Exploiting parallelism in the design of peer-to-peer overlays
John Buford, Alan Brown and Mario Kolberg

*Computer Communications*
Volume 31, Issue 3, 25 February 2008, Pages 452-463

# Exploiting Parallelism in the Design of Peer-to-Peer Overlays

John Buford
*Avaya Research Laboratory*
*USA*
*buford@samrg.org*

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*
phone: +44 1786 46 7440
fax: +44 1786 46 4551
(Corresponding Author)

## Abstract

*Many peer-to-peer overlay operations are inherently parallel and this parallelism can be exploited by using multi-destination multicast routing, resulting in significant message reduction in the underlying network. We propose criteria for assessing when multicast routing can effectively be used, and compare multi-destination multicast and host group multicast using these criteria. We show that the assumptions underlying the Chuang-Sirbu multicast scaling law are valid in large-scale peer-to-peer overlays, and thus Chuang-Sirbu is suitable for estimating the message reduction when replacing unicast overlay messages with multicast messages. Using simulation, we evaluate message savings in two overlay algorithms when multi-destination multicast routing is used in place of unicast messages. We further describe parallelism in a range of overlay algorithms including multi-hop, variable-hop, load-balancing, random walk, and measurement overlay.*

**Keywords:** peer-to-peer overlay, multi-destination multicast routing, distributed hash table

# 1 Introduction

We are interested in improving the performance of peer-to-peer overlays [1] by mapping overlay messaging to native multicast paths for overlay operations that are inherently parallel. This technique is generally applicable to overlay design, and in this paper we give examples from existing structured and unstructured overlays, for one-hop, multi-hop, and variable hop overlays. We show the applicability to overlay mechanisms for measurement, load-balancing, and maintenance. This technique can be used by explicitly specifying multicast messages for parallel overlay operations, or as discussed later the outgoing message queue can be filtered to identify multicast combinations of outgoing messages.

It is well known that a significant percentage of internet traffic is due to peer-to-peer applications. Using the Chuang-Sirbu [2] multicast scaling law where the multicast versus unicast savings $1\text{-}m^{\varepsilon}$ is dependent on the group size m, then 3-way multicast saves about 20% to 30% of messages compared to unicast transmission, and 15-way multicast saves about 40% to 60% of messages. As is discussed in Section 3, the value for ε may differ between -0.2 and -0.34 depending on the topology of the network.

Thus, message traffic reduction is a practical issue both for efficiency of the internet, and for the operation of a specific peer-to-peer overlay. Alternately, for the same level of message traffic, multicast delivery can offer increased parallelism for an overlay operation.

This paper contains the following contributions:

- We formulate criteria for determining whether overlay messages can be parallelized using multicast. These criteria are maximum group size, number of groups, the time to create a new multicast group, and group formation rate.
- We show how multi-destination routing can be used in several categories of overlays for various overlay operations including DHT operations, overlay maintenance, replication, and measurement.
- We show multicast savings for two overlay algorithms based on simulation results.

- We show that the assumptions of the Chuang-Sirbu multicast scaling law are consistent with large-scale peer-to-peer overlays, and estimate the message savings for a number of peer-to-peer overlays.

The remainder of this paper is organized as follows. We formulate the criteria for evaluating the use of multicast messaging in the next section, followed by a review of related work in Section 3. Section 4 analyzes a one-hop overlay and Section 5 analyzes an overlay maintenance algorithm. Section 6 describes and analyzes multi-destination routing in several other overlay categories and message operations. Section 7 concludes the paper.

# 2 Problem statement

## 2.1 Criteria for multicast messaging

The typical use of a peer-to-peer overlay is to provide widely available end-to-end network services that would be difficult to deploy inside the network, or to share computing resources among a large set of users. A variety of peer-to-peer overlays have been developed for applications such as file sharing, instant messaging, and telephony. To explain the potential for using multicast messaging in an overlay, first consider the various types of messages that are possible:

1. A node sends a discovery message to another node during overlay bootstrap.
2. A node announces it is joining or leaving the overlay.
3. A node sends routing tables or routing table excerpts to another node.
4. A node sends node metrics and/or overlay measurements to another node.
5. A node sends a DHT lookup request or response to another node.
6. A node sends an event to one or more subscriber nodes.
7. A node sends its application state to another node, for example, for replication for load distribution and reliability.
8. A node acting as an inter-overlay gateway translates a message from one overlay to another.
9. A node sends a measurement probe, heart-beat or gossip message to another node.
10. A node sends messages which combine or concatenate other types of messages, such as exchanging routing table updates with DHT request or response messages.
11. A node sends messages to create, join, use, manage, and leave application layer multicast (ALM) tree using an overlay routing algorithm.
12. A node sends an acknowledgement to any of the above messages.

Let P be the set of peers in the overlay during some interval T, where $|P| = n$. Let M be the set of overlay protocol message types for the overlay, $M_t \in M$ is one of the above message types, and $m_j$ is a message instance of a given type $M_t$, with j as a unique identifier for each message instance in interval T. Define $F_i$ as the set of all combinations of P of sizes i = 2, 3, … n. We distinguish multicast messages where the destinations all receive the same message type $M_t$ from multiplexed multicast messages where destinations receive multiple possibly overlapping message types (such as probe and lookup messages). In general, the former offer the most efficiency gains since they frequently share message payload. The cases we identify for specific existing overlay designs later in this paper are single-type multicast messages.

