An Aggregation-Based Routing Protocol for Structured Peer to Peer Overlay Networks

Nicolas Hidalgo*, Luciana Arantes*, Pierre Sens* and Xavier Bonnaire†
*Université Pierre et Marie Curie, CNRS
INRIA - REGAL, Paris, France
Email: [nicolas.hidalgo, luciana.arantes, pierre.sens]@lip6.fr

†Department of Computer Science
Universidad Tecnica Federico Santa Maria, Valparaiso, Chile
Email: xavier.bonnaire@inf.utfsm.cl

***Abstract*—Structured peer-to-peer (P2P) overlay networks provide a scalable object location and routing substrate for large scale distributed applications. However, due to the great number of nodes of such systems, message complexity of their routing protocol may considerably increase network traffic and average node hops of a message. This paper presents a novel Pastry-based routing protocol for structured P2P systems, which is specially suitable for handling per node multiple message routing requests. Our protocol exploits message aggregation and implements a multi-slice mechanism which multiplexes the sending of aggregated messages. Experimental results on top of PeerSim show that our protocol can reduce the average number of node hops messages, and thus, the global traffic and load of the network.**

***Keywords*-Peer-to-Peer - Aggregation - Routing**

I. INTRODUCTION

Structured peer-to-peer (P2P) overlay networks such as Pastry [1] or Chord [2] are distributed self-organizing substrates which provide efficient routing and object location for large-scale applications. Each node has a unique $nodeId$ which is randomly assigned from an uniform identifier space. Objects have also unique keys taken from a large identifier space. Every key is mapped to the node whose $nodeId$ is numerically closest to the key. An object lookup service then routes an object lookup request to the node responsible for the key of the object. These systems usually present low object lookup latency since their routing protocol is based on Distributed Hash Table (DHT). DHT-based overlays route messages in a logarithmic number of hops, typically $O(Log(N))$, where N is the number of nodes of the network. Such a property allows nodes to maintain routing tables of small sizes. Furthermore, the system can scale up to a few millions of nodes.

In this paper, we are interested in P2P overlay networks where a node may have many lookup message requests to dispatch at a given time, i.e., a burst of messages. This happens, for instance, when a node wants to get several different objects at the same time (e.g. multi-query). A second scenario is hybrid/hierarchical network architectures composed by two types of network: a structured P2P overlay network, whose nodes are continuously connected to the Internet by wired links, and local networks, such as wireless networks, whose nodes can connect to the P2P overlay but such connections are intermittent. Thus, the P2P overlay network provides routing and lookup object service to the nodes that do not belong to it. Since the latter have temporary connections to the P2P overlay network, when they connect to it, several lookup request messages are probably sent to the node of the P2P overlay, which is their point of connection to the network. In other words, the P2P node behaves like a *proxy* to the former and every time a connection between them is established, the *proxy* node will have many messages to route, i.e., a burst of messages. An example of hybrid architectures is the *nano data centers*. Such centers refer to P2P architectures composed of controlled and stable peers, typically set-up-box, where domestic devices (e.g. PDA, PC, Mobile Phone, etc.) can be connected.

In such a context, we propose a routing protocol which combines several messages into a single one. Since the efficiency of a lookup service is usually measured as a function of the number of node hops to route a message to the node responsible for the message's key, the aim of our aggregation-based protocol is to reduce the average number of hops of messages. A second and important goal of our approach is to reduce message traffic for performance reasons.

We have also added to our protocol a multi-slice mechanism that divides an aggregated message into smaller ones which then are simultaneously routed to different continuous area of the identifier logical space. This mechanism reduces the average transmission delay of an aggregated message.

Our aggregation-based routing protocol was built using Pastry overlay [1]. Performance results obtained from experiments conducted on top of PeerSim [3] confirm that our protocol reduces both network traffic and the average number of node hops of a message.

