Peer-to-Peer Netw. Appl. (2016) 9:344–355
DOI 10.1007/s12083-015-0340-4

A novel replication technique to attenuate churn effects

Zied Trifa · Maher Khemakhem

Received: 21 August 2014 / Accepted: 5 March 2015 / Published online: 17 March 2015

Abstract Dealing with the churn problem is an important requirement in distributed systems in general and more precisely in structured peer-to-peer overlay networks due to their dynamicity. This problem refers to the change in the set of participating nodes due to the frequent joins, leaves or failures. Such variations induce at worst the loss of objects and at least performance degradation. This is due to the reorganization of the set of replicas of the affected objects. Till now, this problem has been mostly addressed at the p2p routing level to ensure the reach ability of peers by maintaining the consistency of the logical neighbors. However, the main challenge for structured p2p systems is to protect the ability of the system to locate any present object. At the storage level, avoiding data losses is still an issue when a reconfiguration of the participant peers occurs. In this paper and after presenting first the existing replication techniques, we propose the zone replication technique used in both Chord and Kademlia protocols in order to attenuate churn effects. Finally, we point out some ways that can lead to better and maybe robust replication protocols.

Keywords Structured P2P overlay networks · Churn · Failure · Zone replication

1 Introduction

With the recent development of structured p2p architectures which are evolved from simple file-sharing networks to efficient alternatives to the classic client–server architecture. Each of peers participates in a logical overlay structure and acts simultaneously as client and as server. It is responsible of maintaining its share of information, which can be provided to the other peers that requesting it.

Actually, these systems are very dynamic and each participating peer may join or leave the overlay at any time. This process is referred to as churn. However, this freedom of having a highly dynamic network has its costs. Too high churn can cause routing failures, loss of stored resources or inconsistent views of the peers on the overlay.

In the other side, the main challenge of these systems is to protect the ability of the system to locate any present object. Thus, it is essential that the overlay structure will be maintained even in the presence of high churn. In structured p2p architectures such as Chord [1] and Kademlia [2], where the integrity of the neighbor's relationship among the peers must be kept at all times. As a consequence, they require more maintenance traffic when the churn rate is high. Besides, these systems operate without a centralized control unit and each peer has only a limited view of the entire network, usually not being aware of the effect of this threat in the network.

In this paper, we present an overview of churn problem and discuss different approaches proposed in literature to handle this problem. In particular, we propose our approach called zone replication based on three protocols: publication protocol, search protocol and maintenance protocol. After that, we explore the performance of DHTs (Distributed Hash Table) in such dynamic environments and we show how to obtain a robust estimate, which is independent of the implementation details of the network structure.

The remainder of this paper is organized as follows. In section 2, we analyze churn in order to understand the impact of such problem on the performance of DHTs systems like Chord and Kademlia. Section 2 reviews the related work. A summary of the relevant research work on the replication

Z. Trifa (✉)
Department of Computer Science, University of Sfax, Sfax, Tunisia
e-mail: trifa.zied@gmail.com

M. Khemakhem
College of Computing and Information Technology, University of King Abdulaziz, Jeddah, Saudi Arabia
e-mail: makhemakhem@kau.edu.sa

techniques to deal with churn problem is presented in section 4. We detail our zone replication technique, in Section 5 along with a novel publishing, searching and maintenance protocols. Section 6 evaluates the performance of our technique. Finally, concluding remarks are provided in section 7.

2 Problem statement

In structured p2p overlay networks, there are no barriers to join or leave the network. A peer joins the network when a user starts the application, contributes some resources while making use of the resources provided by others, and leaves the system when the user exits the application. The independent arrival and departure by thousands of peers creates the collective effect called churn [3]. Such freedom causes unpredictable network environment, which leads to the most complex design challenge of p2p protocols.

The impact of joining peers is usually the less problematic aspect of churn. It mainly results in temporary failures like routing or resources inconsistencies, which might be temporarily located at a wrong position in the overlay. The process of peers leaving the system, however, can result in irreparable damage like loss of the overlay structure or loss of data stored in the overlay. In general, node departures can be divided into ordinary churn and high churn. In the first situation, nodes join the network one by one, or leave gracefully by informing their neighbors. However, in the second situation, a large percent of nodes joins and/or silently leaves the network simultaneously and frequently.

Each instance of join or leave causes a DHT to rebalance its keys and the data among the nodes, which generates a considerable traffic load and degrades the performance of the system. Also, this dynamics can cause routing failures, subsequent lookups to return inconsistent results, loss of stored resources or inconsistent views of the peers on the overlay.

Finally, Churn has a significant effect on the performance of such systems. For example, frequent nodes joining and leaving result in stale routing information in the routing table and inconsistency of the stored resource items. The distribution of session length affects the overlay topology and key design parameters. Consequently, the effects of churn should be taken into account when designing or evaluating a DHT system.

3 Related work

Because the dynamic and decentralized nature of structured p2p overlay networks, join and leave process have been a prime issue from the beginning. Each DHT system employs a specific mechanism to deal with such phenomena. But after churning a large set of peers in a short time interval, the system will crash.

Unfortunately, very few studies examine the impact of churn problem. Abraham and al [4] study the scalability and resilience to worst case joins and leaves. They focus on maintaining a balanced network in the presence of high churn rate. Loguinov and al [5] examine graph-theoretic properties of existing peer-to-peer architectures and proposes a new infrastructure based on optimal-diameter de Bruijn graphs. They study routing performance of Chord, CAN and Bruijin. Li and al [6] present a comparative study of the effects of DHT parameters on the performance of structured p2p protocols under churn. Lam and al [7] address the question of how high a rate of node dynamics can be supported by these systems. They present a measurement studies to CAN protocol only. Furthermore, authors in [8] address the question of how to make p2p service available under churn and where is the extreme of a system's resistibility to high churn. They present a measurement study of some overlay network and propose a crash point to deal with such problem. Khun and al [9] present a dynamic distributed hash table where peers may join and leave at any time. The system tolerates a powerful adversary which has complete visibility of the entire state of the system and can continuously add and remove peers. They provide worst-case fault-tolerance, maintaining desirable properties such as a low peer degree and a low network diameter.

To the best of our knowledge the only solution tolerating churn attack is based on replication. In next section, we present the different replication techniques to deal with such phenomena.

4 Replication techniques to deal with churn problem

Distributed Hash Tables are distributed storage services that use a structured overlay relying on key-based routing protocols. DHTs provide the system designer with a powerful abstraction for wide-area persistent storage, hiding the complexity of network routing, replication, and fault-tolerance.

In such systems, each peer and object is assigned an identifier. An object key is usually the result of a hash function performed on the object. The peer whose identifier is the closest to the object key is called the object root.

