# Performance Evaluation of Structured Peer-to-Peer Overlays for Use on Mobile Networks

**Conference Paper** · December 2013

DOI: 10.1109/DeSE.2013.19

**2 authors:**

Farida Chowdhury
Shahjalal University of Science and Technology
**29** PUBLICATIONS **182** CITATIONS
SEE PROFILE

Mario Kolberg
University of Stirling
**99** PUBLICATIONS **1,287** CITATIONS
SEE PROFILE

# Performance Evaluation of Structured Peer-to-Peer Overlays for Use on Mobile Networks

Farida Chowdhury
Computing Science and Mathematics
University of Stirling
Stirling, Scotland
fch@cs.stir.ac.uk

Mario Kolberg
Computing Science and Mathematics
University of Stirling
Stirling, Scotland
mko@cs.stir.ac.uk

***Abstract*—Peer-to-Peer (P2P) overlay networks are typically deployed on fixed wired networks. Distributed Hash Table (DHT) based structured P2P overlay networks offer scalability, fault tolerance and a self-managing infrastructure which does not exhibit single points of failure. Deploying DHT based overlays on mobile networks is a natural next step to make DHT based systems more widely available. However, supporting P2P on mobile networks poses a number of challenges, such as restrictions in terms of data transmissions on cellular networks, limited battery power of the handsets, high levels of churn due to variations in signal strength and placing or receiving voice calls which suspend the data link. While comparative studies of overlay algorithms for wired networks have been reported, there are a very few studies on the suitability of different existing DHT based overlays for mobile networks. This paper evaluates the performance and efficiency of five popular DHT based structured P2P overlays: Chord, Pastry, Kademlia, Broose and EpiChord, taking into account the conditions as seen in mobile networks. Our experimentation has shown that Kademlia and EpiChord are good candidates for use in high churn mobile environments.**

***Keywords*—Peer-to-Peer overlay, Distributed Hash Table, churn, Mobile Networks.**

## I. Introduction

In the last few years, data service consumption using mobile phones has vastly increased fuelled by the popularity of smartphones. At the same time, Peer-to-Peer (P2P) technology has gained popularity due to its capability to facilitate information storage and retrieval among a potentially very large number of nodes without the use of central servers. P2P has been used primarily for content distribution but more recently it is also used for multicasting, distributed collaboration and grid computing.

P2P applications have been very successful on fixed and wired networks. However, desktop machines have extensive processing power and plentiful bandwidth. In contrast to this, mobile networked devices often have strict bandwidth limits and limited processing power. These limitations are typically caused by expensive bandwidth costs and restrictive battery performance. In addition, sending and/or receiving data on cellular networks is energy intensive and expensive in terms of bandwidth. Furthermore, in a mobile phone network setting, the performance can be affected significantly by frequent join and leave events of the mobile nodes, creating very high levels of churn. As the proportion of mobile nodes increases in the network, many P2P overlays simply collapse under high churn rate [1], [2]. With P2P applications becoming increasingly popular for networked devices, P2P systems are required to meet the challenges of limiting the network overhead to prolong battery life and coping with high levels of churn.

DHT based P2P overlays provide a structured routing architecture, by which any peer can route a search request towards another peer which stores the desired data. Some well-known examples of structured P2Poverlays are Chord[3], Pastry[4], Kademlia[5], Broose[6], Bamboo and EpiChord[7]. The performance of different structured P2P overlays over wired and wireless networks have previously been evaluated in [1], [8], [9]. The effects of low levels of churn have been studied in [10]. However, the more extreme requirements which the public mobile data networks pose, have not been taken into account. This paper addresses this need. The contributions of our paper are:

- Performance evaluation of four DHT based structured *multi-hop* P2P overlays: Chord, Kademlia, Broose and Pastry; and one *one-hop* P2P overlay: EpiChord, considering performance under high levels of churn and bandwidth consumption as required in mobile networks.
- Identifying the most suitable configuration of these five DHTs to achieve the optimal performance in mobile environments.

The rest of the paper is organized as follows. Section II highlights the related work. Section III presents the suitable features of a DHT for use on mobile networks and Section IV presents a brief overview of the selected overlays. Section V describes the experiment design and simulation setup. How the best parameters for each overlay have been selected was discussed in section VI. Performance evaluations are discussed in Section VII and we summarise in Section VIII.

## II. Related Work

DHT based P2P overlays in wired network have received significant attention and therefore the performance have been evaluated in some studies [1], [8], [9], [10].