The rest of the paper is organized as follows. Section

 ISBN: 978-1-61208-102-1

III describe our aggregation-based routing protocol and the multi-slice mechanism. Simulation performance results are shown in Section IV, while some related work are described in Section II. Finally, Section V concludes the paper.

II. Related Work

In order to disseminate information about membership in a ring-organized P2P system, Gupta et al. [4] use the concept of aggregation of messages and slice. The circular identifier space is divided into k equal contiguous slice. The members of each slice are represented by a leader node. Whenever a node detects a membership change it notifies its leader. This one aggregates the notification messages it receives into a single one during a period of time t and then dispatches it to all the other leaders. The latter then diffuses the message to the members of their respective slices. Mizrak et al. [5] also propose to split the circular identifier space into arcs (slice) and assign each one to a super-peer node. A super-peer is a high-capacity node which is responsible for routing messages to the nodes of its slice as well as to other super-peers. Similarly to our approach, message aggregation and/or the concept of slice are exploited by these works. On the other hand, in our multi-slice mechanism, it is not the identifier space that is divided into slices but the message M and the range of all keys of M does not necessarily cover the entire ring. Furthermore, aggregation of message is used for routing multiple messages and not for maintenance reasons.

Performance results presented in Section IV have shown that our protocol provides an effective mechanism to reduce lookup hops. Several works found in the literature such as One-Hop Route [4], EpiChord [6], and Kelips [7], have the same purpose. They are able to deliver a message in a fixed number of hops, typically $O(1)$. These protocols provide low latency lookup on small or low churn networks. However, if it is not the case, they usually present a lot of extra traffic for membership maintenance when compared to DHT-based protocols [8]. Contrarily to our approach, whenever the number of node hops is reduced, we observe a reduction in message traffic as well.

Some works [9][10] propose a hierarchical architecture for P2P systems. Usually, nodes are organized in disjoint groups which are connected by a DHT-based overlay network. Messages are routed between groups on the inter-group overlay and then routed to the node responsible for the key on intra-group overlay. Like these works, our aggregation-protocol is quite suitable for hierarchical architecture as explained in the introduction. However, contrarily to them, we consider that connectivity between nodes of different layers are not permanent and thus when it is established, nodes of the inter-group overlay receive many messages to route.

III. The Protocol

In this section, we present our aggregation protocol which is based on Pastry routing protocol. We have modified Pastry in order to support message aggregation, i.e, a single routing message M can be composed of several messages m. Our protocol also exploits a multi-slice mechanism which allows to split an aggregated message into several ones and each one is simultaneously routed to a different contiguous slice of the circular identifier space of the P2P overlay network. We should point out that even though our protocol is a Pastry-based one, it can be easily adapted to other structured P2P overlays such as Chord [2].

Aggregation of messages: The main goal of our protocol is to route multiples messages like a single one in order to reduce both message network traffic and the average number of node hops to deliver a message. To this end, the original protocol functions *route* and *deliver* of Pastry application programming interface (API) have been modified: instead of just one message, the *route* function accepts a buffer as input which contains k messages. The set of these messages, $G_m = \{m_1, m_2 \dots m_k\}$, is then combined into a single message M, denoted *k-aggregated message*, which is sent over the P2P overlay network. Using the original Pastry routing protocol, M is firstly routed to the node whose id is the closest one to the key of the first message, i.e., m_1, of G_m. When M reaches its first destination, m_1 is delivered (function *deliver*) and the key of the next message, m_2 in this case, is chosen as the next destination of M. Such a routing/delivery process continues until the k messages of M are delivered.

In order to reduce network traffic, the logical proximity in the logical ring of those nodes that will take part in the routing paths of the gathered messages should be exploited as much as possible. With such a goal, messages in G_m are sorted by increasing order of their respective keys before being grouped into the single message M. The sorting is performed by taking into account both the logical proximity, defined by the DHT itself, of the nodes which store the keys of the G_m messages and the identity of the node which gathered the messages into M. Therefore, the first message to be routed is the first one whose destination node is the closest one to the latter. Notice that in the case where one or more messages have the same destination node, they will be consecutive in M and thus they will be delivered at the same time without any additional hop routing. It is also worth remarking that aggregation and sorting of messages are performed just once by the node that initially called the *route* function.