In order to tolerate failures, each object is replicated on k peers, which compose the replica-set of an object. DHTs systems use different replication methods to overcome the problems associated with the low connectivity of some nodes in the network.

There are so many benefits of using a replication method. Besides a better success rate in case of failure nodes, some performance can be enhanced through replication, such as the search time and the number of hops needed to locate an object. Further, replication can increase the speed of recovery of an object by increasing the number of sources.

There are several algorithms for object replication that store and maintain object periodically. We classify these methods in three categories: Sequential neighboring replication, multiple key replication and path replication.

4.1 Sequential neighboring replication

In DHTs systems, each node maintains a list of sequential neighbors used for routing object in the overlay. These nodes are selected according to the algorithm adapted. In Chord, for example, each node maintains a list of its successors in the ring; replication in the successors can be used. In Kademlia, each route node maintains a list of neighbour's numerically closest called leaf set; replication in the leaf nodes can be used [10].

4.1.1 Replication in the successor's nodes

Replication in the successor's nodes places the objects to replicate in the successor's of the route node. If the root node fails, the object is immediately available in the direct successor, which in turn becomes the new root node [11]. On the other hand, replication in the successor focuses the object around the root node and creates an imbalance in the system, especially if a node has many popular objects. In addition, this method does not decrease the number of hops used to find object as the search queries is first routed to the root node. If we want to have access to different copies of the object, we must first reach the root node.

4.1.2 Replication in the leaf nodes

Replication in the leaf nodes places the objects to replicate in both successors and predecessors of the route node. In DHTs protocols, such as Pastry and Kademlia, where the message flows in two directions, it is interesting to replicate the object in both successors and predecessors. Research can therefore lead before reaching the root node by retrieving the object from a predecessor of route node [12]. In this case, the number of hops is decreased and the search time is reduced.

4.2 Multiple key replication

This approach relies on computing r different keys corresponding to each object. It is inserted under the r different nodes. The nodes that publish the object compute just the different r keys and then insert the obtained objects in the corresponding nodes [13]. There are several approaches to compute these keys. In this section, we present the main methods:

4.2.1 Multiple hash function

In multiple hash function, the object is hashed with r hash functions to obtain r different keys. Each copy is then inserted into the node responsible for that key [14]. To search the object, a node can choose from r locations, each associated with a different key. The node can choose the closest object to reduce search time.

4.2.2 Correlated hash function

A second proposed method uses a single hash function, which assigns a correlated index to each instance. The object is placed in the nodes corresponding to the hash function [15].

The replication based on multiple keys provides better fault tolerance and rapid location of object. However, it should be noted that the number of keys assigned to each node increases with r factor. This method has been applied to CAN protocol, but it can be applied to any DHT system.

4.2.3 Advantages and drawbacks

In multiple hash function and correlated hash function, the location of different replicas is simple and immediate, since it is sufficient to compute the different IDs to find and access the replicas. However, unlike sequential neighboring replication, the maintenance of replicas is very complex since a route node does not know the location of the other replicas. Therefore, if the failure rate in the network is high, it is likely that node storing multiple copies of the same object fail, making the object inaccessible.

Also, these methods can be applied to any DHT. However, the nodes must be aware of the key computational technique used to determine where and how to look for this object.

4.3 Path replication

In path replication, when a new object is inserted, it's automatically copied to the search path location. Thus, the object is replicated to all nodes in the route between the node that has been inserted and the nodes in the route path [16].

Path replication technique copies the replica object in the path before the final destination, and thus decreasing the number of hop used to find object. However, this method does not guarantee the load balancing as search queries will all lead near the root node.

4.4 A comparative study

After a brief description of the different replication techniques used to avoid the churn problem, we can argue that:

Sequential neighboring and Path replication methods are transparent and the protocol of publication is implicitly embedded in the DHT routing. These techniques are applied to specific DHT systems. However, multiple key replication methods are applied to all DHT, but nodes must be informed of the technique used to find objects.

In sequential neighboring methods, maintenance is carried out by the root node, which knows the list of its neighbors. On the other hand, in multiple hash functions, maintenance is assured by replicating nodes. In correlated hash functions, maintenance is carried by the root node, which computes the different key attributed to objects. Finally, in path replication method, the different nodes in the path assure the maintenance.

Concerning the cost of the different techniques, we can argue that there is no additional cost in sequential neighboring methods since the list of successors and predecessors is preconfigured. However, in multiple key replication methods, the replicating node must locate and update replicas. In path replication methods, replicating node must record the path [17].

However, these approaches do not have the same advantages and drawbacks. Each approach is generally applied for a specific DHT. Only the multiple key replication are applied to different DHT systems.

In order to avoid churn problem and support the availability and reduce the costs we exploit a novel approach called zone replication.

5 Zone Replication technique

In structured p2p networks, participating peers can join or leave the system at arbitrary times. However, such intensity of dynamism has a significant effect on the performance of such networks, resulting in stale routing information in the routing table and inconsistency of the stored resource items.

To avoid these problems, DHTs systems use replication techniques. However, each technique is applied to a specific DHT. In this section, we propose a new replication technique called zone replication that can be implemented on top of any DHT. Our goal is to ensure the availability of objects in the overlay even at a very high level of dynamism.

Figure 1 portrays the high level view of a zone replication methodology. We mainly propose three steps to ensure the availability of objects: Publishing, Searching, and Maintenance protocols.

The publishing protocol is responsible for replicating and publishing objects among created zones. How to search those objects is handled by the searching protocol. Finally, the maintenance protocol is responsible for maintaining the replication degree and the consistency of the routing table.

5.1 Publishing protocol

The main idea behind zone replication is that the space identifier is divided into independent zones and each identifier in the system should be associated with r other identifiers. The publishing protocol is presented as follows:

First, divide the identifier space into z independent zones. A zone is specified by L higher order bits of the identifier space that is common among all peers, so the number of zone = 2^L. For each zone, the first L bits are fixed and the following are different. The pseudo code of zone creation is depicted in Algorithm.1. Second, associate with each object in the system r identifiers. In each zone, there must be in minimum one replica. Finally, compute the r identifiers of each object from the first computed identifier (ID1) that correspond to the key of the first copy of the object. ID_1=hash(object). The pseudo code of publishing protocol is presented in Algorithm.2.

```
/* Initiate the list of the identifier space */
$K = hash(Node_i)$
/* Initiate the prefix length */
L=n
/* Initiate the number of zones */
Z=$2^L$
/* Initiate the first prefix */
$prefix\_z_1$
/* Find the prefix of the next zones
For i = 2 to Z do
        $prefix\_z_i = prefix\_z_{i-1} + 1$
End For
/* Associate node to a specific zone */
For i = 1 to K do
        /* Compute the prefix of the node */
        $prefix\_k_i = Extract\_L\_bits(k_i)$
        /* Find the corresponding zone */
        For j =1 to Z do
                If ($prefix\_k_i = prefix\_z_j$) Then
                /* Assign node to the corresponding zone */
                        $List\_noeud\_zone_i\ (K_i)$
                End If
        End For
End For
```

