Parallelizing Peer-to-Peer Overlays with Multi-Destination Routing

John Buford
Panasonic Princeton Laboratory
Princeton, NJ, USA
buford@research.panasonic.com

Alan Brown
University of Stirling
Stirling, Scotland, FK9 4LA
abr@cs.stir.ac.uk

Mario Kolberg
University of Stirling
Stirling, Scotland, FK9 4LA
mko@cs.stir.ac.uk

Abstract

Distributed Hash Tables (DHTs) are the basic indexing mechanism for decentralized peer-to-peer systems. How to obtain best performance in a large-scale wide area context for DHT operations is an important question. Here we introduce parallelization of overlay and DHT operations using native multi-destination multicasting, resulting in significant message traffic reduction for both overlay maintenance and lookup operations. We show through simulation savings of up to 30% message reduction for the 1-hop EpiChord peer-to-peer overlay.

1. Introduction

We are interested in improving the performance of peer-to-peer overlays by mapping overlay messaging to native multicast paths for overlay operations that are inherently parallel. Here we investigate the impact of using multi-destination multicast routing in the underlay to support overlay operations in the EpiChord [1] 1-hop overlay developed by Leong et al at MIT .

Many multi-hop peer-to-peer structured overlays have been proposed for peer-to-peer applications, but are characterized by O(log N) hop count. In addition, because each overlay hop is routed in the underlay in potentially many native hops, multi-hop overlays have poor latency characteristic for connecting large numbers of peers. Consequently several systems have been developed to trade-off latency for larger routing tables. However these designs lead to increased network traffic for managing the larger routing tables.

While multicast routing could offer efficiency and concurrency to overlay designers, host-group multicast protocols such as PIM-DM, DVRMP, and CBT are not well suited for P2P communication because of the large amount of state that would be needed in routers and the overhead in creating many small-multicast groups. By design, data and services in P2P overlays are widely distributed, so a particular multicast path is not likely to be reused.

Several researchers have analyzed multi-destination routing for small group multicast applications. In multi-destination routing, each packet contains a list of destination addresses. Routers use unicast routing tables to determine when to split a packet depending on the destination addresses contained in the packet. An experimental protocol for explicit multicast (XCAST) has been specified and used by a number of research groups [2].

The concept of multi-destination routing was proposed in the early years of multicast protocol design [3], but as Ammar observes [4], subsequent protocol design focused on enabling large multicast groups. However in the past several years, there has been recognition of multi-destination routing as a complementary multicast technology that has advantages for applications which feature large numbers of small groups. In addition, we observe here that multi-destination routing can benefit overlay operations, and enhance additional features such as overlay multicast and replication.

This paper contains the following contributions:

- We identify for the first time the benefits of native multi-destination routing in parallelizing and reducing message traffic for several classes of peer-to-peer overlays
- We compare through simulation the performance of EpiChord [1], a 1-hop overlay, with XCAST-enabled EpiChord, and show up to 31% messaging reduction for lookup intensive workload for medium sized overlays and a 25% messaging reduction for churn intensive workload. XCAST-enabled EpiChord otherwise retains the performance advantages of EpiChord versus multi-hop overlays.

Separately in [5] we explain how multi-destination routing can be used in several other categories of overlays, including certain multi-hop designs, and unstructured overlays.

The remainder of this paper is organized as follows. We present related work in the next section. Section 3 describes XCAST-enabled EpiChord and Section 4 presents our simulation results under churn and lookup intensive workloads and Section 5 provides analysis of these results. Section 6 concludes the paper.

2. Related work

2.1 Host group multicast

Oh-ishi et al. have considered the use of Protocol Independent Multicast (PIM) [6] in sparse mode (PIM-SM) and source specific mode (PIM-SSM) [7] to reduce message traffic in peer-to-peer systems. Their analysis focuses on using multicast routes between peers in different ISP networks.

The use of host-group multicast in DHT poses the following problems: There would be too much traffic and router overhead if each node maintained multicast addresses for all or many subsets of the overlay network, due to the large number of nodes involved.

If a peer node wants to use native host-group multicast to some set of nodes in order to issue parallel queries, it must first create the state in the routers and bring the receivers into the multicast. This setup adds delay and is only appropriate if the multicast path is going to be re-used for some time. However in peer-to-peer networks the set of nodes is fairly dynamic and the set of requests between nodes is not predictable, so re-use of such multicast groups is limited.

