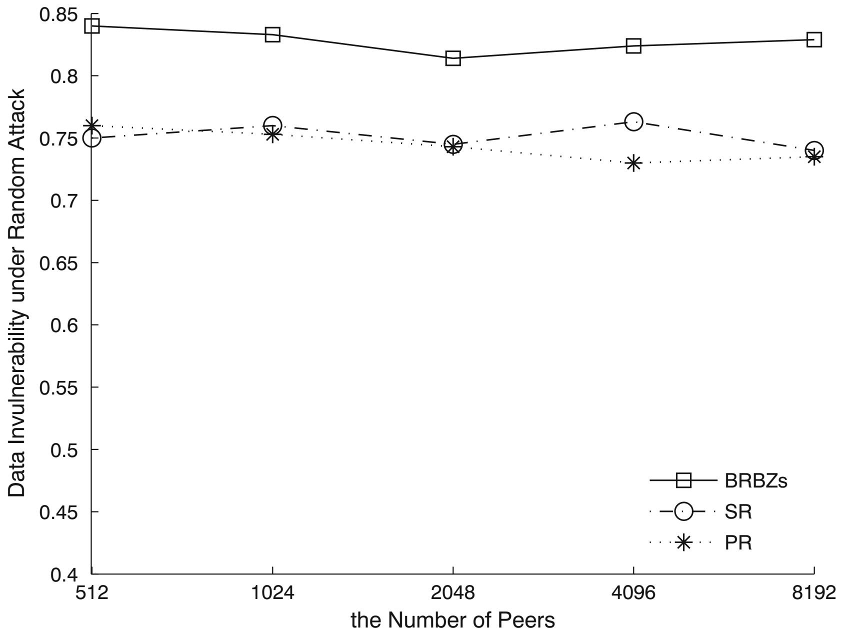

Fig. 11 Evolution of the data availability under random attacks

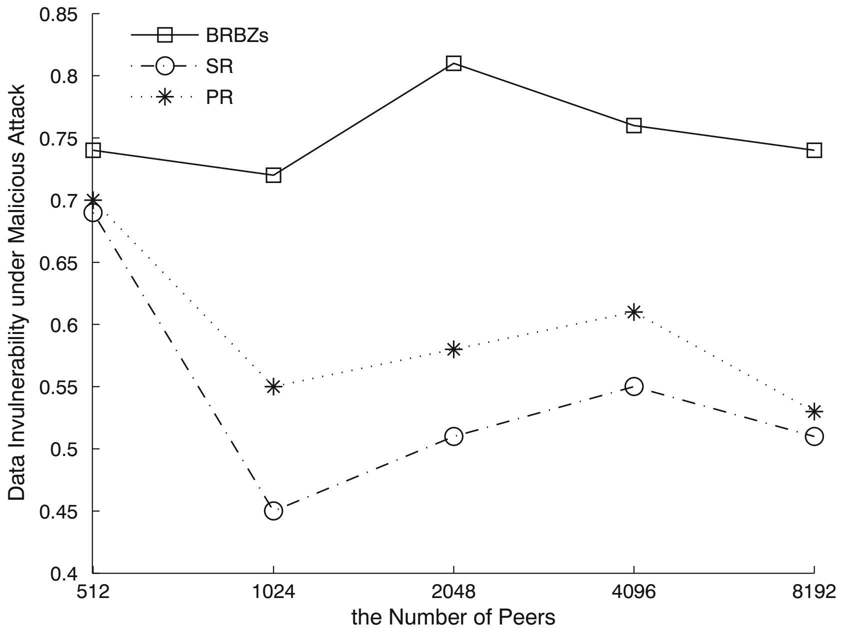

Fig. 12 Evolution of the data availability under malicious attacks

concentrate on high searching speed, which can be evaluated by the aveHops, the Fig. 13 shows the 2 replicas is a better number with lower churn rate and smaller network size and the 6 replicas is the solution for higher churn rate and larger network size.

5.4.2 A contrastive study of *BRBZs* strategy and other replication strategies

According to the analysis above, a summary on BRBZs and other replications is listed as Table 2 shows.