Algorithm 1. Zone creation process

```
/* Compute the first identifier */
$ID_1 = hash(object)$
/* Compute the R identifiers, $ID_2, ID_3, \dots, ID_{r=z}$ */
For i = 2 to Z do
    /* Compute the prefix of the next zone */
    $prefix\_z_i = Extract\_L\_bits(ID_{i-1}) + 1$
    /* Find the list of nodes in this zone */
    For j =1 to N do
        If ($Extract\_L\_bits\ (N_j) = prefix\_z_i$) then
                /* Save the list of nodes */
                $List\_noeud\_zone_i\ (N_j)$
        End If
    End For
    /* Compute the distance between $ID_{i-1}$ and $N_j$ */
        $\delta(ID_{i-1}, N_j)$
    /* Find $ID_i$ */
        $ID_i = Min\ \delta(ID_{i-1}; N_j)$
End For
```

Algorithm 2. Publishing Algorithm

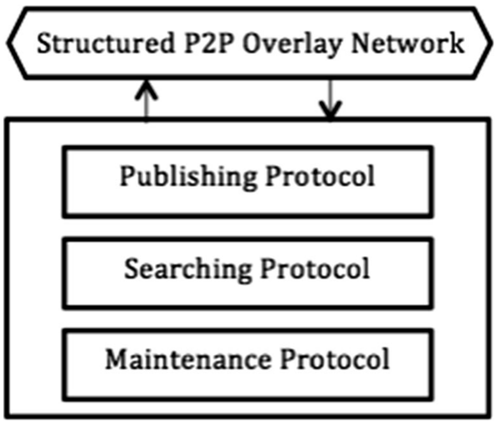


Fig. 1 Zone Replication Methodology

5.2 Searching protocol

```
/* Compute the prefix of the nearest zone to the node (S1) Initiated the search query */
$prefix\_proche\_zoneS_1 = Extract\_L\_bits(S_1) + 1$
/* Find the list of nodes in this zone */
For i =1 to N do
    If ($Extract\_L\_bits\ (N_i) = prefix\_proche\_zoneS_1$) then
        /* Save the list of nodes */
        $List\_noeud\_proche\_zoneS_1\ (N_i)$
    End If
End For
/* Compute the R=Z identifiers of the requested object, $ID_1, ID_2, \ldots, ID_{r=z}$ */
/* Compute the first ID */
    $ID_1 = hash(object)$
/* Compute $ID_2, ID_3, \ldots, ID_{r=z}$ */
    For i = 2 to Z do
        /* Compute the prefix of the next zone */
        $prefix\_z_i = Extract\_L\_bits(ID_{i-1}) + 1$
        /* Find the list of nodes in this zone */
        For j =1 to N do
            If ($Extract\_L\_bits\ (N_j) = prefix\_z_i$) then
                /* Save the list of nodes */
                $List\_noeud\_zone_i\ (N_j)$
            End If
        End For
        /* Compute the distance between $ID_{i-1}$and $N_j$ */
            $\delta(ID_{i-1}, N_j)$
        /* Find $ID_i$ */
            $ID_i = Min\ \delta(ID_{i-1}; N_j)$
    End For
/* Find the IDi corresponding to the nearest zone */
For i = 1 to Z do
    If ($Extract\_L\_bits\ (ID_i) = prefix\_proche\_zoneS_1$) then
        $ID_{rech} = ID_i$
    End If
End For
        $Lookup\ (ID_{rech}\ ; List\_noeud\_proche\_zoneS_1)$
```

Algorithm 3. Searching Algorithm

Figure 2 illustrates the functioning of publishing protocol. The object is replicated into the different created zones which correspond to the identifier obtained by the used hash function according to the publishing protocol. Indeed, the fact that we create r replicas of each object and publish each of them in different zones reduces the search time. Thus, the search space is divided into z independents zone.

The searching process can be done on the closest zones to the node that initiates the searching request in order to reduce the number of required hops and the number of messages. If the desired object is not found in the first zone, the searching process continues in the second and so on till reaching the last if this is needed. Figure 3 gives an example of searching protocol.

To find an object, the request must contain the desired key and the corresponding zone that defines the search space. The limit of the search space for a node n_i is the zone z_{i+1}. Each node is responsible of sending the query to the corresponding zone. The node S1 initiates the lookup of the desired object by determining the nearest zone to his key. After that, node S1 computes the identifier of the object Id that corresponds to the nearest zone. Finally, if after n steps the request fails, the lookup process continues in the next zone. The pseudo code of searching protocol is presented in Algorithm.3.

5.3 Maintenance protocol

Zone replication technique enables each node in the system to find the different replicas identifiers associated to each object and maintain them.

Each node, which disposes one replica, can access or compute the other replicas. Thus, it can monitor the replication degree and the availability of the object periodically.

To maintain the replication degree, it is essential to consider two events: the leave of a node and the arrival of a new one. First, the different identifiers of an object are computed in a