IP multicast is designed for small numbers of very large sets of recipients. So IP multicast is not a good choice for use in parallelizing DHT operations. However, multi-destination multicast routing does not require state in routers. Further, due to overlay routing mechanism, destination peer addresses are already known so there is no group join overhead.

2.2 Multi-destination multicast

In multi-destination multicast, an application request to send unicast packets to p destinations is replaced with a single packet containing the p destinations. Multicast-enabled routers route these packets until a split point is reached (according to unicast routing decision). At each such point, duplicate packets containing the subset of destinations for each forwarding path are created and routed. This continues until a packet contains only a single address in which case it is converted to a unicast packet and is routed to its destination.

Recently an experimental IP protocol for multi-destination multicast called explicit multicast (XCAST) protocol has been specified [2] and several XCAST testbeds have been deployed. He and Ammar [8] analyze the performance of XCAST.

3. Explicit multicast-enabled EpiChord

In EpiChord [1], peers maintain a full-routing table and approach 1-hop performance on DHT operations compared to the O(log N) hop performance of multi-hop overlays, at the cost of the increased routing table updates and storage.

An EpiChord peer's routing table is initialized when the peer joins the overlay by getting copies of the successor and predecessor peers' routing table. Thereafter, the peer adds new entries when a request comes from a peer not in the routing table, and removes entries which are considered dead. If the churn rate is sufficiently high compared to the rate at which lookups add new entries to the routing table, the peer sends probe messages to segments of the address space called slices. Slices are organized in exponentially increasing size as the address range moves away from the current peer's position. This leads to a concentration of routing table entries around the peer, which improves convergence of routing.

To improve the success of lookups, EpiChord uses *p-way* requests directed to peers nearest to the node. During periods of high churn, a peer maintains at least 2 active entries in each slice of its routing table. When the number of entries in a slice falls below 2, the peer issues parallel lookup messages to ids in the slice. Responses to these lookups are used to add entries to that slice in the routing table.

All parallel lookups in EpiChord are carried in separate unicast messages. In our design, we replace these parallel unicast messages with a single XCAST packet. This significantly reduces lookup message traffic for both edge links and internal links.

In addition, probe lookups for slice refresh can be aggregated into *p-way* XCAST messages. That is, during a stabilization cycle, there could be 10 slices that need lookups. These can all be combined in one XCAST message with 10*p addresses. We do not evaluate these savings in this paper.

4. Analysis of XCAST-enabled EpiChord

In this paper we evaluate through simulation the performance benefits of using multi-destination multicast to parallelize the EpiChord overlay. The original EpiChord simulator was extended to enable XCAST packet routing. The EpiChord simulator is implemented on SSFNet.

All simulations were run for both unicast and XCAST cases. Following [1], lookup and churn intensive workloads were used. These workloads are described later in this section. The underlay network contains 10,450-nodes consisting of 25 autonomous systems, each containing 13 routers and 405 hosts.

The simulation parameters are those described in [1] with regards to timeouts, node life span, stabilization intervals, cache entry life span and lookup frequency. Following He and Ammar [8], the cost of a multi-destination routing decision is comparable to that of a unicast routing decision, particularly for small address sets. We did not introduce any routing delay due to multi-destination routing in the simulation model.

4.1 Lookup intensive workload

In this workload, nodes join the network at a rate of 2 per second and issue on average 2 lookups per second. The overlay network grows until it reaches

1200 EpiChord nodes. Measurements made in [1] were repeated for both EpiChord and XCAST-enabled EpiChord with parallelism ranging from 1 to 5-way.

XCAST-enabled EpiChord performed equivalent to unicast EpiChord for average hop count, lookup latency and the success rates of lookups across all degrees of parallelism, thus retaining EpiChord's performance advantages over Chord.

In addition, XCAST significantly reduces message traffic for both overlay maintenance and DHT lookups. We evaluate this reduction for both edge and internal links. We define an edge link as a duplex link connecting a host and a router and an internal link as a router-router connection. Our results for the average number of messages per link follow.