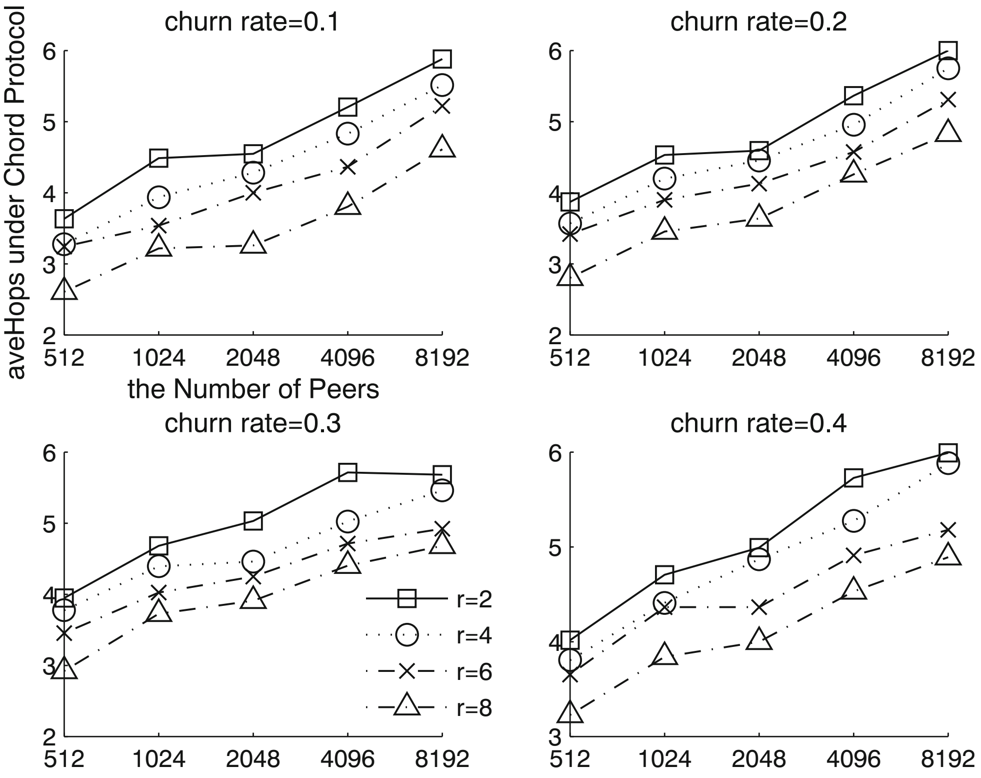

Fig. 13 AveHops of different network size under different churn rate with Chord protocol

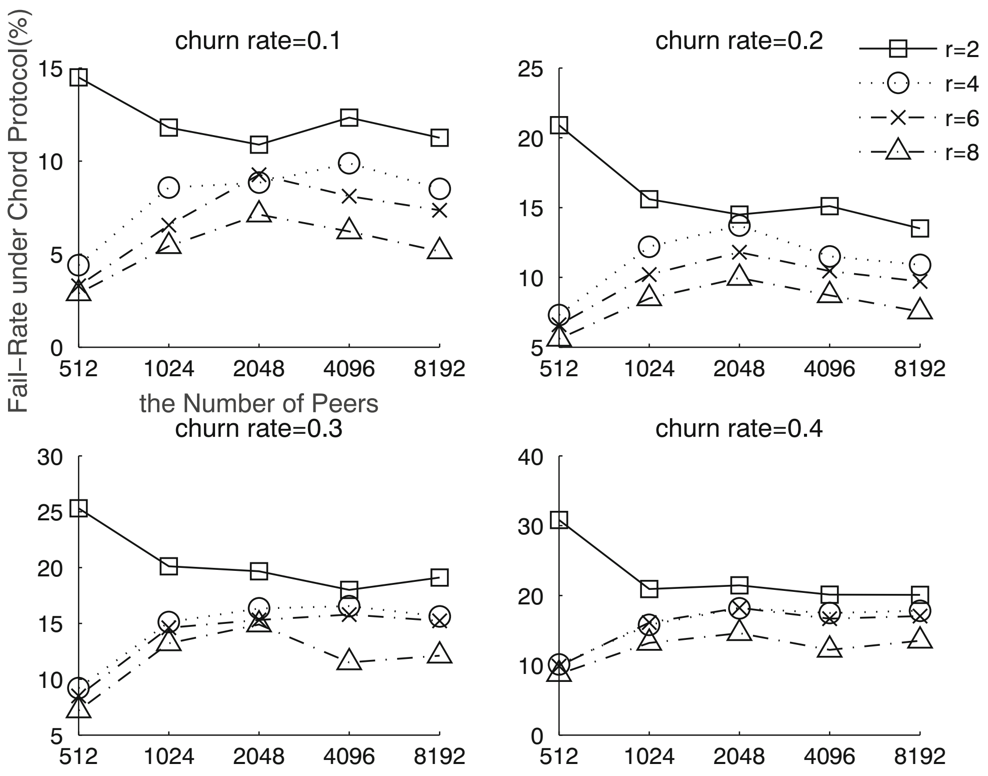

Fig. 14 Fail-Rate of different network size under different churn rate with Chord protocol