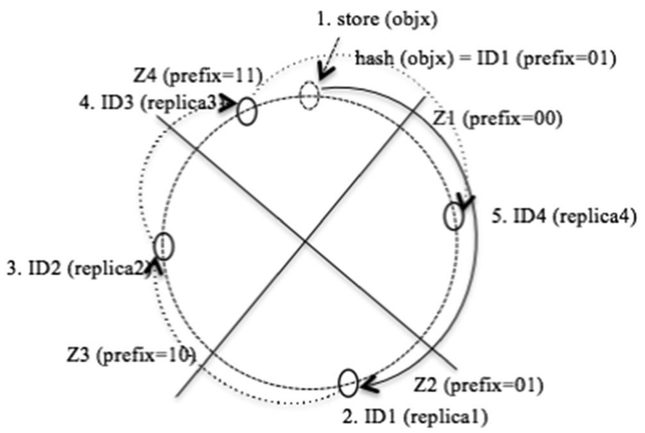


Fig. 2 An example of publishing protocol with 4 zones using chord protocol

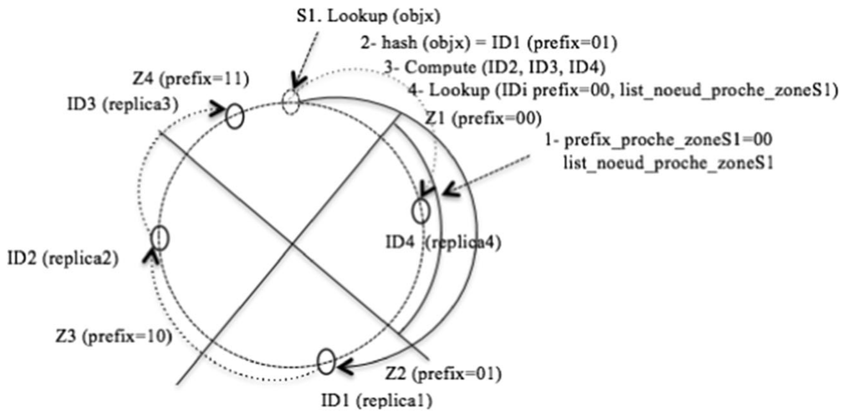


Fig. 3 An example of searching protocol with 4 zones using chord protocol

manner to find each of these identifiers from any ID_{i-1}. Second, each node disposes of one replica, which can access to the other replicas and verify the replication degree. For each object with an identifier ID_i, the replicated node computes the identifier ID_{i+1} corresponding to the next replica associated with the next zone. At regular intervals, the node checks if the responsible node for the identifier ID_{i+1} disposes the object and inserts if necessary. Thus, the node, which disposes the first replica, is responsible for the second; the second is responsible for the third and the last for the first.

5.4 Conclusion

Zone replication technique is a general replica placement method that can be implemented on top of any DHT system.

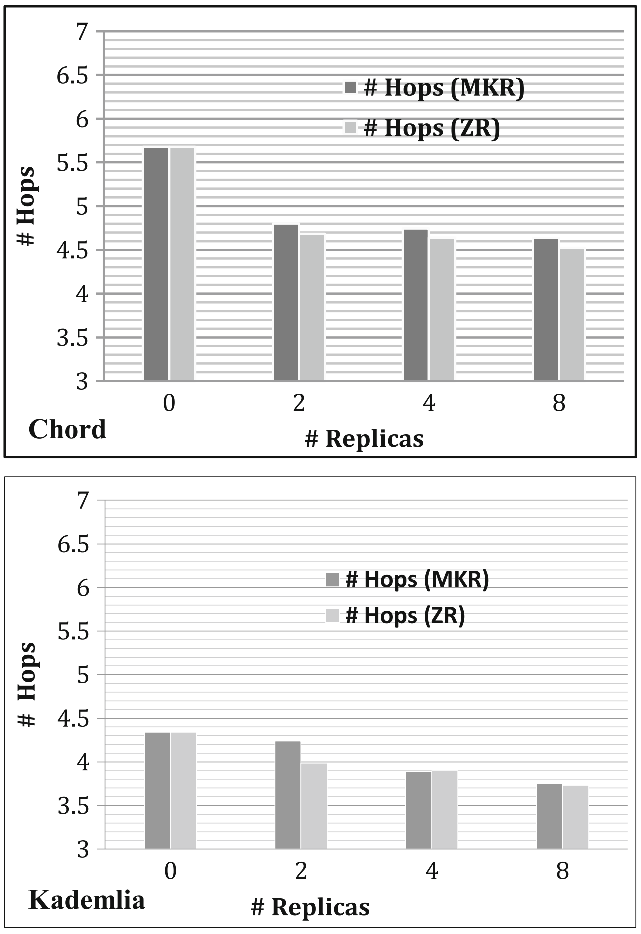


Fig. 4 Effects of the number of replicas on the number of hops

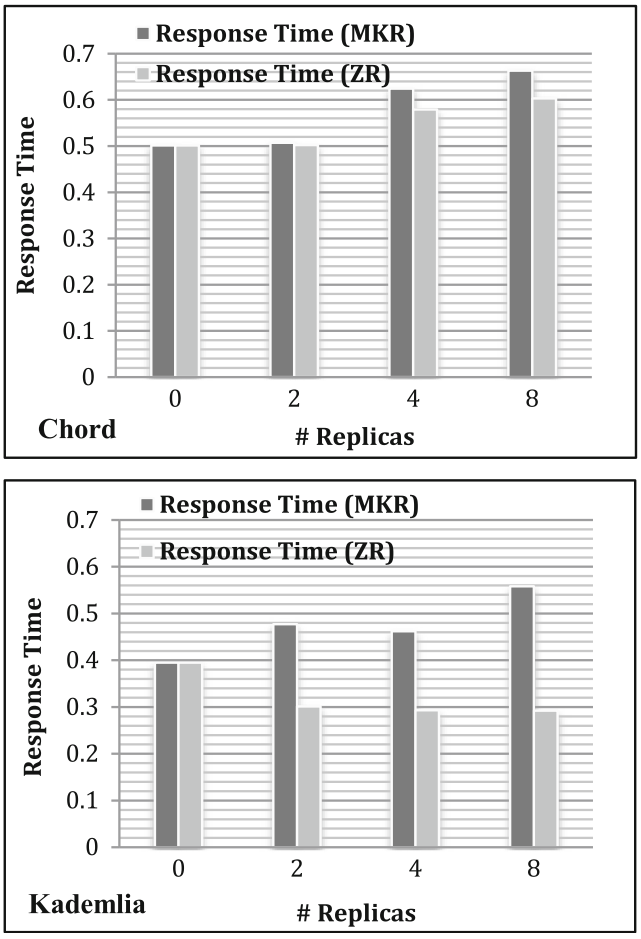


Fig. 5 Effects of the number of replicas on the response time

This has several advantages. It can be used to enhance security by ensuring that the majority of the results match. It can reduce the number of hops needed to find object. Also, it can facilitate the restoring replication degree in presence of a high rate of leave and join nodes.

6 Evaluation

To evaluate zone replication technique in different DHT systems and under the same parameters, we have chosen to implement the different protocols with two different overlays such as Chord and Kademlia. Besides, we have chosen to compare this method to the multiple key replication technique (MKR) since this method is applied to different DHT systems and to assess the effectiveness of our zone replication technique (ZR).

We performed several experiments on PeerfactSim.Kom [18] simulator. Each of which was repeated and tested 10 times with different random seeds to ensure effectiveness. We have integrated also the different protocols to the existing implementation of Chord and Kademlia. We evaluated the different performance metrics by varying two parameters: the number of replicas and the churn rate in order to fix the number of replica and evaluate the effectiveness of the proposed mechanism.

We simulated a network of fixed size (1000 nodes). In the first interval time, all nodes join the network and publish their data. Once the network is stable, nodes are inserted and left the network following an exponential distribution with mean variable churn rate (0 every 20 min, 0.2 every 20 min, 0.3 every 20 min, 0.4 every 20 min, 0.5 every 20 min). Besides, nodes send a message to a randomly chosen node every 60 s. The time out and the number of attempts for each request are set at 5 and 3 min respectively. Finally, the maintenance of replicas is performed every 10 min according to the maintenance protocol defined above.