4.2 Measurement of lookup messages per link in lookup intensive workload

Lookup messages are used by EpiChord for three purposes: *joins, maintenance* and *application.* Join messages are sent when a new node wishes to join the network and issues a *p-way* lookup message to its successor node and p-1 predecessor nodes. Maintenance lookup messages are sent when the routing table does not satisfying the required number of nodes per slice. Application lookups are standard lookups for some value in the DHT, issued twice per second per node.

As shown by comparing Figures 1 and 2, XCAST-enabled EpiChord reduces the number of application lookups per internal link by up to 30% for a 5-way mode versus unicast EpiChord. Similarly, Figure 3 and 4 show that using XCAST reduces the number of messages on the edge link by up to 31%.

In general, for a request-response protocol, replacing p unicast requests with 1 XCAST packet leads to a savings rate of (p-1)/(2*p) for an edge link, assuming all responses are returned as separate unicast packets. For p=5, the expected savings rate is 40%. Three factors account for the reduced savings and are explained in the next section.

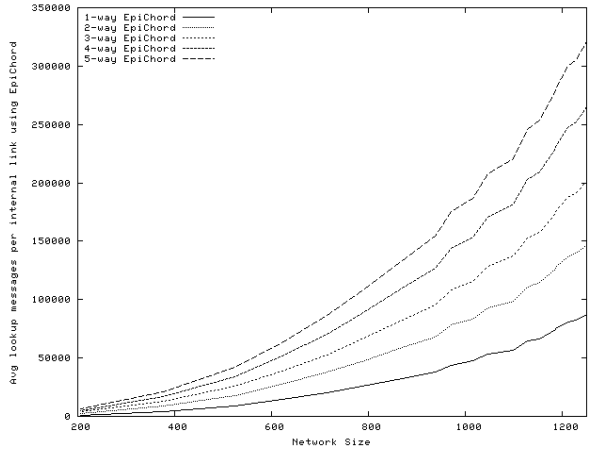

Figure 1: Average number of lookup messages on internal links for a growing network in a lookup intensive configuration using unicast EpiChord.

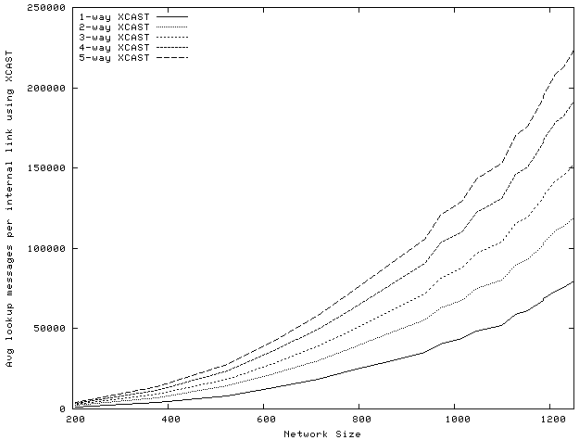

Figure 2: Average number of lookup messages on internal links for a growing network in a lookup intensive configuration using XCAST-enabled EpiChord.

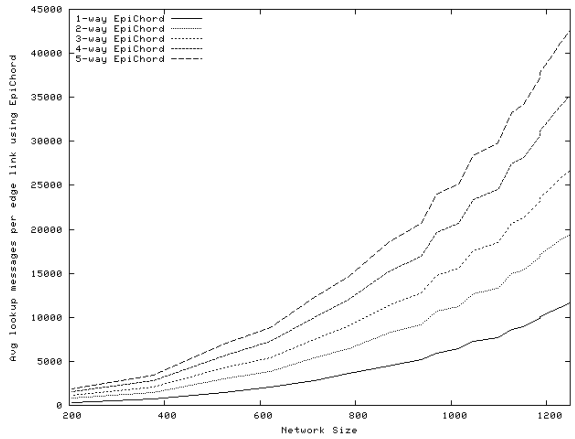

Figure 3: Average number of lookup messages on edge links for a growing network in a lookup intensive configuration using unicast EpiChord.