During interval T, the accumulated count of messages sent by all peers in P is $A = \Sigma a_t$ where each $a_t$ is the count of those messages of type $M_t$. These messages are sent over network links L. For unicast transmission, the average number of messages per link is then $A/|L|$. The goal is to replace sets of unicast messages to multiple destinations from the same peer with a multicast message to the same destinations, reducing the per link message traffic to $A'/|L|$, $A' < A$. The number of unicast messages that can be combined depends on the message type, the degree of parallelism of that message type in a given overlay, and their temporal locality.

To identify temporal locality needed for using a multicast message in place of unicast messages, we construct an outgoing message queue at each peer. Each peer p has a queue Q which has pending messages $m_j$ to send. After adding a message $m_j$ to Q, the peer examines Q and may combine a set u of messages in

Q to create a new multicast message $m_c$ to group $g_k$ where $m_c$ contains the contents of the individual u messages, $p \notin g_k$, $|g_k| = |u|$, $g_k \in F_{|u|}$, and k is a unique group identifier. The peer may flush one or more messages from Q, combine other unicast and multicast messages in the queue, or wait for further messages. The peer acts to maintain the maximum queueing delay of any message below a threshhold $d_q$. Assume peers agree on the rules for combining and extracting unicast messages to and from multicast messages. Assume further that the decision process by which messages are combined considers that the benefit of multicast for network efficiency is proportional to the amount of overlap of the content of the combined unicast messages.

The number of multicast groups used in the overlay in the interval T is then $N_G = |G|$ where $G = \{g_i : \forall i\}$. The maximum group size is $|g_{max}|$ such that $\forall\ g_k : |g_k| \leq |g_{max}|$ and $\exists\ g_k : |g_k| = |g_{max}|$. The rate of group formation is $r = N_G/T$ and the frequency of group use $f(g_k) = |m_c|/T$, the number of multicast messages $m_c$ to the group $g_k$ in interval T.

For example, in Table 1 we show example values for two one-hop overlays which we describe in further detail later in the paper. EpiChord (see Section 4) uses p-way lookups for both application level and maintenance messages, and the number of groups needed during T and the rate of group formation are directly related to the lookup rate $r_L$. Although group size is small, for a high-churn overlay the rate of group formation might be as high as 1 group per second per peer, and the groups have low frequency of re-use due to the random distribution of access in the overlay.

D1HT together with EDRA (see section 5) uses up to (log n)-way messages, and its rate of group formation is directly related to the churn rate. A high churn network is considered to have a mean node lifetime of one hour. The frequency of group use is related to the EDRA event maintenance interval $\theta$, which decreases as the churn rate increases.

**Table 1 Multicast parameters for two O(1) overlays.**

| Parameter | 5-way EpiChord | D1HT with EDRA* |
|---|---|---|
| $\|g_{max}\|$ | 5 | log (N) |
| $N_G$ | $r_L * T$ | (log (N) – 1) * N |
| $r = N_G / T$ | $r_L$ | N * (log (N) – 1) / T |
| $f(g_k)$ | 1/T | ~ $1/\theta$ |

Multicast routing offers efficiency and concurrency to overlay designers. It may improve response time for operations in which parallelism locates a shorter path more quickly. Reliability may be affected, since a lost multicast packet effects multiple messages. For multicast to be practical it is necessary that:

1. The scalability of the multicast algorithm for number of groups meets the scalability requirements of the overlay. If C is the capacity of the network to support simultaneous multicast group state for this overlay, then $N_G \leq C$. Likewise, if v is the maximum group size supported by the network, then $|g_{max}| < v$.
2. The overlay's rate r of group formation and group membership change must be sustainable by the multicast mechanism. The time to create a new multicast group $t_c < d_q$.

Overlays maintain their own routing state. It is preferable that group join in the multicast protocol leverages the overlay formation methods and routing state, to avoid redundancy between network layer group management protocols and overlay routing table management.

## 2.2 Host-group multicast

The prevailing host-group multicast protocols including PIM, DVRMP, and CBT create a group address per multicast tree, and each router stores state for each active group address. The state in the router grows with the number of simultaneous multicast groups. Further, there is delay to create a group, and the network may support a limited number of group addresses.

For a large overlay it is impractical for each node to have a group address for each set of other nodes it sends multicast messages to. Suppose N = 1M, T = 60 minutes, and each peer conservatively uses 5 groups for those $m \in M$ it parallelizes, so $N_G = 5$ N. Worsening the problem is that the average peer session time is as low as 60 minutes in some overlays, meaning that group state is replaced relatively frequently.

Host group multicast is designed for relatively small numbers of very large sets of recipients. So host group multicast is not a good choice for use in parallelizing network overlay operations where there are many simultaneous small groups of peers involved in a message, and the groups are short-lived.

## 2.3 Multi-destination multicast

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 is highly scalable in terms of possible numbers of groups. Hence multi destination routing has considerable advantages for applications which feature large numbers of small groups. It does not pose a bottleneck for overlay networks even with millions of nodes.

In multi-destination multicast routing, instead of two or more unicast messages sent to separate destinations, a single message is sent containing the list of the destinations and the message content from the original messages. Multicast-enabled routers route the message until a split point is reached (according to unicast routing decisions). At each such point, duplicate messages containing the subset of destinations for each forwarding path are created and routed. This continues until a message contains only a single address in which case it is converted to a unicast message and is routed to its destination.

