International Conference on Automation, Control, Engineering and Computer Science (ACECS'14)
Proceedings - Copyright IPCO-2014, pp.164-170
ISSN 2356-5608

Effects of Churn on Structured P2P Overlay Networks

Zied Trifa[a], Maher Khemakhem[b]
[a]Department of Computer Science, University of Sfax, Tunisia
[b]College of Computing and Information Technology, University of King Abdulaziz, Saudi Arabia
[a]trifa.zied@gmail.com, [b]maher.khemakhem@fsegs.rnu.tn

***Abstract*— As structured p2p systems becomes largely deployed, the size of the network becomes large enough that any kind of churn has a significant influence on the performance of such overlay. In structured p2p systems, peers create interactive link with other peers called neighbours that both depend on and influence each other. While peers trust each other's to delivers services, the network effects exerted from peer to peer interactions remain virtually unexplored. This paper focuses on peer churn, where a significant number of peers join and leave the overlay frequently in a short time interval. The effects of such problem remain widely ignored yet may play a vital role in protection of such overlay. The direction and strength of this effect are evaluated experimentally on Chord and Kademlia protocol by measurement the set of metrics that describe the performance attributes such as efficiency, stability and scalability.**

Keywords*— structured P2P overlay networks; churn; measurement; performance.

I. INTRODUCTION

Structured p2p systems, based on distributed hash table (DHT) such as Chord [1] and Kademlia [2], have the potential to harness huge amount of resource shared by peers having high connectivity. Unfortunately, it has been shown that most of these DHT systems suffer from a large fraction of churner peers, who refers to the effect of independent join and leave of a large fraction of peers in short interval time.

The problem of churn, or the independent arrival and departure of participating peer, is a problem facing almost p2p networks. This phenomena is a major source of breaking down the overlay structure, because it is generally much more difficult to notify neighbours to restructure the overlay topology. Thus, causing stale neighbour pointers or data lose. Each instance of join or leave a network causes the protocol to rebalance its keys and the data among the present peers, which generate a considerable traffic load and degrades the performance of the system. Therefore, it is important to understand the effects of churn on the performance of such systems characterized by high dynamicity and decentralized nature.

In structured p2p overlay networks, there is no barrier to join or leave the network. Peers can join or leave the network whenever they want. A peer joins the network when a participant starts the application and leaves when the participant exits. Although, such systems have to deal with churn problem. We define churn as the change in the set of nodes in the networks due to joins, friendly leaves or node failures. Join is such a condition that nodes join the network one by one or leave gracefully by informing their neighbour to restructure the overlay topology and save objects. However, a node failure is such a condition that large percent of nodes join or silently leave the network simultaneously and frequently [3]. This failure damages the structure and affects the performance of the overlay.

In this paper we evaluate the impact of churn on the performance of Chord protocol as the most cited structured p2p overlay and Kademlia protocol as the most deployed one. The evaluation of such overlays is commonly done using defined scenarios and a set of metrics such as the number of hops, the response time, the number of requests and the success rate.

The rest of the paper is organized as follows. Section II reviews the related works. In section III, we provide background information on Chord and Kademlia as DHT systems. Also, we introduce the problems of churn. Section IV describes the set of metrics and scenarios used in our measurement. In Section V, we give details analysis about the results of our measurement. Finally, section VI concludes.

II. 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 crashes.

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] examines 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 [3] 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.

Our aim is to deepen study the impact of churn on the performance of structured p2p overlay networks. A careful evaluation will contribute to find basic solution to such problem. Our study is therefore concentrated towards the analysis of churner peers under Chord and Kademlia protocols, which considered as the most cited and deployed protocols.

III. BACKGROUND

A. Distributed Hash Systems

Structured p2p overlay networks are an important and powerful class of overlays that has emerged in recent years. They are typically targeted at peer-to-peer deployments involving user-managed machines on the Internet. Structured overlays such as Chord [1] and Kademlia [2], presents a distributed hash table abstraction on top of a population of networked participants. Each participating node in the overlay has an identities from a large identifier space, and is responsible for handling messages addressed to an extent of the identifier space around its own ID. In order to route messages in the overlay, every node maintains a routing table of "links". The set of nodes and links in the system forms a structured network graph, over which ID lookups can be routed to the responsible node efficiently. When used to store data, structured overlays are often called distributed hash tables (DHTs), though many applications do not require storage.