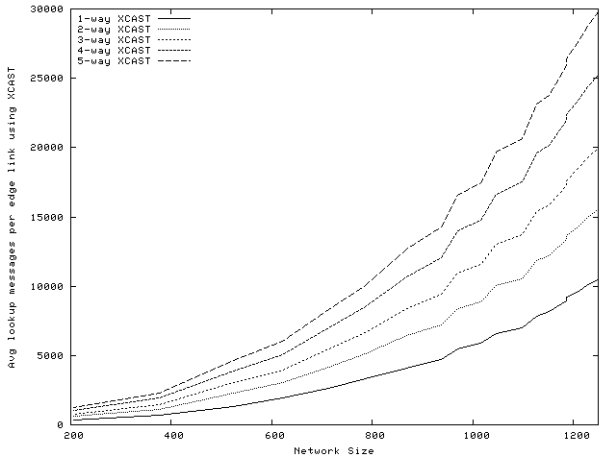

Figure 4: Average number of lookup messages on edge links for a growing network in a lookup intensive configuration using XCAST-enabled EpiChord.

For maintenance messages, the savings achieved are an average of up to 28% per internal link and 29% per edge link. Finally, the message reduction for join

messages using XCAST-enabled EpiChord is an average of up to 25% for 5-way for both edge and internal links.

4.3 Churn intensive workload

For the churn intensive workload, on average 15 nodes join the overlay network per second and issue on average one lookup every 10 seconds. Node lifespan is 550 seconds and the network grows in size continuously to 9000 nodes. All measurements in [1] were repeated for the churn intensive workload with additional measurements taken for lookup messages per link. As before, the XCAST-enabled EpiChord results for the average hop count, lookup latency and failure and timeout rates were consistent with unicast EpiChord.

4.4 Measurement of lookup messages per link in churn intensive workload

In a churn intensive workload, the savings on both the internal and edge links are somewhat reduced. As shown in Figures 5 to 8, XCAST-enabled EpiChord shows a reduction of up to 24 % on the edge link and 23% on the internal link for a 5-way simulation.

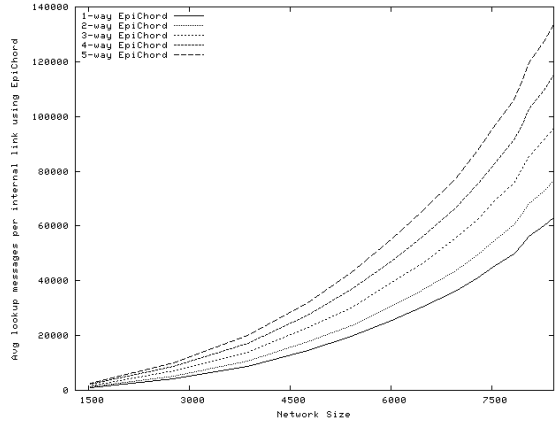

Figure 5: Average number of lookup messages on internal links for a growing network in a churn intensive configuration using unicast EpiChord.

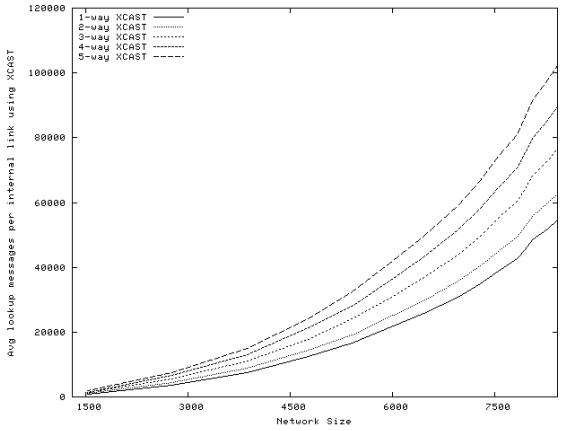

Figure 6: Average number of lookup messages on internal links for a growing network in a churn intensive configuration using XCAST-enabled EpiChord**.**

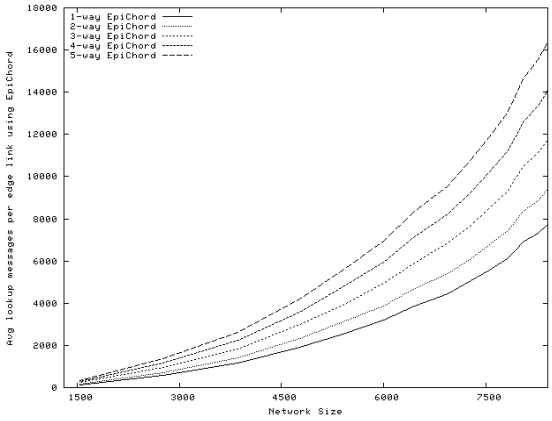

Figure 7: Average number of lookup messages on edge links for a growing network in a churn intensive configuration using unicast EpiChord.