If only a subset of all routers are multicast-enabled, these routers forward multicast packets to other multicast routers using tunnels through unicast routers. Alternately, the network contains hosts which implement the multi-destination multicast routing. Overlay nodes sending a multi-destination multicast message send the message to one such host. The host routes the multi-destination messages to other hosts according to the network routing rules. At each such host, duplicate messages containing the subset of destinations for each forwarding path are created and routed. This continues until a message contains only a single address in which case it is converted to a unicast message and is routed to its destination. Figure 1 illustrates this. Here node A is sending a multi-destination multicast message to 4 nodes B, C, D and E. Routers where a split occurs are shown by filled circles. The numbers indicate the count of destination addresses in the message.

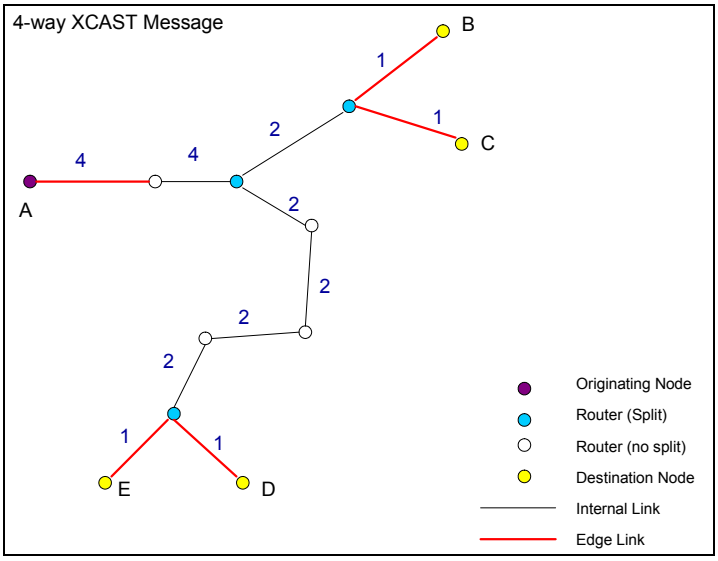



**Figure 1: Progress of a multi-destination multicast message through a network.**

Multi-destination multicast routing does not require state in routers. Thus there is no router state constraint on $N_G$. However there is additional processing overhead at each router for the forwarding algorithm to process the list of addresses. Whereas host-group multicast routers have forwarding state for each group address, for a multi-destination packet with N destinations, there is O(N) work at each router to process the list of addresses and make a forwarding decision for each destination. Packet duplication work for multi-destination routing is similar to that of host-group multicast.

Multi-destination multicast imposes a maximum group size v. Practical values for v appear to be less than 50. Since peers in the overlay maintain routing tables or addresses of other peers, there is no group join overhead when peers are directly reachable in the overlay. Thus the time to create a new multicast group $t_c$ is not a factor.

Recently an experimental IP protocol for multi-destination multicast called explicit multicast (XCAST) protocol has been specified [5] and several XCAST testbeds have been deployed. An important operational consideration is that even if XCAST is partially deployed, then the technique still works for those regions of the overlay which have access to XCAST capability. He and Ammar [6] analyze the performance of XCAST combined with host-group multicasting. For brevity in the remainder of this paper we will use XCAST as a representative multi-destination multicast protocol.

### 2.4 Selection and integration

A peer joins an overlay and through a combination of configuration and discovery determines whether a multicast mechanism is available in the underlay. Depending on how many different message types M are used in the overlay and the inherent parallelism of the associated operations, the overlay itself can issue those messages as multicast messages. In addition, as described in Section 1, outgoing messages can be queued and messages containing overlapping content can be combined.

## 3 Related work

A large number of peer-to-peer overlays have been proposed, and overlay design continues to be an active area of research. Lua et al. [7] survey many contemporary overlays but do not discuss multicast-enabled overlays. Aberer et al. [8] present an overlay design space but also do not address multicast-enabled overlays. Identifying and analyzing parallelism in overlay messaging has not been generally addressed. Use of an overlay for application layer multicasting (ALM) is a separate and different issue, and existing ALMs use unicast messaging at the network layer.

Kelips [9] is a O(1)-hop overlay which uses an epidemic multicast protocol for exchanging overlay membership and other soft state between nodes. As described in [10], such a protocol consists of two sub-protocols: a multicast data dissemination protocol and a gossip protocol to exchange message history for reliability purposes. In [10] the multicast data dissemination protocol can be either IP multicast if available or random spanning trees over the multicast group formed by unicast connections. The Kelips epidemic multicast heartbeat protocol which maintains the soft state in each node gossips to nodes in its own group and to contacts in other groups. Each group size is $< \sqrt{n}$, and each node has two multicast groups in a given gossip round. Intra-group targets for gossip are weighted towards neighbors, so intra-group multicast groups are repeated in subsequent rounds until neighbors change. Inter-group targets vary round by round, so inter-group multicast group membership changes round by round at each node. Consequently, Kelips requires in a given gossip round O(n) number of multicast groups, and group membership for each group changes each round. The epidemic protocol based on [10] used by Kelips does not scale in terms of router state if IP multicast is used and has substantial group membership maintenance overhead.

Oh-ishi et al. have considered the use of Protocol Independent Multicast (PIM) [11] in sparse mode (PIM-SM) and source specific mode (PIM-SSM) [12] to reduce message traffic in peer-to-peer systems. Their analysis focuses on using multicast routes between peers in different ISP networks.

He and Ammar [6] analyze the performance of XCAST combined with host-group multicasting, where XCAST is used for small groups and host-group multicasting is used for large groups. For XCAST sessions they use a dynamic tunneling mechanism between routers corresponding to XCAST branch points in a given session. Since most routers in a multicast path are non-branching, the XCAST routing processing in each router is significantly reduced. However this mechanism is session-oriented and would not be useful for parallelizing overlay operations which use short-lived groups.