1) Chord

Chord [1] is the most cited overlay in the field of distributed hash table systems. Chord assigns m-bit identifiers to each node and object using a base hash function. The node identifier is calculated as the hash of the IP address or the cryptography of the public key of the node. It is worth to underline that the identifiers are ordered in a ring topology. The successor of a given key k, is the node characterized by the identifier equal to or immediately subsequent to the identifier of k. Each peer in Chord knows the list of its predecessors and successors, which called neighbours. For a given peer, the main knowledge of this list is not sufficient to guarantee a good performance especially in terms of number of hops needed to route queried and the response time. For this reason, each peer in Chord connects to other neighbouring nodes, called fingers, which constitute its routing table.

In Chord, nodes can join or leave at any time. To join the overlay, the node n must know at least an existing node s in the ring. It initializes first the predecessor and the fingers of node n. The peer n asks s to look them up. After that it update the fingers and the predecessor regarding the addition of n. Finally, once the node n is entered in the system, it is necessary to move responsibility for all the keys for which n is now the successor. When a node wants to find an object, it sends a request to his fingers to locate the appropriate node whose identifier corresponds to the requested object. The node that receives the request runs in turn the same process until the object is reached. To maintain the structure of the overlay, when a node leave, chord run periodically the maintenance process in order to update the links to sequential neighbours and neighbouring fingers in the routing table. Verification of fingers is similar to the construction of the routing table. In addition to minimize failures of queries, each node maintains a list of its successors, which can alternatively be used for routing.

2) Kademlia

Kademlia [2] is the most deployed overlay in the field of distributed hash table systems. The participant nodes form a virtual tree structure. Each node is identified by a 160-bit node ID. This ID serves not only as identification, but the Kademlia protocol uses to locate objects. In fact, the node ID provides a direct map to objects.

Kademlia uses an XOR metric to define distance between nodes and objects. The XOR metric allows extending routing table beyond single bits. Thus, reducing the number of hops needed to locate an object. Kademlia nodes store contact information about each other to route query messages. Every node keeps a list of IP address, UDP port, Node ID triples for nodes of distance between 2^i and 2^{i+1} from itself. We call these lists k-buckets.

A node that would like to join the net must first go through a bootstrap process. In this phase, the joining node needs to know the IP address and port of another node that is already participating in the Kademlia network. The joining node inserts the bootstrap node into one of its k-buckets. It then does a FIND_NODE of its own ID against the bootstrap node. After this, it refreshes all k-buckets. This refresh is just a lookup of a random key that is within that k-bucket range. When searching for some value, the protocol needs to know the associated key and explores the network in several steps. Each step will find nodes that are closer to the key until the contacted node returns the value or no closer nodes are found. When Kademlia node receives any message from another node, it updates the appropriate k-bucket for the sender's node ID. If the sending node already exists in the recipient's k-bucket, the recipient moves it to the tail of the list. If the node is not already in the appropriate k-bucket and the bucket has fewer than k entries, then the recipient just inserts the new sender at the tail of the list. If the appropriate k-bucket is full, however, then the recipient pings the k-bucket's least-recently seen node to decide what to do. If the least-recently seen node fails to respond, it is evicted from the k-bucket and the new sender inserted at the tail. Otherwise, if the least-recently seen node responds, it is moved to the tail of the list, and the new sender's contact is discarded.

B. The problem of churn

Users like to use p2p, because there are few restrictions. They can join and leave the network whenever they want. The independent arrival and departure of peers creates the collective effect called churn [1]. However, such freedom causes unpredictable network environment, which leads to the most complex design challenge of structured p2p protocols.

The effect of joining peers is usually the less problematic aspect of churn, since it mainly results in temporary failures like routing inconsistencies or resources, 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 friendly join/leave and failure join/leave [3].

Friendly join/leave is such a condition that nodes join the network one by one, or leave gracefully by informing their neighbours. However, failure join/leave is such a condition that large percent of nodes join and/or silently leave the network simultaneously and frequently.