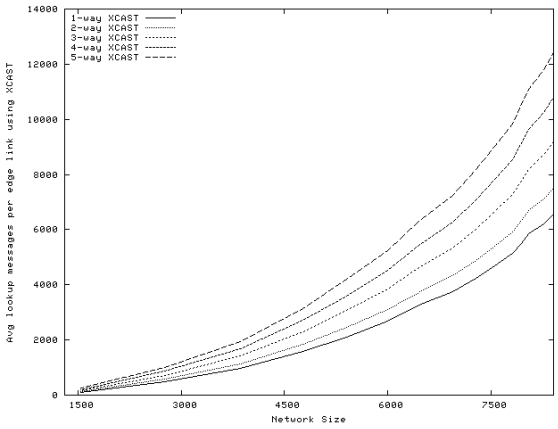

Figure 8: Average number of lookup messages on edge links for a growing network in a churn intensive configuration using XCAST-enabled EpiChord.

Maintenance shows a reduction of 25% on the edge link and 24% on the internal link. Finally, the reduction for application lookup messages on the edge link for join messages is 23% and 22% on the internal link, again for a 5-way simulation.

5. Analysis

5.1 Measured and expected savings

We have shown that XCAST-enabled EpiChord can reduce the number of lookup messages by up to 32% for edge links and 31% for internal links over standard EpiChord for a 5-way mode. So 3-way unicast EpiChord has the same message overhead as 5-way XCAST-enabled EpiChord. Next we discuss factors limiting these savings. These factors include lost messages, retransmissions, and negative responses to lookups. Due to these factors and the nature of the EpiChord lookup algorithm, 5-way mode actually results in a mix of n-way messages, $2 \leq n \leq 5$.

We discuss edge and internal links separately because edge links overhead is important for many users, such as those connecting through a home broadband connection or a mobile node's GPRS connection.

The quantitative benefits of multicasting have been formulated in the Chuang-Sirbu [9] scaling law which shows that the efficiency of multicast vs unicast is ~ $m^{.8}$ (where m is multicast group size). Further evaluation of Chuang-Sirbu has been done in [10] which derives another similar expression and confirms it with respect to various networks, and [11] which finds some shortcomings of Chuang-Sirbu with respect to large groups and provides a revised formulation. In our case, since the size of the group is small, then the Chuang-Sirbu formulation should be accurate predictor of the savings inside the network. So 5-way multicast would provide a 27% savings compared to unicast.

5.2 An example

When a lookup is initiated, the full saving of the XCAST packet is obtained on the local edge link, i.e., for *p-way* XCAST-enabled EpiChord, a saving of *p – 1* is achieved. As the packet traverses the network, it encounters points where its path is split. At these points the router clones the packet and sends the cloned packets along separate links. For every split that occurs, the saving per internal link is reduced. The earlier the split occurs in the lookup path determines the reduction of the potential saving per link. Thus, it can be argued that a greater saving per edge should be possible than on the internal link. However, as our results show, the saving per edge link is not significantly greater. With reference to Figure 9, we show an example of a 4-way XCAST transmission on two simple network configuration to highlight possible reasons for this.

In the diagrams, we denote edge links as dashed lines and the solid lines as internal links.

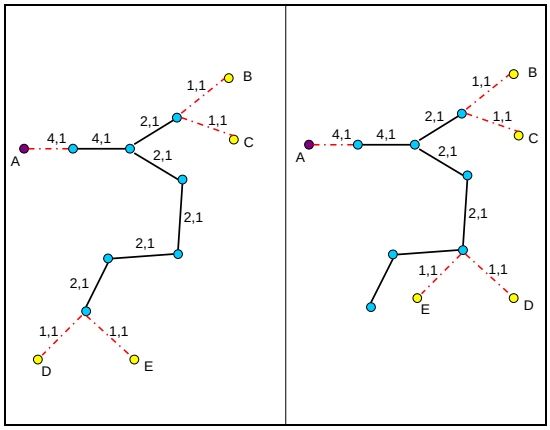

Figure 9: Two example network configurations with number of messages required for unicast EpiChord and XCAST-enabled EpiChord.