In [8], the authors compared Chord, Koorde, Pastry, Bamboo, Kademlia and Broose. In this paper, the evaluation methodology was based on the Performance vs. Cost (PVC) evaluation framework[1]. While churn settings were simulated, the simulation parameters were selected according to the requirements of stable fixed networks. In our paper, we studied P2P overlays under conditions as found in cellular networks such as high levels of churn and limited bandwidth consumption; and identify suitable overlays for use in mobile networks. In [11], the performance and efficiency of Kademlia and Chord was compared. In [9] the authors analysed Chord, Kelips and Tapestry. However, the experiments were conducted with only small networks of 1000 nodes network whereas we have experimented with 10,000 nodes network. In [12], several multi-hop DHT algorithms' suitability for interpersonal communication are analysed.

## III. PROTOCOL OVERVIEWS

This section provides a brief summary of the evaluated DHT overlays.

Chord[3] uses consistent hash function to generate an $m$-bit identifier for each node and resource, where $m$ is a pre-defined system parameter. Nodes are ordered to form a ring circle of modulo $2^m$. Each node has a successor and a predecessor on the ring. Each node maintains a routing table which consists of two parts: the first part is a finger table that contains the routing $m$ entries and the predecessor of this node. The 2nd part is a neighbour table with the successor list of size $r$. When a lookup is initiated, it is forwarded clockwise to its closer nodes. The OverSim Chord model has been validated in [17].

Pastry[4] also consists of a circular identifier space where each node and data item are assigned a 128-bit identifier using a consistent hash function. In addition to the routing table, each node in Pastry maintains a Leaf set and a Neighbourhood set. Pastry uses prefix routing to route messages where each entry in the routing table contains the IP address of peers whose identifiers have the appropriate prefix and it is chosen according to a close proximity metric. An evaluation of the OverSim Pastry model can be found in [17].

Kademlia[5] is another DHT based P2P overlay that assigns a NodeID in each node in the 160-bit key space and the [key,value] pairs are stored on nodes with IDs close to the key. It uses XOR-based metric topology for routing. Kademlia nodes contain lists of entries, known as buckets, which are used to send parallel lookups.

Broose[6] is based on the De-Bruijn topology that allows a distributed hash table to be maintained in a loose manner. Similar to Kademlia, it stores an association of $k$ nodes instead of one to get high reliability without any node failures and uses a constant size routing table of $O(k)$ contacts for allowing lookups in $O(\log N)$ messages exchange.

EpiChord[7] is a DHT algorithm where peers maintain a full routing table and ideally approach $O(1)$ hop lookup performance compared to the $O(\log N)$ hop performance offered in many multi-hop networks. EpiChord is based on the Chord DHT and organized as a one-dimensional circular space where each node is assigned a unique node identifier. The node responsible for a key is the node whose identifier most closely follows the key. In addition to maintaining a list of the $k$ succeeding nodes, EpiChord also maintains a list of the $k$ preceding nodes and a cache of nodes. Nodes update their cache by observing lookup traffic. Therefore nodes add an entry anytime they learn of a node not already in the cache and remove entries which are considered dead.

## IV. FEATURES OF A DHT FOR MOBILE NETWORKS

Before starting the experimentation, we need to determine a set of key features that can be used to compare against the characteristics of Chord, Pastry, Kademlia, Broose and EpiChord to determine the suitability of these DHTs in mobile networks. A set of features suitable for application scenarios running on mobile networks was proposed in [12]. Here, we briefly summarize the key findings:

- **Suitability for Mobile Devices**: Mobile devices connected to mobile networks will be the primary terminals to host such applications. These devices have limited hardware resources and comparatively lower bandwidth connectivity than wired nodes. Thus a suitable DHT should require minimum data transmissions and processing.
- **Delay in DHTs**: It will be important to minimize the lookup delay as much as possible in such settings as it will ensure that network resources such as bandwidth are not wasted during certain operations. One way to minimize the delay during lookup operations in DHTs is to reduce the number of hops for each lookup.
- **Robustness**: A suitable DHT for such settings should have concrete mechanisms to handle the higher degree of churn of extremely volatile mobile nodes which might become unavailable at any time. One metric to indicate the robustness of a DHT is its lookup success ratio. The DHT should maintain an acceptable lookup success ratio during higher churn scenarios.

## V. EXPERIMENT DESIGN

For our experiments, we use the OverSim [13] platform implemented in C++ and developed on top of OMNeT++ framework, due to its strong support of DHT based P2P algorithms. Also OverSim provides an interactive GUI and real-time messages which are extremely useful for debugging.

### *A. Performance Metrics*

The following performance metrics are evaluated from conducted simulation:

- **Lookup Success Ratio:** The percentage of successful lookups in the overlay.
- **Mean Maintenance Traffic Load:** The mean number of maintenance bytes sent per second by each node. This metric will be considered as a parameter for bandwidth consumption in this paper.
- **Hop Count:** The mean number of overlay hops to send a message from a source to a destination node.