Multi-slice mechanism: The multi-slice mechanism is an extension added to the aggregation protocol presented above. It aims at dividing the k-aggregated message M into several messages and then multiplexing the sending of these messages. After the messages of G_m have been sorted and gathered in M, the multi-slice mechanism splits M in S messages, i.e., the logical space defined by the keys of the first and the last message of M is equally divided up in S messages M_s. It is worth remarking that the M_s messages

 ISBN: 978-1-61208-102-1

may not have all the same number of k_s messages since the content of them depends on the distribution of the message's keys of M.

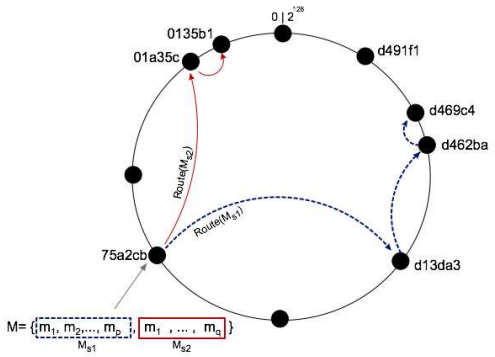

Figure 1. Message routing using the multi-slice aggregation protocol

All messages M_s will then be dispatched at the same time (parallel routing) improving the average message transmission delay when compared to the aggregation approach without the multi-slice mechanism. Each M_s will be firstly routed to the node that corresponds to the key of the first message in the respective M_s. Hence, each M_s will be routed to different slices of the P2P logical ring. In other words, the arc of the logical identifier space ring that encircles the keys of M are divided into S contiguous slices.

An example of the aggregation protocol with multi-slice is shown in Figure 1: the value of S is 2 and the keys of the first message of M_{s1} (p messages) and M_{s2} (q messages) are $d46q1c$ and $01a35b$ respectively.

The number of slices has an influence in the overall performances. On one hand, a high number of slices may induce an important gain in the average transmission delay of messages of G_m; on the other hand, it increases message traffic. Such a behavior is a direct consequence of multiplexing the routing of messages M_s since a smaller number of messages is presented at each *Ms* in relation to the single message M.

IV. Performance Evaluation

This section presents a set of results aimed at evaluating the performance of our aggregation-based protocol and the multi-slice mechanism when compared to Pastry.

A. Simulation environment and configuration

Experiments were conducted on top of the java-based P2P simulator PeerSim[3] jointly with a Pastry protocol plug-in.

The PeerSim Pastry plug-in is an implementation of Pastry[1] overlay over PeerSim. It exploits PeerSim event-based driven model and uses a traffic generator which sends random lookup messages to the system.

Our aggregation protocol implementation is an extension of PeerSim Pastry plug-in. In order to implement both the aggregation of messages and the multi-slice mechanism we have modified the event manager and the delivery functions of PeerSim Pastry. For simulating the per node burst of messages in PeerSim, i.e. the sent of k messages by a node, each node stores every message generated for it in a buffer, instead of sending it immediately. Thus, whenever the buffer of the node contains k messages, the node sends them. To this end, it calls the *route* function: either just once in the case of our protocol (a *k-aggregated* message M), or once for each of the k messages in the case of Pastry. The buffer is then emptied and the node waits for k new messages. To be able to simulate such a per node burst of messages, PeerSim is configured for presenting high message traffic. Notice that for a given k, message traffic increases proportional to network size since for all experiments the average number events per node is the same.