The comma-separated values indicate the cost in messages per link to send the lookup using XCAST (right value) and EpiChord (left value). With reference to the savings per edge link, we first examine both diagrams. Although the saving on the local (sending) edge link is a maximum 75%, the destination edge links receive no benefit from XCAST at all. Subsequently, the maximum saving on the edge link is (*1 – ((n+1)/2n)).* With regards to a 4-way lookup this becomes (*1 – (5/8) * 100*) = 37.5%. We now consider the internal links.

With reference to the network model on the left, we observe an early 2-way split of the original 4-way packet. No more splitting occurs until the packet reaches its final destination. Identical behavior occurs for the network model on the right but with a shorter path length. For an internal link, the maximum saving that can be gained is (*1 – 1/n)*. However the model on the left demonstrates a total saving of *(1 – (6/14) * 100)* = 57.1% and with a shorter path length, the model on the right demonstrates a saving of *(1- (4/10) * 100)* = 60%. This suggests that reducing the path length increases the message saving after a split [12].

5.3 Causes of reduced savings

As we have shown, the saving on the internal link is potentially much greater than that on the edge link. However, our results do not follow such a theory. We identify the following reasons for such behavior:

***Invalid routing table entries*:** In a large scale overlay, the routing table is likely to be out-of-date due to churn. Should a lookup message be sent to a node that is offline, the message will never reach the destination edge link (thus improving the edge link saving) and shall traverse the internal network until its TTL expires (thus reducing the internal link saving). As shown in Figure 21 in [1], the percentage of stale entries in the cache is around 13% for a steady state network of 1200 nodes. This suggests that 13% of all lookups should fail to reach their destination, thus demonstrating the above behavior. If this figure could be reduced, an improvement in the savings per link could be achieved.

Re-transmissions: When an individual message from a *p-way* lookup reaches timeout, the node will check the cache to determine if this node has reached timeout enough times to be considered dead. If not, it will retransmit a single UDP lookup message. After re-transmission, if the number of responses the sending node is waiting for *x* is (> 0 && <= *p)*, then it shall issue a (*p–x) way* lookup. Subsequently, for a *p-way* simulation, many lookups may be issued that demonstrate less than the allowed *p* degree of parallelism.

***Negative responses**:* When a node receives a lookup message, it will check to see if it is responsible for the requested item. If not, it shall respond with the *l* most likely nodes to try next. When the originating node receives this negative response, it shall add the *l* nodes to its routing table and then issue a new lookup message to the (*p – x)* next best nodes where *x* is the

number of responses it currently awaits. Again, as a high number of lookup responses will be negative, for a *p-way* simulation, a very high number of lookups may be issued that are less than the allowed *p* degree of parallelism.

Now, consider that for a 5-way XCAST lookup in a lookup intensive workload, only 21.5% of packets on the edge link and 18.2% on the internal link are actually XCAST packets. All other packets are UDP. This reduces the possible saving per link. Also, from Figure 10, we can see that of these messages, between 55-62% were in actual fact sent 2-way. Again, this will reduce the potential message saving per link. Large numbers of XCAST packets carrying two destinations stem from the number of re-transmissions and negative responses. As an XCAST message can carry at least two destinations, it is natural that this is the most common message.

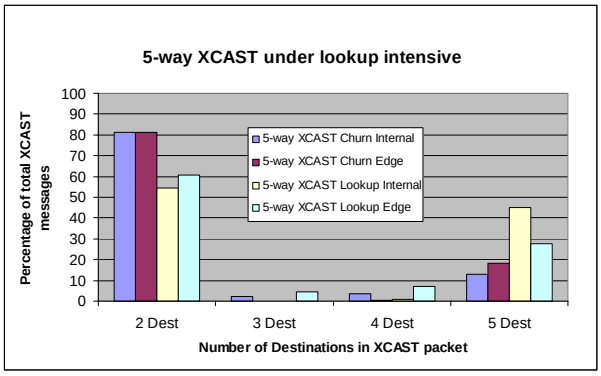

Figure 10: Numbers of XCAST messages, which contain 2, 3, 4, and 5 destination addresses in a 5-way XCAST-enabled EpiChord setup with a lookup intensive configuration.