The settings for all experiments are the following. In Chord, the number of successors and the number of fingers are set to 10. In Kademlia, the bucket size is set to 10 and the degree of parallelism α=3. We evaluate Chord and Kademlia based on the following metrics: the number of hops, the response time, the number of messages sent per second and the success rate.

The number of hops is the common used metric for evaluating the performance of p2p overlays. It presents the number of contacted peers on the way from the source to the destination for an observed query (e.g., lookup in structured overlays) message.

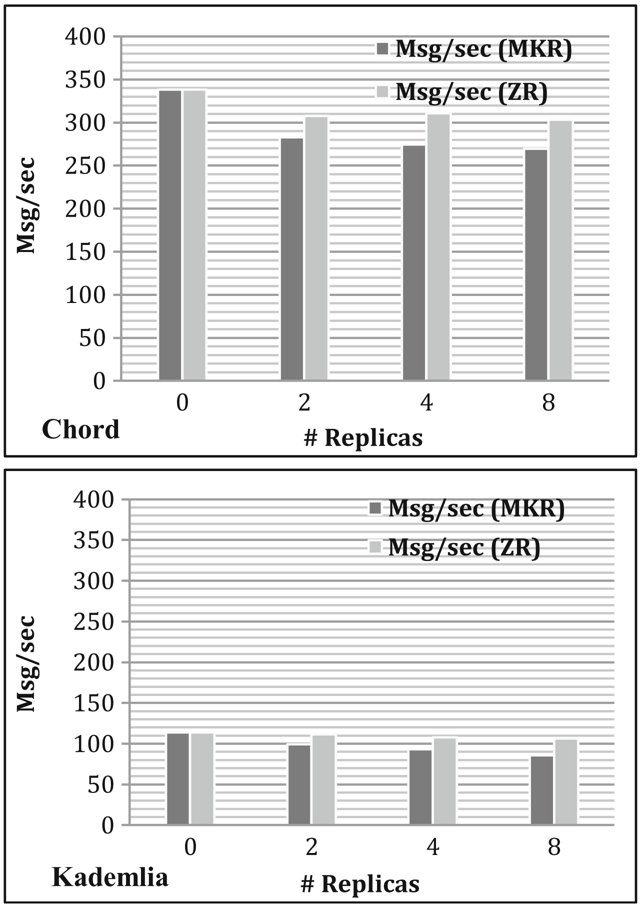


Fig. 6 Effects of the number of replicas on the number of Message/sec

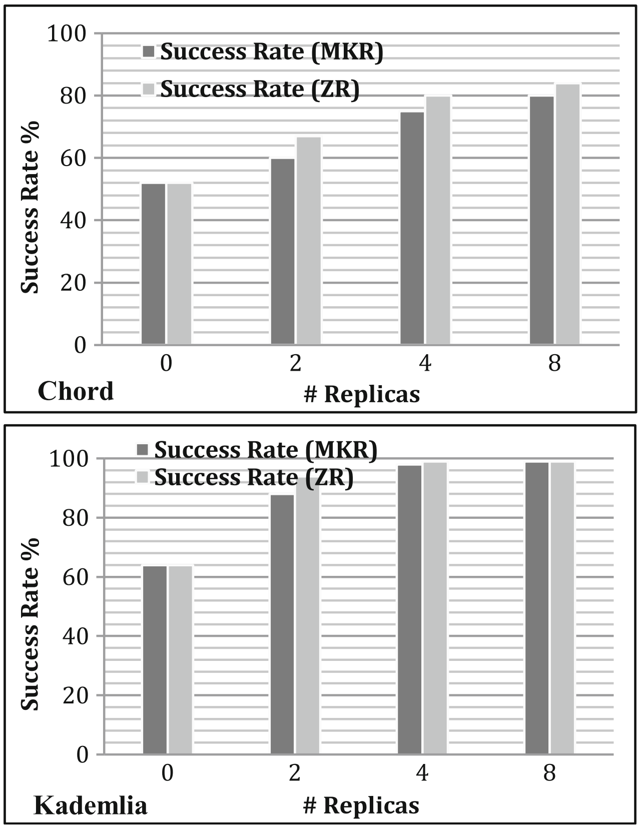


Fig. 7 Effects of the number of replicas on the Success Rate

The response time is defined as the duration of a query operation. It is different in iterative and recursive routing even if the number of hops is the same. The parallelization of look-up queries like in Kademlia brings significant performance benefits, which is evidently reflecting on this metric.

The number of message per second is defined as the total number of messages sent per second for different peers present in the overlay.

Overall success rate is defined as the share of successfully answered query operations. Therefore, the metrics catalogue for the evaluation has to include the average success rate of requests defined as ratio of number of successfully resolved and overall number of query operations.

6.1 Effects of the number of replicas

As a first step, we vary the number of replicas and we fix the churn rate (0.5 every 20 min). The increase in the degree of replication can provide better robustness to churn. However, this requires more memory and signaling for storage and maintenance of replicas. For our evaluation, we set the number of replicas 0, 2, 4 and 8 respectively.

Firstly, we evaluate the effect of n replicas on the number of hops. From Fig. 4, we can notice that the number of hops decrease slightly with the number of replicas in both replication techniques. We can observe that the effect of the number of replicas in Kademlia protocol is much better than Chord protocol. This is due to the fact that Kademlia uses parallel research on all nodes in the k buckets, which reduce the search paths. Also, we notice that the zone replication technique improves a trifle the number of hops relative to the multiple key replication technique in both Chord and Kademlia protocol. This is due to the fact that in multiple key replication technique, to locate an object, a node must be aware of the key computational technique used to determine where and how to look for this object. However, in zone replication technique, a node initiates the lookup of the desired object by determining the nearest zone to his key. After that, the node computes the identifier of the object ID that corresponds to the nearest zone.