Several experiments were conducted with different configuration values for the number of participant nodes, number of messages k of a burst, and number of slices S. Each experiment was repeated 5 times and the results shown in the graphs are the average among the obtained results. The number of nodes of the system was fixed for each experiment, i.e., there was no failure nor churn and nodes did not leave the system. Messages could not be lost either. However, message transmission delays could vary. Aiming at evaluating both the scalability and stability of the protocols, three different sizes of networks were considered: 100, 1.000, and 10.000 nodes. The value of k varied from 20 to 50 while the number of slices was set to 5 and 10 when the multi-slice mechanism was activated.

B. Evaluation Results

The metrics used to evaluate our protocol are:

- *Network communication traffic* : total number of messages transmitted over the network during the experiment.
- *Average message size*: average size of the messages related to the above network traffic.
- *Average number of hops*: average number of hops required to route a message till its destination node.
- *Average message transmission delay*: average delay for transmitting a message till its destination node;
- *No extra hop message delivery*: number of consecutive messages of M delivered to the same node at the same time, i.e., without additional hop.

1) Network communication traffic: Figure 2 shows the message traffic in logarithmic scale for the original Pastry routing protocol and our aggregation protocol with different k values and network size. We can observe that message aggregation has a direct impact on the reduction of the overall number of messages in the network. Intuitively, the factor of network traffic reduction depends on k. In the best case, our protocol obtains up to 50 times less traffic than traditional Pastry routing. However, when the multi-slice mechanism is applied, the message traffic for a given

 ISBN: 978-1-61208-102-1

network configuration increases as shown in figure 3 since the *k-aggregated* message M is split in S messages M_s (5 and 10 slices in the figure) which are then routed in parallel. In fact, the increase is proportional to the number of slices.

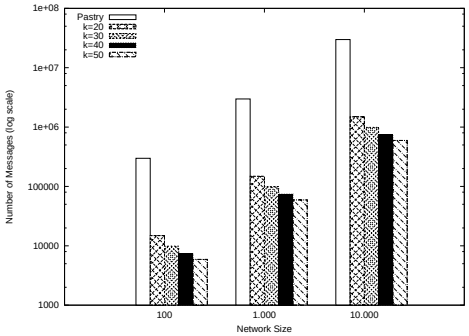

Figure 2. Message Traffic

2) Average Message Size: Table 1 shows the average size of messages for different network sizes, k in M, and number of slices S for the aggregation protocol. Every message generated by the traffic generator of PeerSim Pastry plug-in has 1024 bytes.

k	Slices	Network Size 100	1.000	10.000
20	1	10.601	12.718	19.609
	5	2.147	2.440	3.031
	10	1.066	1.217	1.497
30	1	16.353	20.002	30.160
	5	3.356	3.813	4.006
	10	1.677	1.908	1.937
40	1	22.156	27.545	40.560
	5	4.525	5.211	5.054
	10	2.293	2.617	2.411
50	1	28.047	35.451	50.560
	5	5.721	6.691	6.408
	10	2.908	3.347	3.290

Table I
AVERAGE MESSAGE SIZE (IN BYTES)

One direct consequence of the aggregation protocol is that the average size of messages increases considerably and such a grow depends directly on k. On the other hand, the multi-slice mechanism significantly reduces such a size by a factor proportional to the number of slices. However, we can remark in the table that even with the multi-slice mechanism, the average size of the messages increases with network size. This happens because the greater the size of the network is, the greater the number of hops of a message and the smaller the number of messages of M delivered without extra hops (see the "No extra hop message delivery" discussion bellow). In other words, in smaller networks, messages are routed in less hops when compared to larger networks and the size of a routing message M decreases faster than in larger networks due to the multiple delivery of messages to the same node.

3) Average number of hops: When routing a bundle of k messages, the aggregation protocol (AP) provides a significant reduction in the average number of routing hops necessary to deliver a message to the node associated with the key of the message in comparison with Pastry routing protocol, as we can see in Figure 5. Such an improvement is possible without any change in DHT table. It is in fact due to the routing of key-ordered messages which allows the exploitation of logical proximity of the nodes which correspond to the keys included in M. In addition, such a key-ordered approach enables the delivery of more than one message to the same node at the same time as discussed below, which also justifies why the average number of hops for the 100-node network is smaller than 1.