### *B. Simulation setup*

Experiments have been carried out using a 10,000 nodes network. The level of churn is simulated by various mean lifetime values ranges from 100 seconds to 1000 seconds using Oversim's Lifetime Churn model using Weibull distribution. A shorter lifetime means a higher level of churn and vice versa. This has been chosen as the authors in [14] showed that churn in P2P overlays is much more accurately modelled by weibull distribution. Each configuration was repeated 5 times, and results were averaged. The overlays have been configured to use iterative routing.

## VI. Best Parameter Selection

The simulation process includes two steps where in the first step, wide ranges of parameter values of Chord, Kademlia, Pastry, Broose and EpiChord have been used as input. This step is to establish the values of key parameters for each overlay to yield the best possible performance under high churn. Based on the results obtained from the first step, the second step involves only the selected 'best' values of the parameter set to evaluate performance under churn. Table I summarizes the parameters and their values for Chord, Kademlia, Pastry, Broose and EpiChord.

TABLE I. Simulation Parameters

| P2P Protocols | Parameters | 1st step | 2nd step |
|---|---|---|---|
| Chord | Stabilize Delay (sec) | 5, 10, 20, 30, 40, 50, 60, 120 | 20 |
| | Fix-finger Delay (sec) | 30, 60, 90, 120, 180, 240, 300 | 30 |
| | Successor List Size | 4, 8, 16, 32 | 8 |
| | CheckPredecessor delay(s) | 5, 10, 30, 60 | 5 |
| | Size of Extended Finger Table | 0, 1, 4, 8, 16 | 0 |
| Kademlia | Bucket Size, k | 4, 8, 16, 32, 40, 64, 72, 80 | 8 |
| | Number of Siblings, s | 2, 4, 8, 16, 32 | 2 |
| | Number of Bits, b | 1, 2, 4, 6, 8 | 1 |
| | Lookup Redundant Nodes | 1, 2, 4, 8, 16, 20 | 8 |
| | Bucket Refresh Interval (sec) | 100, 600, 1000, 3000, 5000 | 1000 |
| | No. of Parallel Lookups | 1, 2, 3, 4, 5 | 3 |
| Pastry | Bits per Digit | 1, 2, 4, 6 | 4 |
| | Number of Leaves | 4, 8, 16, 32 | 8 |
| | Lookup Redundant Nodes | 2, 4, 8 | 4 |
| | Number of Neighbors | 0, 2, 4, 8, 16 | 0 |
| Broose | Bucket Size | 4, 8, 16, 32 | 8 |
| | Shifting Bits | 2, 3, 4 | 2 |
| | Bucket Refresh Interval (sec) | 30, 60, 120, 180, 300, 600 | 30 |
| | No. of Parallel Lookups | 1, 2, 3, 4, 5 | 3 |
| EpiChord | Stabilize Delay (sec) | 10, 20, 60, 100 | 60 |
| | Successor List Size | 2, 4, 8 | 4 |
| | No. of Parallel Lookups | 1, 2, 3, 4, 5 | 3 |

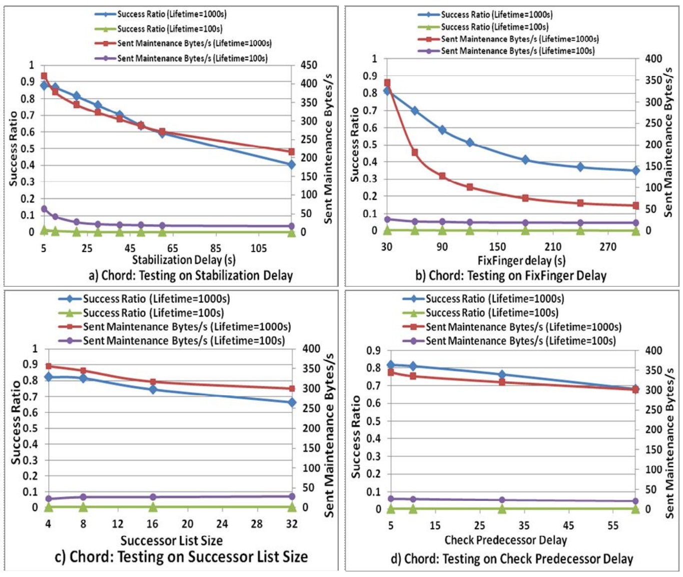
### A. Chord

In the first step of the simulation, we have experimented with Chord using different parameters. Fig. 1 shows the results for *success ratio* and *bandwidth consumption* with varying values for stabilization delay, fixfinger delay, successor list size, check predecessor delay and extended finger table size.