There have been very few large-scale DHT based application deployments to date, and so it is hard to derive good requirements on churn-resilience. These systems provide a simple indexing service for locating files on those peer nodes currently connected to the network, a function that can be naturally mapped onto a DHT based mechanism. While some DHT applications might require greater client availability, others may show similar churn rates to file-sharing networks. As such, we believe that DHTs should at least handle churn rates observed in to day's file-sharing networks. There is always a price to churn which may manifest itself as dropped messages, data inconsistency, increased user-experienced latency, or increased bandwidth use.

A pervasive requirement of structured p2p systems is to deal with churn. A high churn rate can increase costs or decrease service quality. For example, a large number of maliciously controlled peers could leave the network simultaneously; the power is cut off over a wide area. Each instance of join and 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. The higher the churn rate is, the more difficult it becomes for the network to maintain its consistency. Too high churn can affect the overlay structure and damage the selection of key design parameters. Also this dynamics can cause routing failures that cause subsequent lookups to return inconsistent results, loss of stored resources or inconsistent views of the peers on the overlay.

Churn has a significant effect on the performance of structured P2P systems, e.g. 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, just to name a few. Consequently, the effects of churn should be taken into account when design or evaluate a structured p2p system.

IV. THE EFFECTS OF CHURN

The nature of structured p2p overlay networks introduces some relevant quality aspects each application has to consider. For example, efficiency, stability and scalability are issues brought by the fact that peers can randomly leave, join, or perform queries. It results in a big variation of network size, number of exchanged messages, number of stale contacts in routing tables, etc. Efficiency is defined as the performance of overlay operations and costs of the services provisioning. Stability describes the behaviour of the system under changed condition friendly join/leave or failure join/leave. Scalability is the quantitative adaptability of the overlay to a changing number of participants.

While many metrics of a system (e.g. number of hops, response time, efficiency of routing, message overhead, file popularity) affect its usefulness to the participant, one commonly problem is the ability of overlay to stay connected and ensure the availability of resources in the face of random failures. It may be explored that compromised connectivity is one of the most fundamental by-products of churn that directly affects routing efficiency and other metrics observed by the user.

In order to make statements on the performance of an overlay under churn, a suitable scenario and the appropriate metrics are needed.

A. *Metrics*

Following set of metrics is relevant to the mentioned performance of structured p2p overlays:

Number of hops is the common used metric for evaluating the performance of peer-to-peer 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. Routing in distributed hash tables (DHTs) can be either recursive or iterative.

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 lookup queries like in Kademlia brings significant performance benefits, which is evidently reflecting on this metric.

Number of request is defined as the number of exchanged request during a simulation, which presents all messages of overlay operations involved in node interaction and all necessary messages for transfer of the data.

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

Overall success rate is important as both the number of hops and the response time cannot show 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.

B. *Scenario*

The scenario defines which overlay operations certain peers or a group of peers perform at which point of time. The scenarios considered in this benchmark set are described in the following:

Ideal is the scenario where peers first join the network and once the bootstrapping process is over, peers start to perform specific overlay operations. A new overlay operation will not take place before an appropriate stabilization phase is over. Churn is not expected. Ideal scenario is depicted in Fig.1.

During the first interval time [0m, 60m] all peers join the network and publish their data. In the next interval time [60m,

90m] the system stabilizes. After that, peers perform a number of overlay operations starting at 90m.

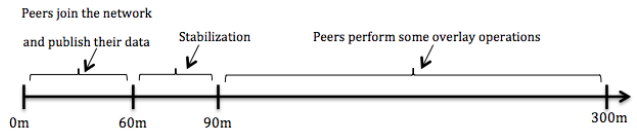

Fig. 1. Ideal scenario

Join/Leave is the scenario where a significant number of peers joins and leaves the overlay at specific time interval. Fig.2 describes the time line of this scenario.