Figure 5 shows the effect of the number of replicas on the response time with respect to the churn rate 0.5 every 20 min. We can notice that the response time increases in both protocol Chord and Kademlia when we use a multiple key replication technique. This is caused by replacement of nodes at different position due to the high churn rate, which affects the search

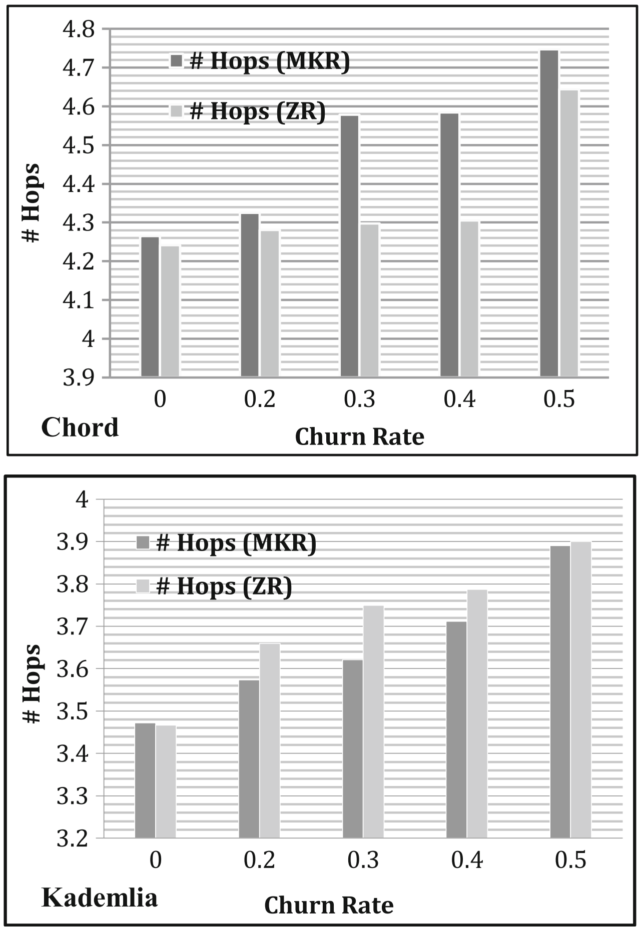


Fig. 8 Effects of the Churn rate on the number of hops

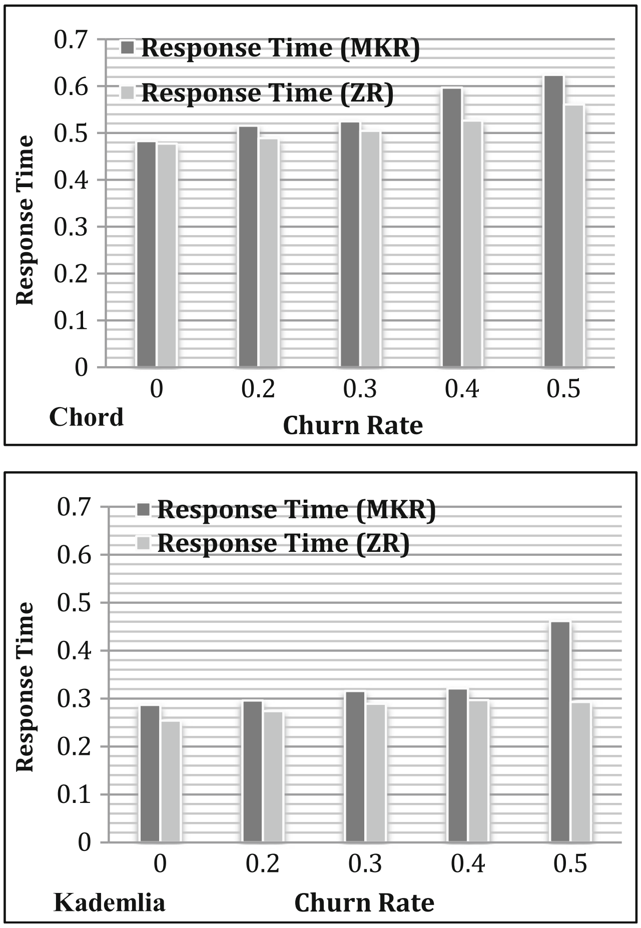
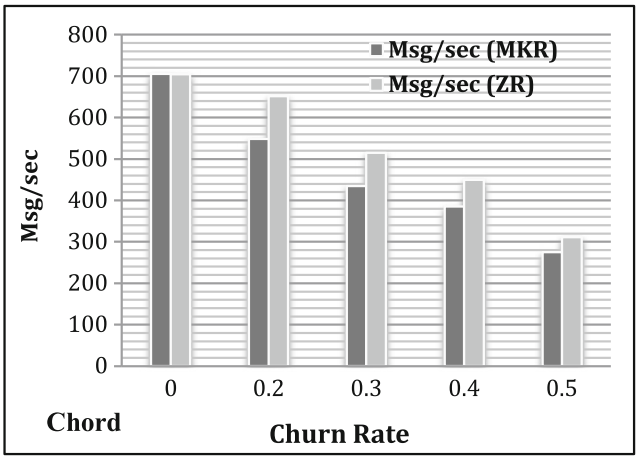


Fig. 9 Effects of the Churn rate on the response time

process. However, when we use zone replication technique, the response time increases slightly in Chord case and decreases slightly in Kademlia case. It increases slightly in chord protocol due to the fact that to maintain the structure of the overlay, when a node leaves, chord run periodically the maintenance process in order to update the links to sequential neighbors and neighboring fingers in the routing table. This verification of fingers is similar to the construction of the routing table, which is not sufficient to guarantee a good performance especially in terms of the response time. Although, it decreases in Kademlia since such protocol uses parallel research in k buckets nodes (k=10 and α=3). Also, the maintenance protocol in zone replication technique stabilizes the number of replicas even in a high churn rate, which makes the replicas always available.

For the mean number of messages sent per second, the zone replication technique has slightly higher values than multiple key replication technique in both Chord and Kademlia protocols. The trends with the mean number of messages as depicted in Fig. 6 decrease slightly in both replication techniques as the number of replicas increases. This is because the high churn rate, which decreases the number of nodes in the overlay. Thus, each protocol initiates the maintenance protocol to update the routing table and maintain the number of replicas. The maintenance protocol in zone replication technique needs much more messages to maintain the different information presented above.

Figure 7 presents the effect of the number of replicas on the success rate. From Chord protocol, we can notice that the success rate increases with the number of replicas. This is due to the fact that when the degree of replication increases, the number of source for the same object therefore increases. Also, we can see that Chord protocol has a relatively low



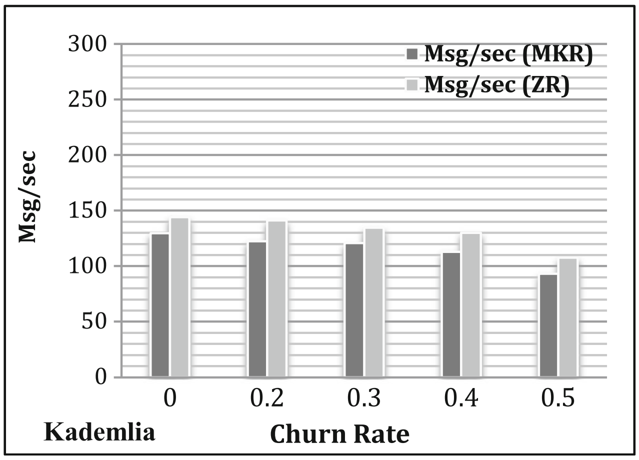
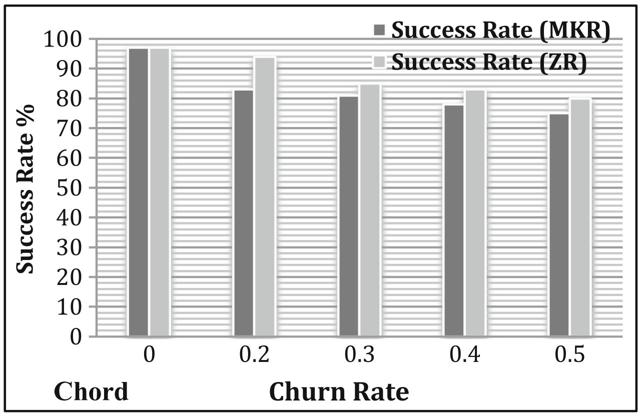