In Chord, each node learns about newly joined nodes and updates their successor(s) and predecessor after a stabilisation delay. In Fig. 1(a) the stabilisation delay was varied from 5s to 120s, which showed that a low stabilisation delay improves the success ratio. Therefore a low value of 20s was chosen.

In Chord, finger (routing) tables are updated in each fix fingers interval. The fix fingers interval was changed in Fig. 1(b) and unsurprisingly we found that a lower value is the better to get a good success ratio. We have selected a value of 30s to get the best possible lookup success ratio at a moderate level of bandwidth consumption.

The successor list size and check predecessor delay were altered in Fig. 1(c) and 1(d) respectively. A successor list size of 8 and the check predecessor delay of 5s were chosen for the next step to maintain a reasonable success ratio while minimizing the maintenance traffic.



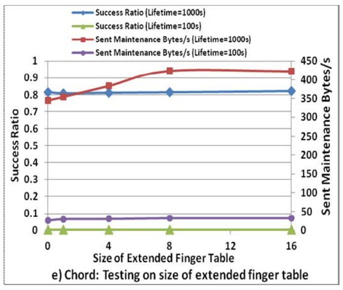


Fig.1. Chord: Selecting best parameters according to a) stabilization delay, b) fixfinger delays, c) successor list sizes, d) check predecessor delays and e) extended finger table sizes.

Fig. 1(e) shows the effect of using extended finger tables. Based on these results, we have not increased the size of the extended finger table as increasing it leads to more traffic with no significant improvement in performance.

Overall, we note the poor lookup performance of Chord at high churn. We revisit this fact in the second simulation step when comparing the different overlays against each other.

### B. Pastry

In Fig. 2, key parameters of Pastry are evaluated. Fig. 2(a) shows the effect of different numbers of leaf nodes. In low churn scenarios, there is no significant performance difference whereas we observed the highest success ratio at the value of 8 in high churn scenarios. Hence the number of leaf nodes of 8 was selected. The results of varying levels of bits per digit are shown in Fig. 2(b). Using higher number of bits can improve the success ratio as well as hop count (figure omitted) but it increases the maintenance traffic cost. Considering the high churn scenario results, we have selected 4 bits per digit. In Fig. 2(c), we simulated Pastry with neighbour sets and without neighbour sets and did not notice any major improvements in success ratio (and bandwidth consumption). Therefore we decided not to use neighbour sets.

In Fig. 2(d), the number of redundant nodes was altered. Higher values of redundant nodes can improve the success ratio but came with an increased amount of maintenance traffic. Therefore a fixed value of 4 was chosen to keep a balance between a good lookup success ratio and maintenance traffic required.

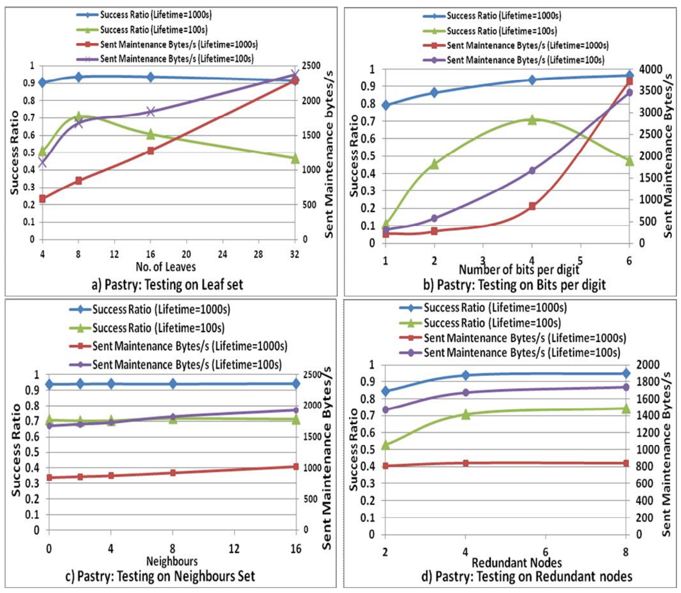


Fig. 2. Pastry: Selecting best parameters according to a) Leaf set, b) Bits per digits, c) No. of redundant nodes and d) No. of Neighbours.

## C. *Kademlia*

Fig. 3 illustrates the *success ratio* and *bandwidth consumption* for different parameters of Kademlia.