The quantitative benefits of multicasting have been formulated in the Chuang-Sirbu [2] scaling law which shows that the ratio of the number of edges in the multicast versus the average unicast path length equals $m^{0.8}$ (where m is the multicast group size). The per link reduction in message traffic from multicast is then $1-m^{-0.2}$ . The exponent $\varepsilon = -0.2$ represents the multicast efficiency. The assumptions of the Chuang-Sirbu law are:

1. Receivers are randomly distributed throughout the network.

2. Multicast trees are shortest-path trees constructed using Dijkstra's algorithm and extend only to the edge router.
3. There is no control overhead or control overhead is <1% of total traffic.
4. It applies to networks that are representative of inter-domain routing topologies of the Internet, with average node degree in the range of 3 to 4, and a range of sizes from 47 routers to 5000 routers.
5. The multicast routing at the network layer does not support any reliability mechanism.

The first assumption relates to the placement of peers in the network and the distribution of the set of peers in a parallelized overlay operation. In overlay analysis, it is generally assumed that peers are randomly distributed through out the network. Except for proximity-aware overlays, neighbors in the overlay are not close to each other in the network and are unlikely to cluster on the same subnet. As a result, parallel overlay operations which are directed at a set of adjacent overlay nodes, such as a parallel lookups, are sent to a random set of network locations. In the case of proximity-aware overlays, the multicast savings can be no worse than predicted by Chuang-Sirbu and may be better due to the sharing of the tree path. The remaining assumptions relate to the network and the multicast mechanism. Using multi-destination routing satisifies assumptions 2, 3, and 5. The networks of interest satisfy assumption 4.

Further evaluation of Chuang-Sirbu has been done in [13] which derives another similar expression and confirms it with respect to various networks, and [14] which finds some shortcomings of Chuang-Sirbu with respect to large groups and provides a revised formulation. Chalmers and Almeroth [15] using actual multicast data sets on the Internet and synthesized multicast trees find a slightly lower multicast efficiency $\varepsilon$ in the range $-0.34 < \varepsilon < -0.30$. These results are based on actual multicast infrastructure in place at the time of the data collection which may constrain multicast branching points more so than in synthetic topologies. Further, their analysis includes multicast trees extended to the end points, which produces increased savings from Chuang-Sirbu if there is clustering of end points at subnets.

In the case of parallelized overlay operations discussed here, the size of the multicast group is within the Chuang-Sirbu formulation but is frequently at the lower end of the formulation, $m < 20$. In this range, the multicast efficiency exponent derived by Chalmers and Almeroth in [15] is also applicable. For brevity, in this paper we refer to multicast savings $1-m^{\varepsilon}$ with $-0.34 < \varepsilon < -0.20$ as the Chuang-Sirbu law. In comparison, Fahmy and Kown [16] give a range of $-0.4 < \varepsilon < -0.2$ for IP multicast efficiency, but do not present experimental results to substantiate the larger range. Figure 2 shows message savings versus group size for a set of values of $\varepsilon$. Even though the graph shows the savings for group sizes up to 20 to illustrate the potential, for the work reported in this paper the typical groups size is up to 10 destinations.

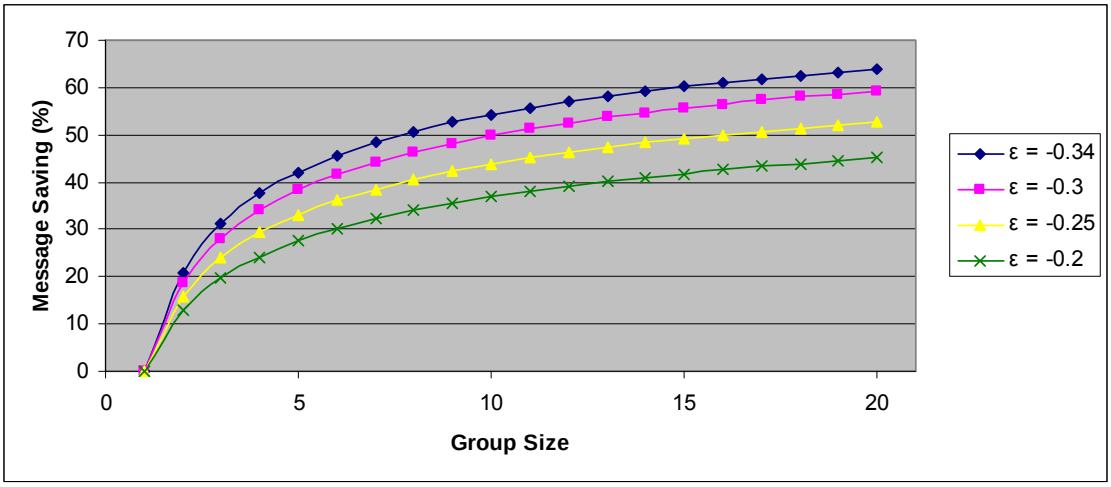



**Figure 2: Message savings versus group size for Chuang-Sirbu, varying ε.**

# 4 Parallelism in a one-hop overlay with opportunistic stabilization

In one-hop overlays, peers maintain a full-routing table and approach O(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. Because of the increased message overhead of one-hop overlays compared to multi-hop overlays, potential message savings from parallelism is particularly important.