Fig. 10 Effects of the Churn rate on the number of Message/sec



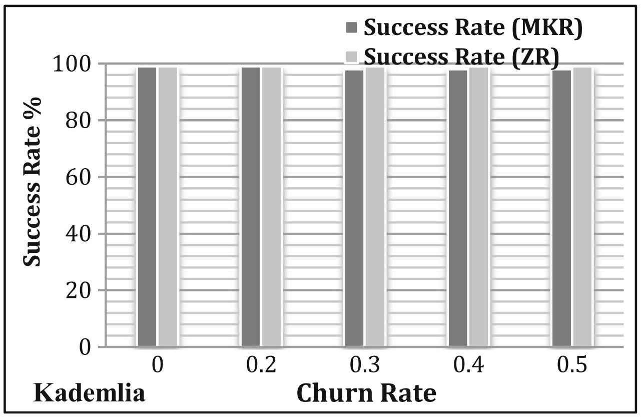


Fig. 11 Effects of the Churn rate on the Success Rate

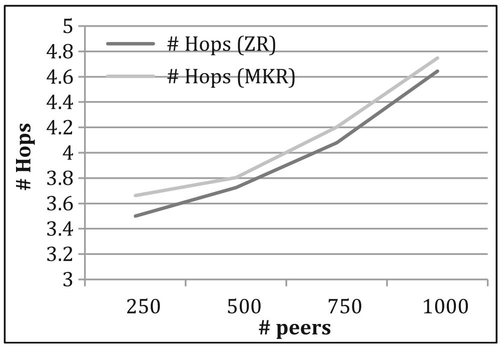


Fig. 12 Evolution of the number of hops

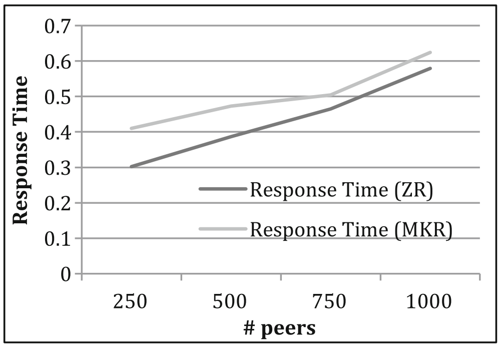


Fig. 13 Evolution of the response time

success rate that reaches 80 % when we use 4 replicas, while in the other case; Kademlia has a significant success rate that reaches 99 % when we use the same number of replicas. From Kademlia, we can also see that the parallel lookup and replication approach used in Kademlia protocol achieves a significant performance improvement. This increase is more visible in the zone replication technique since the maintenance protocol of the different replicas ensures that each query locates the object in the nearest zone, which is often before the root node.and Kademlia based on the following metrics: the number of hops, the response time, the number of request, the number of messages sent per second and the success rate.

6.2 Effects of the churn rate

After that, we chose to study the effect of churn on the performance of Chord and Kademlia protocol under the same number of replicas. The increase in the churn rate can affect the performance of the DHT. However, the replication techniques reduce these effects. Thus, we present in this section the different results obtained in both replication techniques (Multiple key replication and Zone replication). For our evaluation, we set the number of replicas at 4 and the churn rate is set at 0 churn, 0.2/0.3/0.4/0.5 every 20 min respectively.

In Fig. 8, the number of hops increases with different churn rate. The numbers of nodes present impacts on the number of hops. As the overlay size decreases, the number of hops increases. Besides, we can notice that zone replication technique

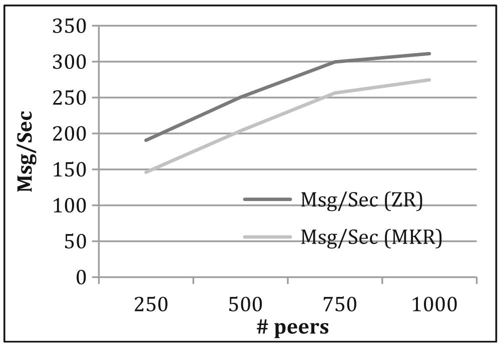


Fig. 14 Evolution of the number of message

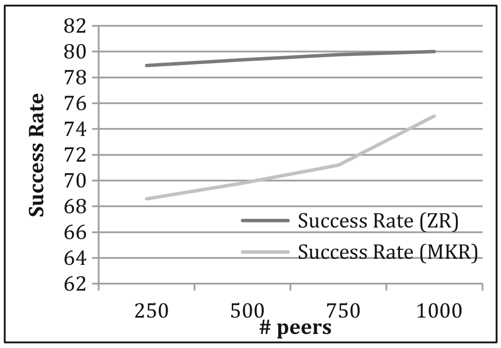


Fig. 15 Evolution of the success rate

produces a smaller number of hops than multiple key replication technique in Chord protocol. However, it produces a higher number of hops in Kademlia protocol. This is due to the structure of such overlays. Chord is based on ring structure and Kademlia is based on tree structure. The structure of the overlay is undamaged by the high rate of churn which make the maintenance of such a structure a very difficult task.

From Fig. 9, we can see that the churn rate affects also the response time in both overlays. We notice that Kademlia response time is much better than Chord because this protocol is fault tolerant by guaranteeing multiple paths for a single replica. Chord topology is more compact and that is why the response time and numbers of hops are a bit low.

An overview on the number of messages sent by second is depicted in Fig. 10. The effects of the node departures are clearly visible in the results. As we can notice, the number of messages sent per second decreases with the churn rate. Also, we notice that the zone replication uses much more message. This results from the fact that such technique updates the routing table and maintains the number of replicas via the maintenance protocol, which needs more messages to maintain this information. The maintenance protocol in zone replication technique needs much more messages to maintain the different information presented above.

In a stable network and for the two replications techniques, more than 97 % of queries succeeded (Fig. 11). However, when the churn rate increases, the success rate in Chord protocol drops significantly to reach 75 % when we use the multiple key replication technique and 80 % when we use the zone replication technique. We can also notice that the success ratio in Kademlia protocol is very stable (99 %) even under a high churn rate. This is notably due to the parallel lookup (α=3) and the size of the k buckets (k=10).