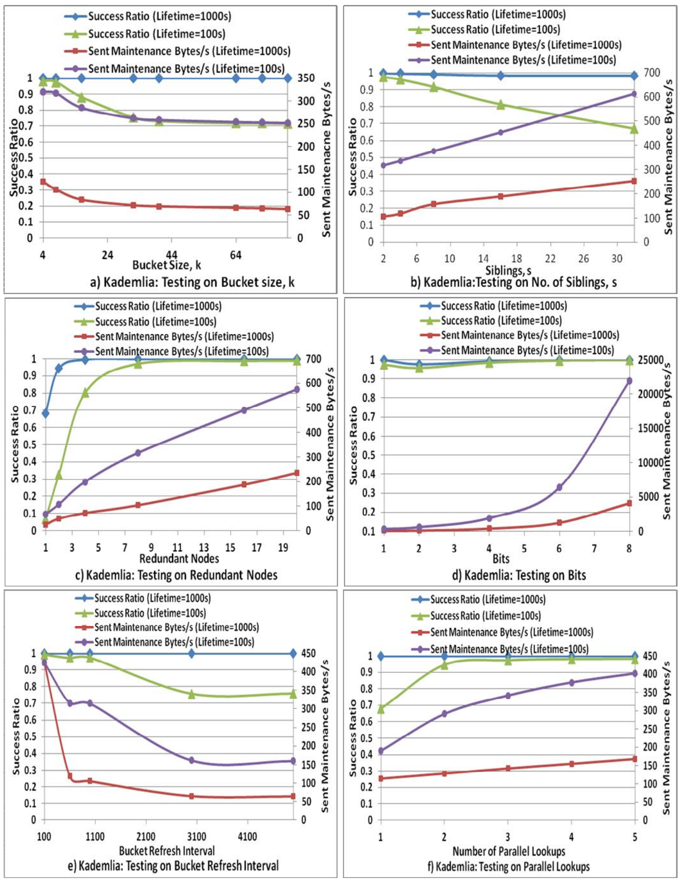


Fig. 3. Kademlia: Selecting best parameters according to a) Bucket size, b) No. of siblings, c) No. of redundant nodes, d) Bits per digits, e) Bucket refresh interval and f) No. of parallel lookups.

The number of buckets (*k*), number of siblings (*s*), number of redundant nodes (*r*) and number of bits per digit (*b*) were varied in Fig. 3(a), 3(b), 3(c) and 3(d) respectively. To achieve the best performance, we evaluated the traded-off between a higher success ratio and the increased maintenance traffic. Consequently, we selected the value of 8 for *k*, 2 for *s*, 8 for *r* and 1 for *b*. In Fig. 3(e), the bucket refresh interval was altered. As shown in the figure, an interval of 1000s is sufficient to keep the buckets consistent. As Kademlia supports parallel lookups, we investigated the effect of parallel lookups in Fig. 3(f). In high churn environments, the higher degree of parallelism increases success ratio significantly. Therefore we selected a value of 3 for lookup parallelism with a moderate increase in bandwidth.

## D. *Broose*

Fig. 4 shows the results for different parameters of Broose.

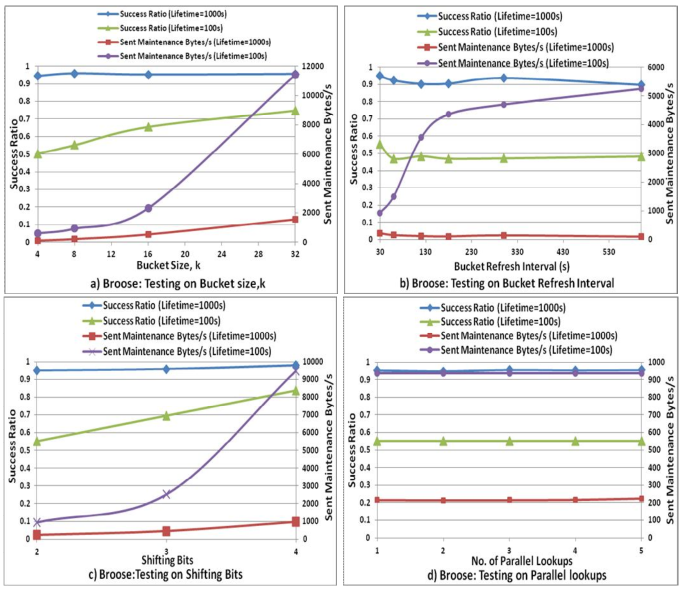


Fig. 4. Broose: Selecting best parameters according to a) Bucket size, b) Bucket refresh interval, c) Shifting bits and d) No. of Parallel lookups.