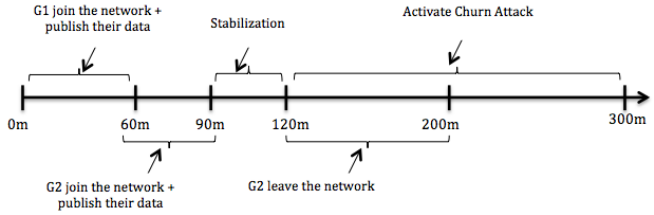

Fig. 2. Join/Leave scenario

Peers are divided in two groups G1 and G2. G1 first join the network and publish their data. Once the bootstrapping process is done, G1 start to perform some overlay operations. Besides G2 join the network and perform the appropriate actions analogue to G1. After an appropriate stabilization phase, G2 leave the network during the interval time [120m, 200m]. We activate the churn at t=120m, in which peers from G1 joins and leaves the network in a short interval time (0,5/20m).

Churn is the scenario where a significant number of peers joins and leaves the overlay in a short time interval. Churn scenario is the same like ideal scenario, with difference that we activated churn at t=90m and each time we vary the churn rate as depicted in Fig.3.

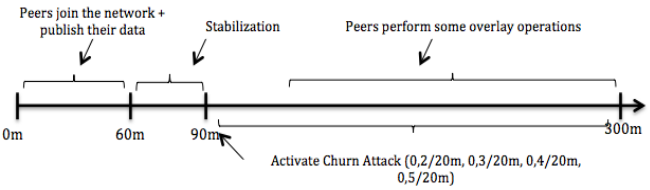

Fig. 3. Churn scenario

V. EVALUATION

We performed several experiments on PeerfactSim.Kom [8] simulator. With the proposed metrics and scenarios we are able to evaluate the effects of churn on the performance of existing structured p2p overlays. Among them we identified Chord [1] (Ring structure) and Kademlia [2] (Tree structure).

We evaluate the effect of churn on the Chord protocol and Kademlia by varying two parameters: the network size and the churn rate. First, we simulate a network of fixed size (Ideal scenario: 1000 nodes). Once the network is stable, nodes send a message to a randomly chosen node every 60 seconds. Second, we simulate a network of variable size 250 to 1000 nodes (Join/Leave scenario).

The settings for all experiments are the following. In Chord, the number of successor is 10 and the number of fingers is set to 10 also. In Kademlia, the bucket size is set to 20 and the degree of parallelism is α=3. We evaluate Chord and Kademlia based on the following metrics: the number of hops, the response time, the number of request, and the number of message sent per second and the success rate.

A. *Effects on Chord*

Fig.4 shows the evolution of the number of hops in both scenarios, Ideal and Join/leave. In the first scenario, we note that the number of hops stabilize during the interval [90m, 300m] to reach 5,8. This is due to the stabilization of the number of peers in the network (1000 nodes) and the number of queries sent by each peer. However, in the second scenario, we note that the number of hops has significant variations during the same interval time. This is due to the variation in the network size. The effect of the peer arrivals and departures are clearly visible in the results. For example, between [90m..120m], the number of hops increase slightly to reach 6,8. After that, we notice a high variation in the number of hops between [200m..300m], to reach 11 since the activation of churn that reach 0,5 each 20 minutes and also the departure of 750 peers.

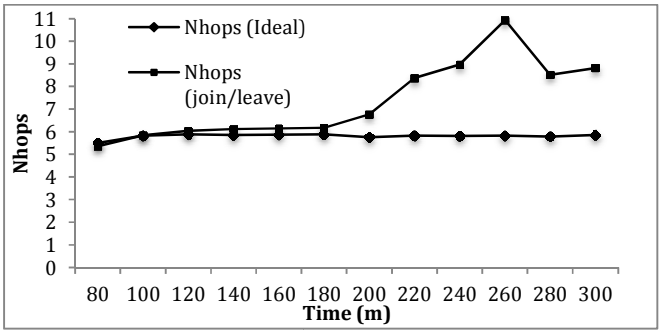

Fig. 4. Evolution of the number of hops (Chord)

Fig.5 presents the evolution of the response time. We note that response time is steady in the first scenario. Therefore, it has a lot of variation in the second one. The stability in the first scenario is produced by the stability in the network size. However, the variation in second scenario is caused by two events, the join/leave of nodes and the high churn rate.