6.3 Performance of the proposed mechanism

Finally, we choose to present the behaviours of the proposed mechanism under chord protocol and compare our replication technique to multiple key replication mechanism. To achieve this, we fix the number of replica to 4 and we vary the network size (250, 500, 700 and 1000 peers). In the first interval time peers join the network and publish 10 objects. After a stabilization period, nodes are inserted and left the network following an exponential distribution with mean variable churn rate 0.5 each 20 min. Besides, nodes send a message to a randomly chosen node every 60 s. Finally, the maintenance of replicas is performed every 10 min.

We present next the performance of the chord protocol based on the number of hops, the response time, the number of message sent per second and the success rate.

We notice from Figs. 12, 13, 14 and 15 that the zone replication technique gives better results in terms of number of hops, response time and success rate. However, the number of messages sent per second is higher than the other one.

Dividing the search space reduces both the number of hops and the response time. The total gain is may be not very substantial, but it is costless. Our proposed method produces shorter routes and better resistance to churn. The success rate exceeds 80 % even in presence of a high churn rate. However, the maintenance of a higher number of replicas increases the total number of exchanged messages on the network. Consequently, the use of the zone replication technique improves the resistance of the system against the churn phenomenon.

7 Conclusion

Structured p2p networks are considered as efficient and highly resilient platform for large-scale distributed applications. One of the challenges of such systems is how these systems behave during users churn and how to deal with. Unlike other distributed systems where failures may be considered rare or abnormal, most p2p networks constantly remain in the state of churn and embrace frequent failures as a part of their normal operation. But under a high churn rate most overlays fail to remain in the stable state. To overcome this problem, researchers proposed different replication techniques to ensure the availability of objects since the main objective of nodes is to find them.

While many metrics in such systems (e.g., number of hops, response time, message overhead, success rate) affect their usefulness to the user. One of commonly studied problem in the literature is the ability of such systems to ensure the availability of objects in case of failures. It may be argued that compromised connectivity is one of the most fundamental by-products of churn that affects directly routing efficiency and other metrics observed by the user.

In this paper, we studied first the existing replication techniques that deal with churn problem, and then we proposed a new one called "zone replication technique" applied to all DHTs structure. The conducted simulations have shown some advantages of zone replication technique compared to the existing one. Several investigations are under study in order to improve and ascertain the viability of our proposed technique.

References

1. Stoica I, Morris R et al (2003) Chord: a scalable peer-to-peer lookup protocol for internet applications. IEEE/ACM Trans Net 11(1):17–32
2. Maymounkov P, Mazieres D (2002) Kademlia: a peer-to-peer information system based on the XOR metric. Proc. IPTPS, Cambridge, pp 53–65
3. Liu H, Liu X, Song W, Wen W (2011) "An age-based membership protocol against strong churn in unstructured p2p networks". In: Proc. of International Conference on Network Computing and Information Security, vol. 2. Guilin, China, pp 195–200
4. Abraham I, Awerbuch B, Azar Y, Bartal Y, Malkhi D, Pavlov E (2003) a generic scheme for building overlay networks in adversarial scenarios. In: Proc. 17th Int. Symp. on Parallel and Distributed Processing (IPDPS)
5. Loguinov D, Kumar A, Rai V, Ganesh S (2003) graph-theoretic analysis of struc- tured peer-to-peer systems: routing distances and fault resilience. ACM SIG- COMM
6. Li J, Stribling J, Gil T, Morris R, Kaashoek F (2004) Comparing the performance of distributed hash tables under churn. IPTPS
7. Lam S, Liu H (2004) Failure recovery for structured P2P networks: protocol design and performance evaluation. ACM SIGMETRICS/ Performance'04
8. Liu Z, Yuan R, Li Z, Li H, Chen C (2006) "Survive under high churn in structured p2p systems: evaluation and strategy". In Proceedings of ICCS 2006.
9. Kuhn F, Schmid S, Wattenhofer R (2005) A self-repairing peer-to-peer system resilient to dynamic adversarial churn. In: Proc. 4th Int. Workshop on Peer-to-Peer Systems (IPTPS)
10. Leslie M, Davies J, Huffman T (2006) A comparison of replication strategies for reliable decentralised storage. J Netw 1(6):36–44
11. Shafaat TM, Ahmad B, Haridi S (2012) ID-replication for structured peer-to-peer systems. In: Kaklamanis C, Papatheodorou T, Spirakis PG (eds) Euro-Par 2012. LNCS, vol 7484. Springer, Heidelberg, pp 364–376
12. Knezevic P, Wombacher A, Risse T (2005) Enabling high data availability in a DHT. In: 2nd International Workshop on Grid and Peer-to-peer Computing Impacts on Large Scale Heterogeneous Distributed Database Systems
13. Ghodsi A, Alima L O, Haridi S (2005) "Symmetric replication for structured peer-to-peer systems". 3rd Intl. Workshop on Databases, Information Systems and P2P Computing
14. Lv Q, Cao P, Cohen E, Li K, Shenker S (2002) Search and Replication in Unstructured Peer-to-Peer Networks. In: Proceedings of the 16th annual ACM International Conference on supercomputing
15. Sit E, Haeberlen A, Dabek F, Chun BG, Weatherspoon H (2007) Proactive replication for data durability. In Proceedings of the 2007 ACM CoNEXT Conference, New York
16. Kim B J, Yoon C N, Han S K, Jeong H (2002) "Path finding strategies in scale-free networks". Phys Rev E 65
17. Ktari S, Zoubert M, Hecker A, Labiod H (2007) Performance evaluation of replication strategies in DHTs under churn. In: Proceedings of the Sixth International Conference on Mobile and Ubiquitous Multimedia, MUM'07, pp 90–97
18. Graffi K (2011) "PeerfactSim.KOM – a peer-to-peer system simulator: experiences and lessons learned". In: Proc. of IEEE International Conference on Peer-to-Peer Computing (IEEE P2P'11)

Zied Trifa received his master degree of computer science from the University of Economics and management of Sfax, Tunisia in 2010. He is currently PhD student in computer science at the same University. His research interests include P2P Computing, distributed systems, and performance evaluation.

Maher Khemakhem received his Master of Science, his Ph.D. and Habilitation accreditation degrees, respectively, from the University of Paris 11 (Orsay), France in 1984, 1987 and the University of Sfax, Tunisia in 2008. He is currently Associate Professor of Computer Science Department at the Faculty of Computing and Information Technology at King Abdulaziz University, Jeddah, Saudi Arabia. His research interests include distributed systems, performance analysis, Networks security and large scale OCR.