The bucket size, *k* and bucket refresh interval were varied in Fig. 4(a) and 4(b) respectively. The results in 4(a) exhibit that success ratio is improved with higher values, however due to the expensive joining process the maintenance traffic increases drastically. Therefore a *k* value of 8 is appropriate. In terms of the bucket refresh intervall, to get the best performance as well as moderate level of maintenance traffic under high churn, a value of 30s was selected. Shifting more than one bit at each routing step clearly improves the success ratio, however, also increases the maintenance traffic. Thus we have selected a value of 2 bits (Fig. 4(c)). Increasing parallel lookups does not show any significant lookup performance improvement (Fig. 4(d)). Hence the value of 3 was selected.

## E. *EpiChord*

Fig. 5 reveals the results of *success ratio* and *bandwidth consumption* for EpiChord. The stabilization delay and successor list size were altered in Fig. 5(a) and 5(b) respectively. Increasing the stabilization delay does not affect

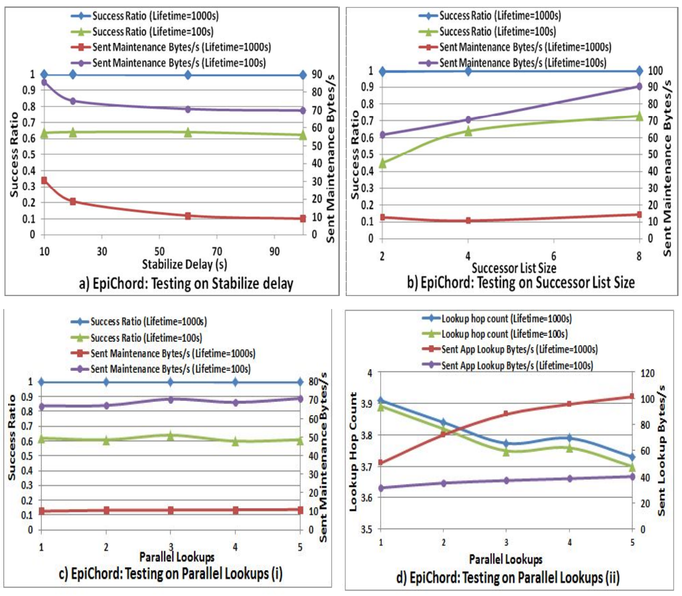


Fig. 5. EpiChord: Selecting best parameters according to Stabilize delay, b) Successor list size, c) Parallel lookups(i) and d) Parallel lookups (ii).

performance of EpiChord regarding the success ratio and bandwidth consumption; therefore a value of 60s was selected. The successor list size of 4 is a fair choice to get a good performance regarding the higher success ratio and lower maintenance traffic. Fig 5(c) shows the effect of parallel lookups on success ratio and maintenance traffic. However, in EpiChord, the parallel lookup does not substantially affect the success rate and the maintenance traffic. Rather the hop count (path length) to the destination node is improved. Also as the parallelism applies to lookup messages only, considering the lookup traffic is a better measure of cost. In Fig. 5(d), we show the lookup hop count and lookup traffic for varying levels of parallelism. We have opted for a level of parallelism of 3 as a reasonable choice for the next step.

## VII. Performance Evaluation

In the following, we present the simulation results of the chosen P2P overlays with the selected parameters from the previous step, and evaluate the performance under high level of churn according to lookup success rate and bandwidth consumption.

### A. *Effects of Churn*

In this experiment, the performance of the overlays under high levels of churn was observed. Results in Fig. 6 show the lookup success rate under varying levels of churn. It is evident from the figure that Chord's performance is heavily affected by high levels of churn. For a node lifetime of 100 sec the lookup success ratio collapses to a mere 2%. While this increases to about 80% as the node lifetime increases to 1000 sec, it appears Chord is not a good candidate for a network consisting of mobile nodes. One likely reason for Chord's performance is that Chord does not immediately correct its successor and predecessor pointers of all relevant nodes as new nodes join, but waits for the periodic stabilization process to update the pointers. Therefore under high levels of churn when nodes join and leave the network at a very high rate, routing table entries are not updated sufficiently frequently.

Broose uses the De Bruijn topology and allows every message exchange to reinforce its contact information. Broose maintains a simple routing table by keeping only two buckets of 'neighbours'. In our experiment, Broose exhibits around 55% of lookup success ratio at 100 sec of node lifetime (this increases to about 95% at 1000 sec of node lifetime). However, if nearly half of lookups fail in high churn networks, Broose also does not seem well suited for such environments. This is emphasized by the fact that Broose has comparatively high bandwidth costs in high churn situations (Fig 7 a).

In Pastry, the lookup success ratio is 70% at 100 sec of node lifetime, increasing to around 93% at 1000 sec of node lifetime. While this performance is acceptable, this comes at the price of the highest maintenance cost (with quite a large margin). Thus Pastry has to be ruled out due to this cost.