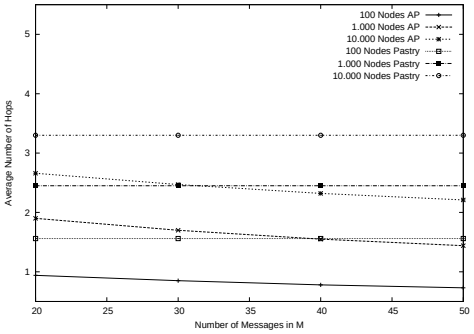

Figure 5. Average number of hops

We can also remark in the same figure that the effectiveness of the aggregation of key-ordered messages increases when the value of k increases for all network configurations. This happens because the probability that the keys of two messages of M have the same destination node increases for higher values of k.

In Figure 4 we can observe the average number of hops necessary to route the k messages of M (*Aggregation*), all the messages of a sliced message M_s when the multi-slice mechanism is applied (5 and 10 slices), and the k messages for Pastry. We can note a slight increase of the average number of hops when the number of slice increases which can be explained by the same reason of the previous figure: when S increases, the number of messages k_s of each M_s decreases, and therefore the effectiveness of the aggregation approach is reduced.

4) No extra hop for message delivery: Figure 6 presents the number of consecutive messages of M which are delivered to the same destination node at the same time, i.e., without extra hops for the aggregation protocol. As we can remark, for a given network configuration, the higher the value of k is, the greater the number of messages delivered without extra hops. On the other hand, for a given k, such a number decreases when the size of the network increases since the probability that two consecutive messages

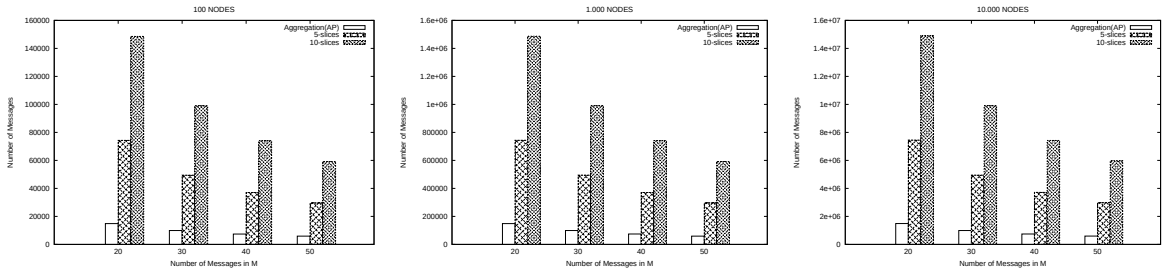

Figure 3. Message Traffic with Multi-Slice Mechanism

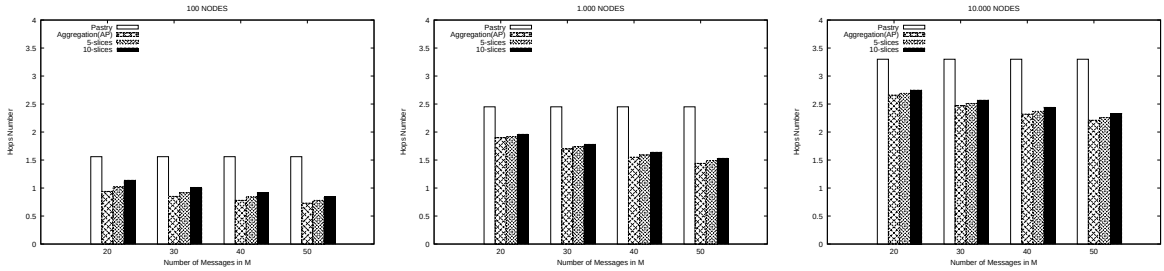

Figure 4. Average number of hops with multi-slice mechanism

of M should be delivered to the same node decreases for larger networks. However, as shown in Figure 7, when the multi-slice is applied for a given network configuration, the number of no extra hop decreases since the number of messages of M_s is inversely proportional to the number of slices S, i.e., the number of message that are delivered with no extra hops decreases when S increases.