Example one-hop overlays include EpiChord [17], OneHop [18], Kelips [9], Structured SuperPeers and D1HT [19]. Compared to EpiChord, these other systems require constant background traffic for routing table maintenance, and in the case of Structured SuperPeers place asymmetric loading requirement on a subset of the peers. In this section we analyze EpiChord, an important one-hop overlay that uses an opportunistic routing table maintenance algorithm, and in the next section we analyze parallelism in D1HT, a one-hop overlay that uses the EDRA active stabilization routing table maintenance algorithm.

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.

Figure 3 shows the detailed model of EpiChord's request and probe mechanism. For p-way request, p requests are initially sent and are placed in the pending queue. For any peer, after a first or second timeout, a unicast request is resent to that peer. After the third timeout or if a peer responds with a NAK, the peer is removed from the pending queue. If the pending queue is of size p-1, two new peers are sent a 2-way requests and are added to the pending queue, making its size p+1. Lookup terminates when an ACK is received. One consequence of this algorithm is that the amount of parallelism in practice is not simply p due to the unicast retransmissions and 2-way messages used by EpiChord to recover from timeouts and negative responses. Further, the proportion of unicast retransmissions and 2-way messages are dependent on the churn rate and the level of routing table accuracy.

If parallel unicast lookup messages and slice refresh messages are replaced with a single multi-destination packet [20], this can reduce the number of lookup messages by up to 32% for edge links and 31% for internal links over 5-way unicast. Alternately, for a given message load, a higher routing table accuracy can be obtained. Note that p-way EpiChord results in parallel message traffic that is on average less than p-way due to invalid routing table entries, re-transmissions, and negative acknowledgements [20].

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 next compare unicast EpiChord with XCAST-enabled EpiChord for lookup intensive and churn intensive workloads. Simulations were carried out on a 10,450 node underlay network using the SSFNet simulation environment [32].

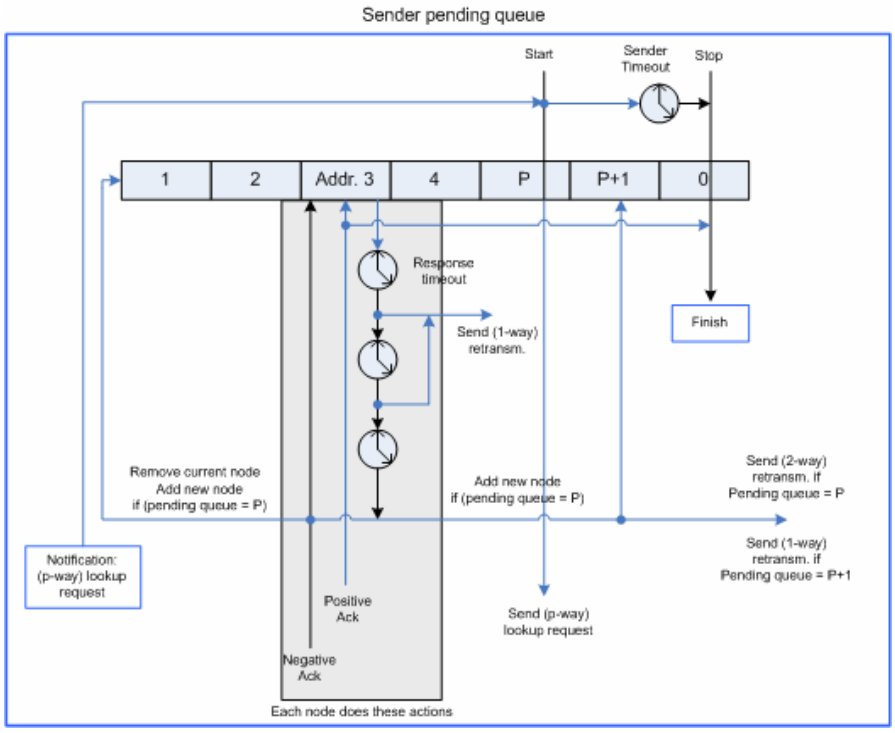



**Figure 3: EpiChord pending request queue states and actions.**

## 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. 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. During the simulation, the network grows from 200 nodes to 1200 nodes. We count all message traffic on each link in regular fixed sample intervals.

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 in Figure 4 (left), 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 4 (right) shows 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.

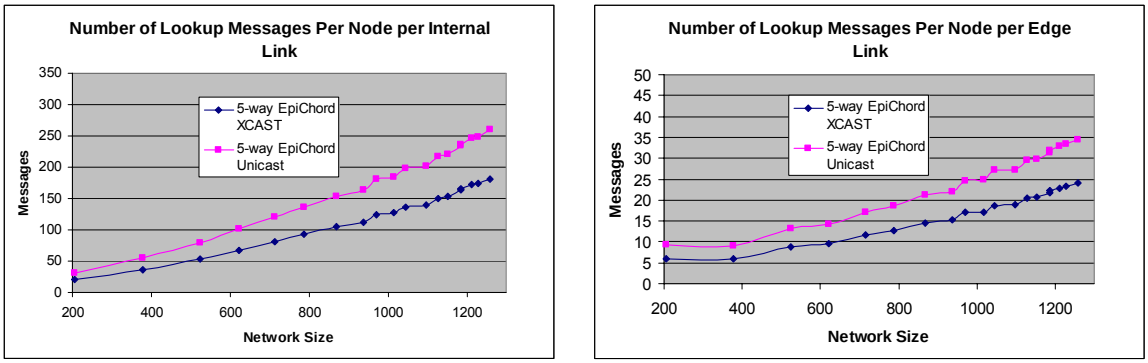



**Figure 4: Average number of lookup messages per node on internal links (left) and edge links (right) for lookup intensive workload.**

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.