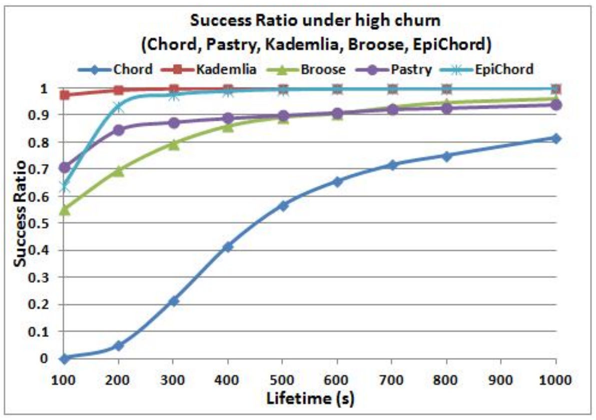


Fig. 6. Successful Lookup Ratio of Chord, Kademlia, Pastry, Broose and EpiChord operated on 10,000 networks under various level of churn

Kademlia exhibits a 97% successful lookup rate in the high churn environment (100 sec mean node lifetime). Kademlia sends parallel lookup requests to avoid timeout delays from failed nodes. Also Kademlia updates routing table entries from data attached to lookups rather than requiring separate maintenance requests. These characteristics paired with a modest bandwidth usage and modest lookup count (4 for 10,000 nodes network) as shown in Fig 7(a) and (b) respectively, make Kademlia well suited for mobile environments.

Finally, EpiChord, an overlay which can approach one-hop performance, approached 99% success ratio when the churn level went low (lifetime=1000s). In very high churn, the overlay performed less favourably with a lookup success rate of about 63%. This increases to a more usable 97% as the mean lifetime approaches 300 seconds. Like Kademlia, EpiChord uses reactive routing state maintenance employing lookup response messages to carry routing table update information. EpiChord also uses parallel lookup messages.

### B. *Bandwidth Consumption*

Figure 7(a) shows the results on the required bandwidth for the selected overlays under varying levels of churn. In high churn environments when the node lifetime is 100s, Chord and EpiChord have the lowest bandwidth cost using 26 bytes/s and 70 bytes/s respectively. However, as already discussed, Chord's lookup performance is very poor under these conditions. Pastry consumes the highest bandwidth which is

1674 bytes/s at this level of high churn. This is due to its expensive joining process and reactive maintenance for the routing table, leaf set and neighbour list. Broose also requires significant bandwidth at high churn levels (934 bytes/s per node at the node lifetime of 100s). Kademlia consumes higher traffic (316 bytes/s) than Chord and EpiChord however less than Pastry and Broose in very high churn rate.

It is noticeable that Chord behaves differently, unlike all other P2P overlays compared here, in scenarios when the node lifetime increases the maintenance traffic slowly increases instead of decreasing. It is because that Chord periodically calls its stabilization process and sends some other time related messages to update finger table (routing table) which increases the bandwidth as nodes stay in the network for longer.

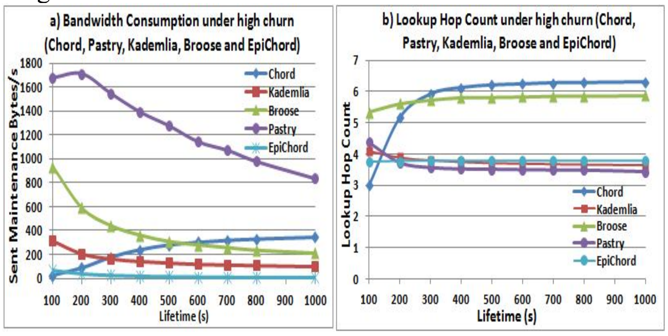


Fig. 7. a) Bandwidth consumption and b) Lookup hop count of Chord, Kademlia, Pastry, Broose and EpiChord under various levels of churn.

Lookup hop count is plotted in Fig. 7(b). Kademlia, Pastry and EpiChord show similar hop counts (~3.5) under all varying levels of churn whereas Chord and Broose exhibit the highest lookup hop count (besides low success rates).

In our experiments, each node performs lookups every 60 seconds in all overlays. However, as we show sepaerately, EpiChord can achieve a better hop count performance at the cost of an increased number of lookups [15], [16].

## *C. Discussion*

Based on the simulations of five popular structured P2P overlays, the following insights have been found:

- Kademlia shows the best result in terms of successful lookup ratio at very high churn and is well suited for mobile environments.
- Chord lookup performance collapses to 2% under high churn and thus is not suitable for such environments.
- EpiChord shows moderate bandwidth consumption and a good lookup success performance even under high churn. Note that EpiChord's performance can be improved further by introducing additional lookups to the network.
- Broose shows the average result for lookup success but consumes comparatively more bandwidth than EpiChord and Kademlia.
- Pastry requires considerably higher bandwidth under high churn than the other overlays and thus is not a good candidate for mobile environments.