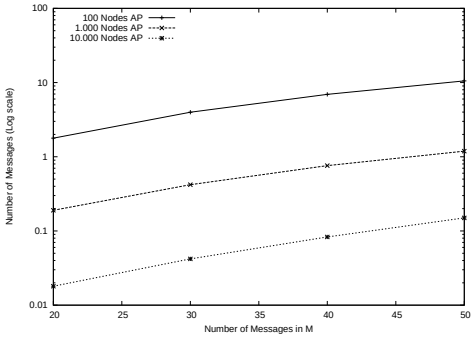

Figure 6. Average number of messages delivered with no extra hops

5) Average message transmission delay: Figure 9 compares the average transmission delay and its standard deviation of both Pastry and our protocol. For the former, such a delay corresponds to the average delay to deliver a bundle of k messages that are individually sent while for the latter it corresponds to the average delay to deliver all the k messages of a *k-aggregated message* M.

As can be observed in the figure, aggregation increases considerably the average accumulated transmission delay of messages. Furthermore, the standard deviation of our protocol is higher than Pastry's which, in its turn, is quite uniform. This happens because in our protocol, transmission delay of a message m of M depends on its position in M, i.e., when M is routed, the last delivered message of M has the highest transmission delay since it has to wait for all the other messages of M to be delivered before it. Such differences on transmission delay explain the standard deviation curves and the fact that it increases with k.

The average transmission delay of messages for different values of k and multi-slice configurations is shown in Figure 8 and its standard deviation.

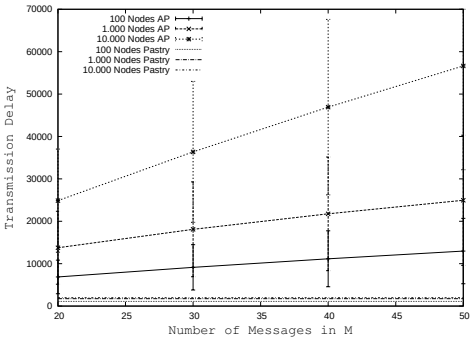

Figure 9. Average transmission delay and standard deviation

The multi-slice mechanism is an effective mechanism to reduce the average transmission delay of messages as well as the corresponding standard deviation. The gain provided by it allows to reduce up to 9 times the message transmission delay when compared to the aggregation approach without

 ISBN: 978-1-61208-102-1

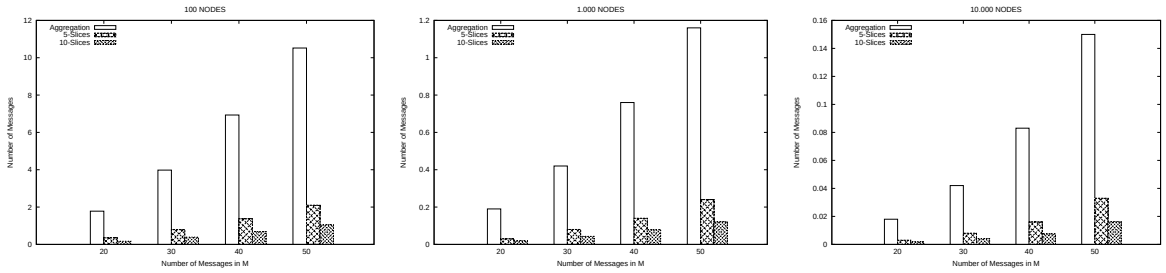

Figure 7. Average number of messages delivered with no extra hops and multi-slice