## 4.2 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 [17] were repeated for the churn intensive workload with measurements taken for the 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.

In a churn intensive workload, the savings on both the internal and edge links are somewhat reduced. The primary reason for this is that increased churn causes lower routing table accuracy, leading to a higher percentage of unicast retransmissions and 2-way requests, reducing the amount of parallelism compared to the lookup intensive workload. As shown in Figure 5 (left), XCAST-enabled EpiChord shows a reduction of up to 24 % on edge links and in Figure 5 (right) a 23% reduction on internal links for a 5-way simulation.

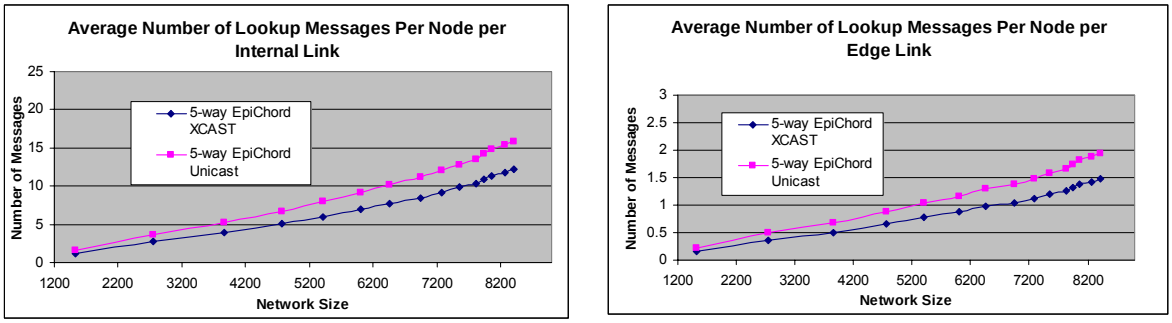



**Figure 5: Average number of lookup messages per node on internal links (left) and edge links (right) for a churn intensive workload.**

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

### 4.3 Measured and expected savings

We have shown that, depending on the workload, 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. Factors limiting these savings include lost messages, retransmissions, and negative responses to lookups. Due to these factors and the nature of the EpiChord lookup algorithm, the 5-way mode actually results in a combation of 5-way, 2-way, and unicast messages. Separately we have developed a Markov model of EpiChord [23] which confirms our simulation results and shows that when including the EpiChord retransmission and timeout related unicast messages, the expected savings correlate with Chuang-Sirbu for $\varepsilon = 0.7$.

## 5 Analysis of a one-hop overlay active stabilization algorithm

D1HT [19] is a one-hop overlay that uses an active stabilization overlay maintenance algorithm called EDRA (Event Detection and Reporting Algorithm), where an event is any join/leave action. EDRA propagates all events throughout the system in logarithmic time. Each join/leave event is forwarded to $\log_2(x)$ successor peers at relative positions $\log_2(0)$ through $\log_2(n)$ (Figure 6).

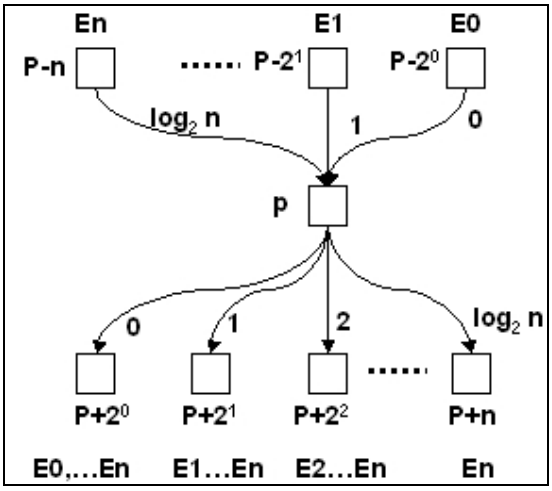



**Figure 6: EDRA event propagation as a multicast tree.**

Following the notation of [19], $\Theta$ is the interval at which a peer propagates events to its successors in the ring, and $\rho = \lceil \log_2 n \rceil$ is the maximum number of messages a peer sends in the interval. Propagated events are those directly received as well as those received from predecessors since the last event message. Each message has a time to live (TTL) and is acknowledged. If there are no events to report, only messages with TTL=0 are sent. The EDRA algorithm proposes that every event is reported only once at each node in the overlay.

Separately we have analyzed EDRA [22] and identified a number of corrections to the algorithm; we refer to this version of EDRA as EDRA*. These changes do not affect the message parallelism described next. During any interval $\Theta$, a peer sends at most $\rho = \lceil \log_2 n \rceil$ messages containing its current observed and propagated events. Each message contains the same set of events but different TTL in the range (0.. $\rho$). During any interval $\Theta$, we replace the $2 \le \rho' \le \rho$ unicast messages with a $\rho'$-way multi-destination packet containing the set of events and a list of [peer,TTL] pairs. Each peer receiving the multicast message extracts its own TTL from the list to determine which events in the message belong to them.

In our simulation environment described below, replacing unicast propagation messages with a single multi-destination packet obtains propagation message savings of around 35% per link. This saving depends on the distribution of values for $\rho'$, with a range of $2 \le \rho' \le \lceil \log_2 n \rceil$. The distribution of values for $\rho'$ are in turn dependent on the churn rate. For a high churn rate, each peer will receive more events to propage with different TTL values, increasing $\rho'$ up to the maximum value of $\lceil \log_2 n \rceil$. Thus the Chuang-Sirbu

value for messages savings at ρ-way multicast is an upper bound on the potential savings when using multi-destination routing with EDRA*.