Table 2 A table of contrastive study between BRBZs and other replications

Technique	Approach	Weakness	BRBZs-doing
Loo, H [42]	Uses flooding and DHT to locate popular and rare items	Suffers from fault positive error caused by the flood searching	Query mechanism based on routing protocol of different DHT makes query reach at most *m-log*$_2$*(r)* + *c* hops as long as the data requested exists.
LARD [43]	Uses decentralized learning automata to sending the searching information through shortest paths	Cannot guarantee the number of hop	According to routing protocol of DHT systems and attribute of finger table, it automatically choose replica node set, guaranteeing the number of hops in the worst condition.
DARE [27]	A node survival strategy based on critical collapsing point to detect crash state and trigger by one neighborhood node.	High maintenance cost and coarse detection caused by the crash point which is calculated by massive experiments.	Basic and periodic maintenance makes its lower cost and more comprehensive mataining, especially the *One-to-One* method.
Jin, X., et al. [31]	Uses random matrix by shift right logical and successor replication to produce node set of data placement.	High time complexity caused by right logical shifting matrix, especially when r is relatively high and imbalance load.	Distribution mechanism based on replica factor and zone principle ensures each zone have one data, balancing the load of storage network.
Flocchini, P., Nayak, A., et al. [44]	Introduces some redundancy in the system via laying multiple Chord rings on top of each other and using multiple successor lists of constant size.	Only used to small scale of network, because the number of Chord ring and maintenance cost increase linearly with the replica number increasing and the greedy algorithm to forward queries may not guarantee the smaller hops.	The zone number relies on the network size and the replica factor makes BRBZs have better scalability and each zone has one data reduces the hops to m-log$_2(r) + c$.

6 Conclusion

Structured P2P networks produce an efficient and highly resilient platform for large-scale distributed systems to store data and how to design a highly churn-defended strategy is one of challenges of these systems. In this paper, we research on different replication methods dealing with churn problems, at first. Then, we propose a novel replication strategy based on the principal of the zone partition- *BRBZs* and it can be applied in different DHT systems. We extend the OMNet++

simulator and simulate the state-of-the-art decentralized churn-aimed replication algorithms both in Chord and Pastry. In comparison to other replications, the results show that the *BRBZs* algorithm gives better performance in terms of number of hops, data invulnerability and searching fail-rate. From the scalability point of view, the proposed algorithm shows the same performance regardless of the increasing number of the network size. As our future work, we plan to add the popularity of storing data, reliability information of nodes (e.g: the average online duration of a certain period of a user in social network is his own probability of the reliability) and so on as measuring indexes when to choose nodes of a replica set, rather than only depends on the searching table of protocol in given DHT system. The replication strategy considering these factors together is under go, which would likely result in the uniform distribution of replicas among high available nodes as many as possible.

Acknowledgements We would like to thank the two anonymous reviewers for their insightful comments and suggestions, which have helped to improve the quality of our paper. This work is supported by the National Natural Science Foundation of China (Grants No. 61877067, 61572435), Joint fund project the Ministry of Education - the China Mobile (No. MCM20170103), Xi'an Science and Technology Innovation Project (Grants No.201805029YD7CG13-6), Ningbo Natural Science Foundation (Grants No. 2016A610035, 2017A610119).