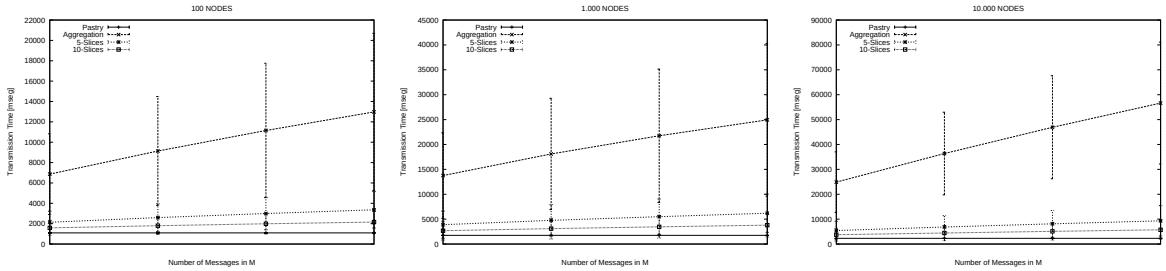

Figure 8. Average message transmission delay and standard deviation in multi-slice mechanism

such a mechanism. The explanation for the reduction is due to the simultaneously routing of M_s messages whose size are smaller than M.

It is worth mentioning that the aggregation protocol with multi-slice provides an interesting tradeoff between the transmission delay of a message and its number of hops: when compared to the aggregation protocol without multi-slice, the transmission delay of a message is reduced, but the average number of hops increases as a result of routing smaller aggregate messages.

V. Conclusion

This paper has presented a novel routing algorithm for structured P2P overlay networks which exploits aggregation of messages to reduce both the message traffic and the load of the network. Performance simulation results have confirmed that our protocol is an effective technique for reducing the average number of node hops of a set of messages without needing any change in the nodes' routing tables. Therefore, our protocol can be easily adapted to other P2P routing protocols.

We have also proposed a multi-slice mechanism extension for our aggregation protocol which provides parallel routing of aggregate messages. In the simulation experiments, an important reduction in the average transmission delay of messages was observed compared to the aggregation protocol without multi-slice.

References

[1] A. I. T. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Middleware*, 2001, pp. 329–350.

[2] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 17–32, 2003.

[3] M. Jelasity, A. Montresor, G. P. Jesi, and S. Voulgaris, "The Peersim simulator," `http://peersim.sf.net`.

[4] A. Gupta, B. Liskov, and R.Rodrigues, "Efficient routing for peer-to-peer overlays," in *NSDI*. USENIX, 2004, pp. 113–126. [Online]. Available: http://www.usenix.org/events/nsdi04/tech/gupta.html

[5] A. T. Mýzrak, Y. Cheng, V. Kumar, and S. Savage, "Structured superpeers: Leveraging heterogeneity to provide constant-time lookup," in *WIAPP '03*, 2003, p. 104.

[6] B. Leong, B. Liskov, and E. D. Demaine, "Epichord: Parallelizing the chord lookup algorithm with reactive routing state management," *Computer Communications*, vol. 29, no. 9, pp. 1243–1259, 2006. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2005.10.002

[7] I. Gupta, K. Birman, P. Linga, A. Demers, and R. van Renesse, "Kelips: Building an efficient and stable P2P DHT through increased memory and background overhead," in *IPTPS*, vol. 2, 2003.

[8] J. Li, J.Stribling, R. Morris, and M. F. Kaashoek, "Bandwidth-efficient management of DHT routing tables," in *NSDI05*, Boston, Massachusetts, May 2005.

[9] M. S. Artigas, P. G. Lopez, and A. F. Skarmeta, "A comparative study of hierarchical dht systems," in *LCN '07*, 2007, pp. 325–333.

[10] L. Garces-Erice, E. Biersack, K. Ross, P. Felber, and G. Urvoy-Keller, "Hierarchical p2p systems," in *Euro-Par 2003*, August 2003, pp. 1230–1239.

 ISBN: 978-1-61208-102-1