In this section we have demonstrated that our 30% saving per link is a realistic value when stale cache entries, re-transmissions and negative responses have all been considered. As we discussed in Section 5.1, modeling a 5-way XCAST enabled network using the Chuang-Sirbu [9] scaling law should demonstrate a message saving of 27%. Clearly, this figure is close to our 30% but does not consider networking factors such as those discussed in this section. So the derivation may be slightly larger in practice.

This may be due to the fact that the Chuang-Sirbu scaling law is a heuristic model, and is easily affected by a number of variables. Previous work identifies the randomness of the multicast trees in the overlay, the underlay network topology and the nodal degree [13].

6. Conclusion

We have shown that parallelizing the EpiChord 1-hop overlay algorithm to use multi-destination multicasting instead of parallel unicast lookups benefits in significantly reduced message traffic on both edge and internal links. Message reduction occurs for EpiChord messaging for joins, routing table maintenance, and application lookups. The reduction for 5-way EpiChord is about 30%. The EpiChord latency behavior and operational semantics are retained.

We identified three factors that limit the gains of message reduction. One of these (invalid routing table) seems to be a general issue not specific to EpiChord, while the other two (re-transmissions and negative responses) are somewhat associated with EpiChord itself. We described but did not simulate possible further parallelization of the EpiChord routing table maintenance that could lead to further gains.

7. Acknowledgement

The EpiChord simulator based on SSFNet and developed by Ben Leong at MIT was kindly provided to us.

8. References

[1] Ben Leong, Barbara Liskov, and Erik D. Demaine. EpiChord: Parallelizing the Chord Lookup Algorithm with Reactive Routing State Management. Computer Communications, Elsevier Science, Vol. 29, pp. 1243-1259.

[2] R. Boivie, N. Feldman , Y. Imai , W. Livens , D. Ooms, O. Paridaens, E. Muramoto, Explicit Multicast (Xcast) Basic Specification, draft-ooms-xcast-basic-spec-09.txt, Work in Progress. Dec. 2005.

[3] L. Aguilar, Datagram Routing for Internet Multicasting, Sigcomm 84, March 1984.

[4] Mostafa Ammar. Why Johnny Can't Multicast: Lessons about the Evolution of the Internet. Keynote - NOSDAV 03.

[5] J. Buford, A. Brown, M. Kolberg. Multi-Destination Routing and the Design of Peer-to-Peer Overlays. In preparation.

[6] Tetsuya Oh-ishi, Koji Sakai, Hiroaki Matsumura, Akira Kurokawa, Architecture for a Peer-to-peer Network with IP Multicasting," 18th Intern. Conf. on Advanced Information Networking and Applications (AINA'04) Vol. 2, 2004.

[7] Tetsuya Oh-ishi, Koji Sakai, Kazuhiro Kikuma, and Akira Kurokawa. Study of the Relationship between Peer-to-Peer Systems and IP Multicasting. IEEE Communications Magazine. Jan. 2003.

[8] Qi He, Mostafa Ammar. Dynamic Host-Group/Multi-Destination Routing for Multicast Sessions. J. of Telecommunication Systems, vol. 28, pp. 409-433, 2005.

[9] Chuang, J., and Sirbu, M. Pricing multicast communications: A cost-based approach. In Proceedings of the INET'98 (1998).

[10] G. Phillips, S. Shenker, and H. Tangmunarunkit. Scaling of multicast trees: Comments on the Chuang–Sirbu scaling law. ACM SIGCOMM'99.

[11] Van Mieghem, P., Hooghiemstra, G., and van der Hofstad, R. 2001. On the efficiency of multicast. IEEE/ACM Trans. Netw. 9, 6 (Dec. 2001), 719-732.

[12] Chalmers, R.C, Almeroth, K.C, Developing a Multicast Metric, Proceedings of the IEEE Global Internet (GLOBECOM '00), California, 2000.

[13] L. Lao, J.-H. Cui, and M. Gerla. A Framework for Realistic and Systematic Multicast Performance Evaluation. Computer Networks, Volume 50, Issue 12, Pages 2054-2070.