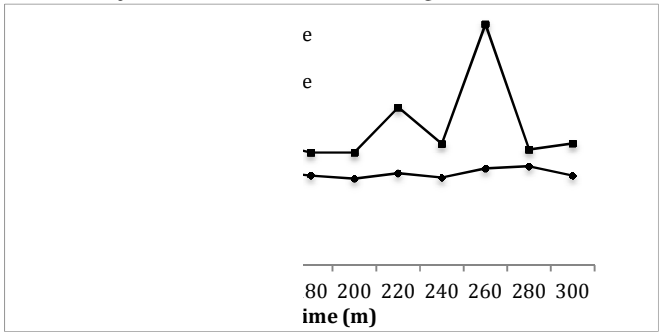

Fig. 5. Evolution of the response time (Chord)

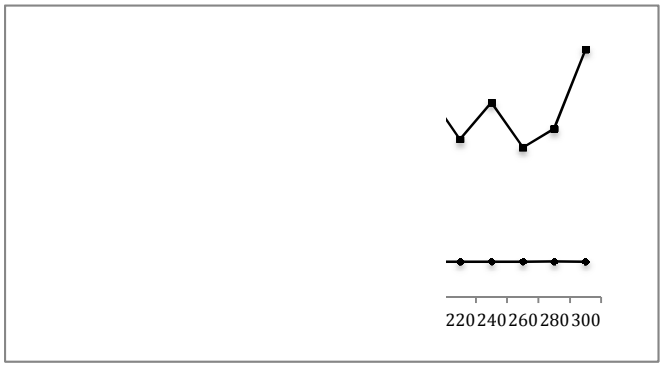

Fig. 6. Evolution of the number of message/sec (Chord)

Fig.6 demonstrates the evolution of the number of messages sent per second. We note that the total number of messages increase in both scenario between [40m..90m]. This is caused by the join process. We notice that this number stabilize in the first scenario since the stabilization of the number of peers and the number of messages sent by each peer. However, the total number of messages increase linearly in the second scenario between [120m..200m] since the evolution of the network size. Also, we notice a high variation between [200m..300m]. This is due to the high churn rate.

Thereafter, we vary the churn parameter. We used a Churn scenario with 1000 nodes. We activated the churn process at t=90m and each time we vary the churn rate (0/20m: ideal, 0,1/20m, 0,2/20m, 0,3/20m, 0,4/20m, 0,5/20m).

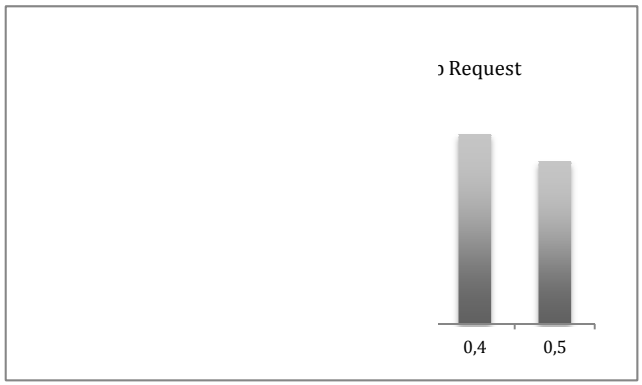

Fig. 7. Evolution of the number of request (Chord)

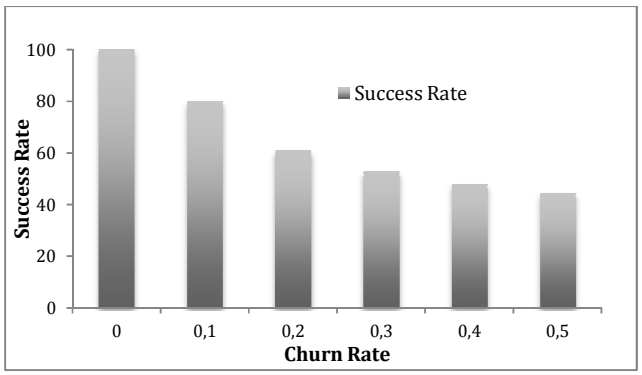

Fig. 8. Success Rate (Chord)

Subsequently, we evaluate the evolution of the total number of requests and the success rate of search queries. Fig. 7 and 8 show the results respectively. We note that the success rate decreases each time we increase the churn rate. For a churn rate equal 0,5/20m, the success rate drops below 45%. Churn also affects the total number of requests, we note that the number of requests decreases. Indeed the number of nodes decreases, it goes from 1000 peers at t=60m to reach 350 at t=299m. This is caused by the high churn rate in the network.