Next we describe our simulation environment and results when unicast propagation messages are replaced by multi-destination propagation messages.

## 5.1 EDRA* with multi-destination messaging

We implemented the EDRA* maintenance algorithm using the SSFNet simulation environment [32]. Simulations were carried out on a 10,450 node underlay network consisting of 25 autonomous systems, each containing 13 routers and 405 hosts. The overlay network is grown to 1,200 nodes to reaach steady state. Node lifespans are distributed using a heavy-tailed Pareto distribution of $\alpha$=1, $\beta$=1800 and lookups are issued on average every 10 minutes per node. A median lifespan of one hour is achieved using this distribution and is consistent with past studies, as described in [26].

The EDRA* algorithm can be dynamically tuned to calculate the value of $\Theta$ required to satisfy the maximum fraction of routing failures *f*. As shown in Table 2, the message saving ranges from 33% to 37%.

**Table 2: Percentage message saving for multi-destination multicast routed EDRA*.**

| Percentage Saving for XCAST over Unicast EDRA* | | |
|---|---|---|
| **Parameters** | **Edge Link** | **Internal Link** |
| EDRA*, F=10 | 33.1% | 35.1% |
| EDRA*, F=30 | 33.5% | 35.9% |
| EDRA*, F=50 | 33.6% | 37.0% |
| EDRA*, F=70 | 33.1% | 35.7% |

In Figure 7 we show the bandwidth required to satisfy various levels of *f*. When calculating bandwidth usage we consider all messages which directly result in routing table updates. That is, for EDRA* we consider lookup messages, propagation messages and all replies. Also, proxy messages for new nodes in the join interval and any probe message which carries a routing table entry are included. From Figure 7 we show unicast EDRA* operates in the range of 35-55 bytes per second per node to maintain levels of *f*=10-70. XCAST EDRA* operates using significantly less bandwidth, in the region of 10-18 bytes per second per node.

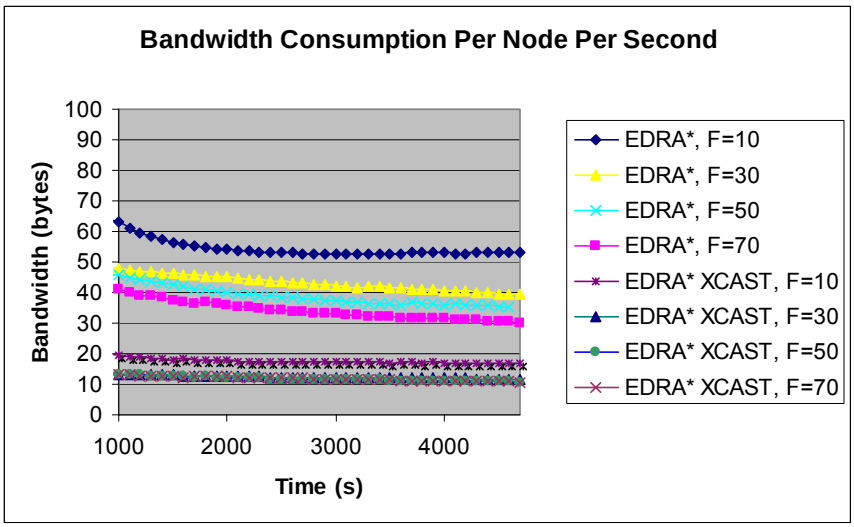



**Figure 7: Bandwidth consumption for EDRA* unicast vs EDRA* XCAST for varying degrees of f.**

The bandwidth requirements are further shown in Figure 8. Here we show the relation between Hop Counts for lookup messages and Bandwidth requirements for routing table maintenance using EDRA* with unicast and XCAST.

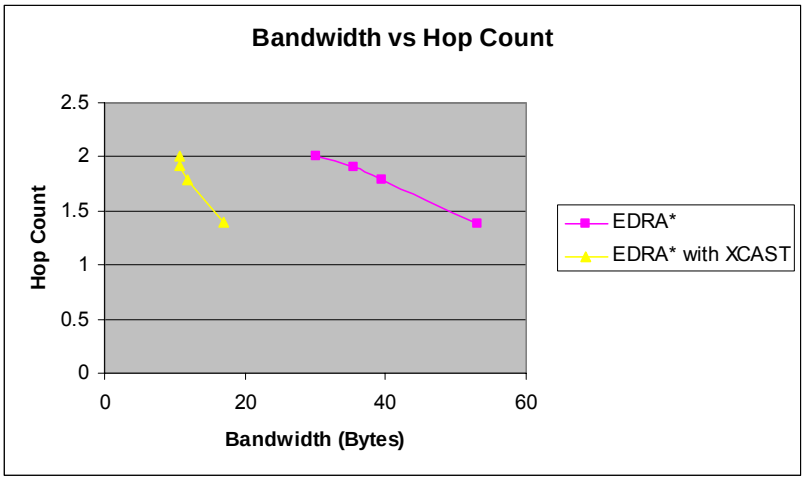



**Figure 8: Bandwidth vs Hop Count measurements for XCAST enabled EDRA* vs unicast EDRA* and Unicast EpiChord.**

# 6 Parallelism in overlay operation

In this section we illustrate the generality of applying multi-destination addressing to a variety of overlay designs. While many overlay algorithms are not designed to take advantage of parallel messaging using any type of multicast routing, these examples are taken from a broad range of overlay techniques and thus illustrate the generality of this method [21].