References

1. Karafiloski E, Mishev A (2017) Blockchain solutions for big data challenges: a literature review. Int Conf Smart Technol. https://doi.org/10.1109/EUROCON.2017.8011213
2. Chen K, Shen H, Sapra K (2016) A social network based reputation system for cooperative P2P file sharing. IEEE Trans Parallel Distrib Syst 26(8):2140–2153
3. Waluyo AB, Taniar D, Rahayu W (2017) Trustworthy data delivery in Mobile P2P network. J Comput Syst Sci 86:33–48
4. Park SH, Park JK (2016) IoT industry and security technology trends. Int J Adv Smart Converg 5(3):27–31
5. Raj P, Raman A, Nagaraj D, Duggirala S (2015) High-performance peer-to-peer systems. In: High-performance big-data analytics. Computer communications and networks, 317–337. Springer, Cham
6. Masseglia F, Poncelet P, Teisseire M (2006) Peer-to-peer usage analysis: a distributed mining approach. Int Conf Adv Inf Netw Appl. https://doi.org/10.1109/AINA.2006.262
7. Yahya HN, Alptekin K, Öznur Ö (2018) Decentralized and locality aware replication method for DHT-based P2P storage systems. Futur Gener Comput Syst 84:32–46
8. Spaho E, Barolli A, Xhafa F, Barolli L (2015) P2P data replication: techniques and applications. In: Xhafa F, Barolli L, Barolli A, Papajorgji P (eds) Modeling and processing for next-generation big-data technologies, 145–166. Springer, Cham
9. Arnedo J, Matsuo K, Barolli L, Xhafa F (2011) Secure communication setup for a P2P based JXTA-overlay platform. IEEE Trans Ind Electron 58(6):2086–2096
10. Barolli L, Xhafa F (2011) JXTA-overlay: a P2P platform for distributed, collaborative, and ubiquitous computing. IEEE Trans Ind Electron 58(6):2163–2172
11. Enokido T, Aikebaier A, Takizawa M (2011) Process allocation algorithms for saving power consumption in peer-to-peer systems. IEEE Trans Ind Electron 58(6):2097–2105
12. Waluyo AB, Rahayu W, Taniar D, Scrinivasan B (2011) A novel structure and access mechanism for Mobile data broadcast in digital ecosystems. IEEE Trans Ind Electron 58(6):2173–2182
13. Ratnasamy S, Francis P, Handley M, Karp R, Shenker S (2001) A scalable content -addressable network. Conference on applications, technologies, architectures and protocols for computer communications. 31, 161–172
14. Rowstron A, Druschel P (2001) Pastry: scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: Guerraoui R (ed) Middleware, 329–350. Springer, Heidelberg
15. Stoica I, Morris R, Karger D, Kaashoek MF, Balakrishnan H (2001) Chord: a scalable peer-to-peer lookup service for internet applications. Conference on Applications, Technologies, Architectures and Protocols for Computer Communications 31, 149–160
16. Malik SUR, Khan SU, Ewen SJ, Tziritas N, Kolodziej J, Zomaya AY, Madani SA, Min-Allah N, Wang L, Xu CZ, Malluhi QM, Pecero JE, Balaji P, Vishnu A, Ranjan R, Zeadally S, Li H (2016) Performance analysis of data intensive cloud systems based on data management and replication: a survey. Distrib Parallel Databases 34(2):179–215
17. Liu GX, Shen HY, Chandler H (2016) Selective data replication for online social networks with distributed datacenters. IEEE Trans Parallel Distrib Syst 27(8):2377–2393
18. Matri P, Pérez MS, Costan A, Bougé L, Antoniu G (2017) Keeping up with storage: decentralized, write-enabled dynamic geo-replication. Futur Gener Comput Syst 86:1093–1105
19. Paiva J, Rodrigues L (2015) On data placement in distributed systems. ACM SIGOPS Oper Syst Rev 49(1):126–130
20. Varga A (2010) OMNeT++. In: Wehrle K, Güneş M, Gross J (eds) Modeling and tools for network simulation. Springer, Heidelberg
21. Abraham I, Awerbuch B, Azar Y, Bartal Y, Malkhi D, Pavlov E.: A generic scheme for building overlay networks in adversarial scenarios. In: Parallel and distributed processing symposium. 9, 40b
22. Medrano-Chávez AG, Pérez-Cortés E, Lopez-Guerrero M (2015) A performance comparison of chord and Kademlia DHTs in high churn scenarios. Peer-to-Peer Netw Appl 8:807–821
23. Kuhn F, Schmid S, Wattenhofer R (2005) A self-repairing peer-to-peer system resilient to dynamic adversarial churn. In: Castro, M., van Renesse, R. (eds). Peer-to-Peer Syst 3640:13–23
24. Guidi B, Amft T, De Salve A et al (2016) DiDuSoNet: a P2P architecture for distributed Dunbar-based social networks. Peer-to-Peer Netw Appl 9(6):1177–1194
25. Li S, Lan T, Ra MR (2016) Background traffic optimization for meeting deadlines in data center storage. Proceedings of the conference on information science and systems, 372–377. Princeton, America
26. Lam S, Liu H (2006) Failure recovery for structured P2P networks: protocol design and performance evaluation. Commun Netw 50(16):3083–3104
27. Liu Z, Yuan R, Li Z, Li H, Chen G (2006) Survive under high churn in structured P2P systems: evaluation and strategy. In: Alexandrov, V.N., van Albada, G.D., Sloot, P.M.A., Dongarra J. (eds). Comput Sci 3994:404–411
28. Silva T, Kamienski C, Fernandes S, Sadok D (2015) A flexible DHT based directory service for information management. Peer-to-Peer Netw Appl 8:512–531
29. Ohmata H, Ishikawa K, Sakakihara H et al. (2011) Churn-resistant method for DHT-based content delivery systems. IEEE Consumer Commun Netw Conf. https://doi.org/10.1109/CCNC.2011.5766536
30. Rahmani M (2014) A comparative study of replication schemes for structured P2P networks. ResearchGate.https://www.researchgate.net/publication/264419857_A_Comparative_Study_of_Replication_Schemes_for_Structured_P2P_Networks. Accessed 4 Dec 2017