B. Effects on Kademlia

We present now the results of the Kademlia protocol. The bucket size is fixed at 20 and the degree of parallelism is α=3. We use the same parameters and scenarios simulation.

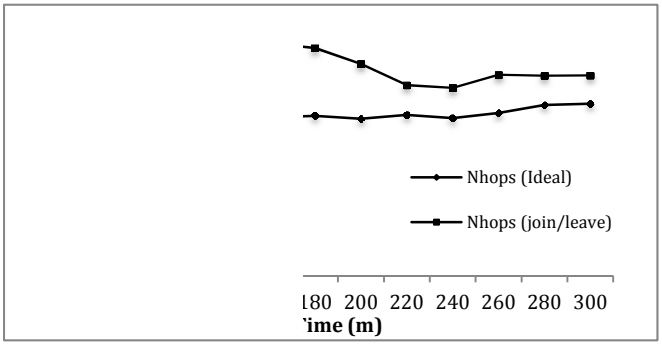

Fig. 9. Evolution of the number of hops (Kademlia)

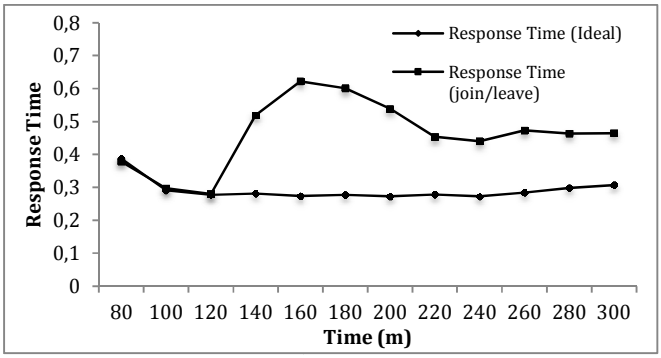

Fig. 10. Evolution of the response time (Kademlia)

Fig.9 and 10 demonstrate the evolution of the number of hops and the evolution of response time in Kademlia protocol. As we can notice that the number of hops and the response time in Kademlia protocol are shorter than Chord protocol. This due to the fact that Kademlia uses parallel research on all nodes in the buckets, which reduces the search paths and thus reduce the response time. Also varying the number of peers in the network and the churn rate has some influence on the performance of such protocol. Figure 9 show that the number of hops stabilize in the first scenario, to reach 4. However, in the second scenario the number of hops increases linearly to reach 5,5. Similarly, the response time stabilize in the first scenario to reach 0,3, but increase in the second to reach 0,6.

Fig.11 presents the evolution of the number of messages sent per second. We note that the total number of messages increase linearly in both scenario between [40m..90m]. This is caused by the joining process. Also we notice that the number of messages decreases at t=90m due to the fact that the system stabilize and during this period of time there is only the stabilization messages. Besides, figure 11 demonstrates that the number of messages stabilizes in both scenarios since Kademlia uses parallel research on all nodes in the buckets. But this number increases in the join/leave scenario to reach 800 messages per second since we activated the churn attack.

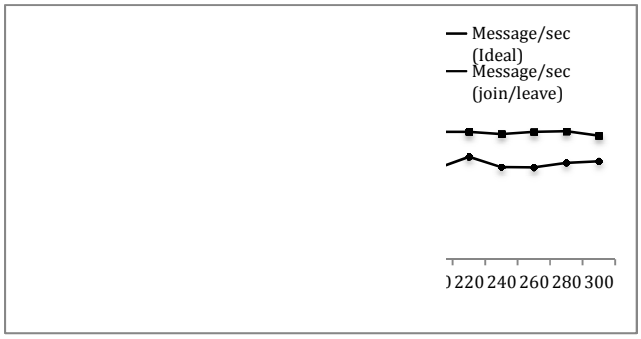

Fig. 11. Evolution of the number of message/sec (Kademlia)

By varying the churn rate, we can notice through the Fig. 12 and 13 that the number of requests and the success rate decrease. As we can notice that the number requests are worst than Chord protocol. However, the success rates are better. This results from the fact, that Kademlia protocol uses parallel research request on all nodes in the k buckets.