The requirements of mobile networks where the churn rate is expected to be high and the bandwidth availability is low matches well with Kademlia and EpiChords' high lookup success ratio, reasonable amount of bandwidth consumption and low hop count and thus make these two suitable candidates to be used in mobile networks.

## VIII. Summary

This paper presents a performance evaluation of Chord, Pastry, Kademlia, Broose and EpiChord in the presence of high churn and investigated their suitability for mobile networks. The simulation results suggest Kademlia as the most appropriate P2P overlay to implement on the mobile network according to lookup success ratio under high levels of churn. At the same time Kademlia consumes moderate level of bandwidth (i.e. 316bytes/s) which is also acceptable in mobile networks where the typical transfer rate is 10-100KB/s. Similarly, EpiChord also achieves a high success ratio while consuming the least amount of bandwidth making it also a strong candidate algorithm for mobile environments.

## References

[1] J. Li, J. Stribling, R. Morris, M. Kaashoek, and T. Gil, "A performance vs. cost framework for evaluating DHT design tradeoffs under churn", 24th IEEE Infocom, vol. 1, Mar. 2005, pp. 225–236.

[2] H. Pucha, S. M. Das, and Y. C. Hu, "How to implement DHTs in mobile ad hoc networks", 10th ACM Intl. Conf. on Mobile Computing and Network, Sept. 2004.

[3] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. "Chord: A scalable Peer-To-Peer lookup service for internet applications", ACM SIGCOMM, pages 149–160, 2001.

[4] Antony Rowstron and Peter Druschel. "Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-peer Systems". IFIP/ACM Int. Conf. on Distributed Systems Platforms (Middleware), pages 329–350, Heidelberg,Germany, 2001.

[5] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the XOR metric," 1st Intl. Workshop on Peer-to-Peer Systems (IPTPS 02), London, UK, 2002 pp. 53-65.

[6] T. Gai and L. Viennot, "Broose: a practical distributed hashtable based on the de-bruijn topology," in Fourth International Conference on Peer-to-Peer Computing, 2004, Aug. 2004, pp. 167–174.

[7] B. Leong, B. Liskov, and E. D. Demaine. EpiChord: Parallelizing the Chord Lookup Algorithm with Reactive Routing State Management. *Computer Communications,* Elsevier Science, Vol. 29, pp. 1243-1259.

[8] I. Baumgart and B. Heep. "Fast but economical: A simulative comparison of structured peer-to-peer systems", in Proceedings of the 8th Euro-NF Conference on Next Generation Internet. IEEE, June 2012.

[9] Hung Nguyen Chan, Khang Nguyen Van, Giang Ngo Hoang, "Characterizing Chord, Kelips and Tapestry algorithms in P2P streaming applications over wireless network", ICCE 2008, Hoi An, Vietnam.

[10] Sean Rhea, Dennis Geels, Timothy Roscoe and John Kubiatowicz. "Handling churn in a DHT". USENIX Annual Tech. Conf., 2004.

[11] Harjula, E.; Koskela, T.; Ylianttila, M., "Comparing the performance and efficiency of two popular DHTs in interpersonal communication", *Wireless Communications and Networking Conference,* March 2011.

[12] Jani, H. and C. Gonzalo. 2007. Evaluation of DHTs from the viewpoint of interpersonal communications. 6th Int. Conf. on Mobile and ubiquitous multimedia. ACM: Oulu, Finland.

[13] B. Heep I. Baumgart and S. Krause, "Oversim: A flexible overlay network simulation framework", 10th IEEE Global Internet Symposium in conjunction with IEEE INFOCOM, Anchorage, AK, USA, 2007.

[14] D. Stutzbach and R. Rejaie, "Understanding Churn in Peer-to-Peer Networks", 6th ACM SIGCOMM conference on Internet measurement, Rio de Janeiro, Brazil.

[15] J. Furness, F. Chowdhury, M. Kolberg. "An Evaluation of EpiChord in OverSim", 5th Int. Conf. on NETCOM-2013, Chennai, India,2013.

[16] F.Chowdhury, M.Kolberg. Performance Evaluation of EpiChord under High Churn. 8th ACM PM2HW2N Workshop, Barcelona, Spain, 2013.

[17] J. Furness, M. Kolberg, M. Fayed. "An evaluation of Chord and Pastry models in OverSim". European Modelling Symposium (EMS2013), Manchester, UK, 2013.