31. Jin X, Li D, Yu X et al (2010) Design of Distributed Secure Storage System Based on random matrix and chord. International conference on computer, mechatronics, control and. Electron Eng 6:402–405
32. Rowstron A, Druschel P (2001) Pastry: scalable, decentralized object location, and routing for large-scale peer-to-peer systems. Middleware. 2218:329–350
33. Dabek F, Li J, Sit E et al (2004) Designing a DHT for low latency and high throughput. Proc Nsdi 1:85–98
34. Zhou SY, Wang K, Zhang YF et al (2011) Generalized Minimum Information Path Routing Strategy on Scale-Free Networks. Chin Phys B 20(8):501–505
35. Zhao BY, Huang L, Stribling J, Rhea SC, Joseph AD, Kubiatowicz JD (2004) Tapestry: a resilient global-scale overlay for service deployment. IEEE J Sel Areas Commun 22(1):41–53
36. Li Z, Xie G, Kai H et al (2011) Churn-resilient protocol for massive data dissemination in P2P networks. IEEE Trans Parallel Distrib Syst 22(8):1342–1349
37. Ghodsi A, Alima LO, Haridi S (2006) Symmetric replication for structured peer-to-peer systems. Databases Inf Syst Peer-to-Peer Comput 4125:74–85
38. Mohammad AK, Hillol D, Cristian B (2016) Balanced content replication in peer-to-peer online social networks. Proceedings of the IEEE international conferences on big data and cloud computing, 274–283. Atlanta, America
39. Huang CQ, Li Y, Wu HY (2014) Modeling and maintaining the reliability of data replica service in cloud storage systems. J Commun 35(10):89–97
40. Kermarrec AM, Merrer EL, Straub G et al (2012) Availability-based methods for distributed storage systems. Reliable Distrib Syst 42(1):151–160
41. Nakashima T, Fujita S (2015) Scalable Tree-Based Consistency Maintenance in Heterogeneous P2P File Sharing Systems. Proceedings of the 44th International Conference on Parallel Processing Workshops, 250–256. Beijing, China
42. Loo B, et al. (2005) The case for a hybrid P2P Search Infrastructure. In: Voelker, G., Shenker, S. (eds) Peer-to-Peer Systems III. Springer, 141–150
43. Akbari TJ (2012) A distributed resource discovery algorithm for P2P grids. J Netw Appl 35(6):2028–2036
44. Flocchini P, Nayak A, Xie M (2005) Hybrid-chord: a peer-to-peer system based on chord. In: Ghosh, R.K., Mohanty, H. (eds) Distributed computing and internet technology. 194–203

Xiaogang Qi was born in Baoji, Shaanxi. He is now a Ph.D. supervisor and professor in Xidian University, China. His current research interests include system modeling and fault diagnosis.

Min Qiang was born in Tianshui, Gansu. She received Bachelor's degree from Hunan Agriculture University. She is now a graduate student at Xidian University. Her current research interest includes data storage and data invulnerability.

Lifang Liu was born in Lanzhou, Gansu. She is now a professor in Xidian University, China. Her current research interests include data processing and intelligent calculation.