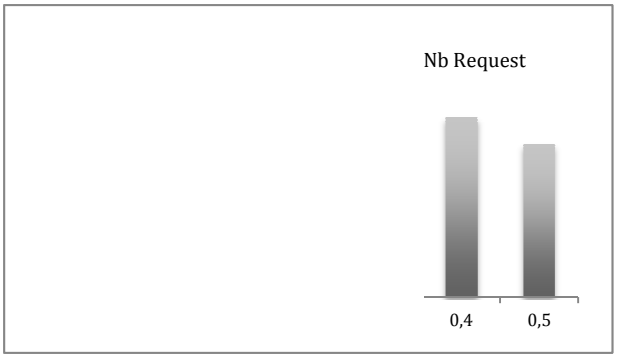

Fig. 12. Evolution of the number of request (Kademlia)

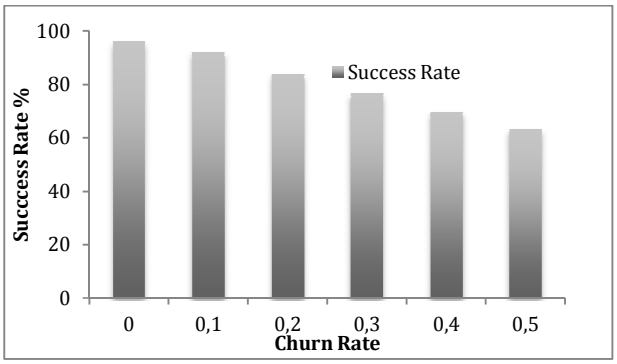

Fig. 13. Success Rate (Kademlia)

VI. CONCLUSION

In this paper, we showed the effects of churn on the performance of structured p2p overlay networks. We presented a measurement study of Chord protocol as the most cited overlay in the field of distributed hash table systems and Kademlia as the most deployed one. The results show the weakness of such overlay to deal with such problem. As we can notice, there are differences between topologies. More precisely, the performance of Kademlia under a high churn rate is much better than Chord because this protocol is fault tolerant by guaranteeing multiple paths for a single destination. Chord topology is more compact and that is way the performance of such overlay is a bit lower.

Also, in such systems, peers are autonoums, they join and leave the network whenever they want. There is no way to make barriers to join the overlay. Thus, the main challenge is to protect the ability of the system to locate every object. In future work, we plan to analyze in detail the replication techniques employed to ensure the availability of objects under high churn rate and develop a new replication technique to improve the performance of such systems.

REFERENCES

[1] I. Stoica, R. Morris et al., Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications, IEEE/ACM Trans. Net, vol. 11, no. 1, 2003, pp. 17–32.

[2] P. Maymounkov and D. Mazieres, “Kademlia: A Peer-to-Peer Information System Based on the XOR Metric,” Proc. IPTPS, Cambridge, MA, USA, Feb. 2002, pp. 53–65.

[3] Z. Liu, R. Yuan, Z. Li, H. Li, and C. Chen, “Survive under high churn in structured p2p systems: evaluation and strategy,” in *Proceedings of ICCS 2006*, 2006.

[4] I.Abraham, B.Awerbuch, Y.Azar, Y.Bartal, D.Malkhi, and E.Pavlov. A Generic Scheme for Building Overlay Networks in Adversarial Scenarios. In Proc. 17th Int. Symp. on Parallel and Distributed Processing (IPDPS) 2003.

[5] D. Loguinov, A. Kumar, V. Rai, S. Ganesh: Graph-Theoretic Analysis of Struc- tured Peer-to-Peer Systems: Routing Distances and Fault Resilience. ACM SIG- COMM (2003)

[6] J. Li, J. Stribling, T. Gil, R. Morris, F. Kaashoek: Comparing the performance of distributed hash tables under churn. IPTPS (2004)

[7] S.S.Lam and Huaiyu Liu: Failure Recovery for Structured P2P Networks: Protocol Design and Performance Evaluation. ACM SIGMETRICS/Performance ’04 (2004)

[8] K. Graffi: 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), 2011