Generally, the parallelism in the overlay can be decided at design-time and at run-time. At design time, the overlay algorithms are analyzed manually to identify operations which can be parallelized. At run-time, the identification of parallel operations identifies messages which have a common payload within a time frame.

## 6.1 Kademlia – multi-hop overlay

Kademlia [24] is a multi-hop overlay that by virtue of its symmetric distance metric (the XOR function) is able to issue parallel requests for its routing table maintenance, lookups and puts.

During a node lookup, a peer computes the XOR distance to the node, looks in the corresponding k-bucket to select the $\alpha$-closest nodes that it knows of already, and transmits parallel requests to these peers. Responses return closer nodes. Kademlia iteratively sends additional parallel requests to the $\alpha$-closest nodes until it has received responses from the k-closest nodes it has seen. A typical value of $\alpha$ is 3. Figure 9 shows a node lookup for a node in the 110 k-bucket. For a 160-bit address space there will be up to 160 buckets.

Node lookup is used by other Kademlia operations including DHT store, DHT refresh, and DHT lookup. A Kademlia peer does at least $k/\alpha$ iterations for a node lookup in a given bucket. For $k = 20$ and $\alpha = 3$, that is 3-way queries to seven multicast groups. With 160 buckets each peer would need at least 160 groups to do queries across its address space. If the multicast queries were $\alpha$-way, Chuang-Sirbu estimates a 20% to 30% savings, and if the queries were k-way, k=20, Chuang-Sirbu estimates a 45% to 64% savings from multicasting Kademlia requests, although responses would be unicasted.

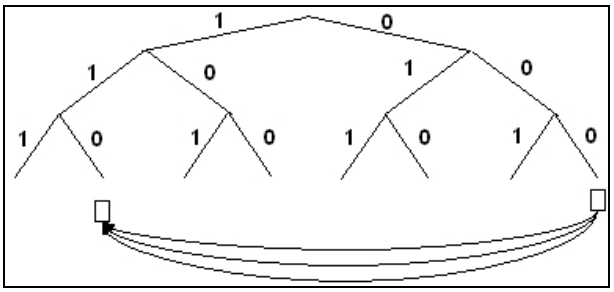



**Figure 9: Kademlia node lookup using $\alpha$=3 to nodes in k-bucket 110, i.e., nodes whose distance is in the range [$2^5$..$2^6$].**

## 6.2 Meridian – measurement overlay

Meridian [25] is a measurement overlay in which relative distance from other nodes in the overlay is used for solving overlay lookups like closest node discovery and central leader election. Each peer organizes its adjacent nodes into a set of concentric rings, each ring contains k = O(log N) primary entries and l secondary entries. In a simulation of N=2500 nodes, k=16, and the number of rings i* = 9.

Meridian uses a gossip protocol to propagate membership changes in the overlay. During a gossip period, a message is sent to a randomly selected node in each of its rings. The message contains one node randomly selected from each of its rings. Unicast gossip messages can be multicast to i* destinations using a single i*-way message. For i* = 9, the savings is 36% to 53% over unicast messaging.

## 6.3 Accordion – variable hop overlay

Accordion [26] is a variable hop overlay, in which a peer limits its routing table update message level based on its available bandwidth. During periods of low bandwidth, routing table accuracy can approach that of multi-hop overlays while for higher bandwidth, routing table accuracy reaches one-hop.

Unlike Kademlia and EpiChord, Accordion uses *recursive* parallel lookups so as to maintain fresh routing table entries in its neighborhood of the overlay and reduce the probability of timeout. The peer requesting the lookup selects destinations based on the key and also gaps in its routing table. Responses to forwarded lookups contain entries for these routing table gaps. Note that recursive parallel lookups create more load on the target peer compared to iterative parallel lookups, since the target node receives p messages for each request.

Excess bandwidth is used for parallel exploratory lookups to obtain routing table entries for the largest scaled gaps in the peer's routing table. The degree of parallelism is dynamically adjusted based on level of lookup traffic and bandwidth budget, up to a maximum configuration such as 6-way.

Replacing Accordion p-way forwarded and exploratory lookups with multi-destination lookups will reduce edge traffic by (p-1)/2p; e.g., p=5 means 40% reduction on the edge. For a fixed bandwidth budget, this means that a peer can increase its exploration rate by factor of 2.5, substantially improving routing table accuracy. Alternately, a peer can operate at the same level of routing table accuracy (and number of hops per lookup) for a lower bandwidth budget.

## 6.4 Parallel random walk

Random walk has been shown to be the most efficient search technique in unstructured topologies that are represented as power-law graphs. In a random walk, if an incoming query can not be locally matched, the request is forwarded to a randomly selected neighbor, excluding the neighbor from which the request was received. Systems using random walk include Gia [27] and LMS [28].

The idea of using parallel random walks to reduce latency is first studied in [29] where k parallel random walks perform better than flooding and expanding rings. Suitable values for k are in the range of 16 to 64. Cooper [30] evaluates parallel random walk latency for graphs with power-law and square-root topologies (Table 3).

**Table 3 Search latency in a simulated network of 20,000 peers [30].**

| Walks | Square-root | Power-Law Low skew | Power-Law High skew |
|---|---|---|---|
| 1 | 8930 | 12090 | 16350 |
| 2 | 4500 | 6210 | 8970 |
| 5 | 1800 | 2490 | 3740 |
| 10 | 904 | 1250 | 1880 |
| 20 | 454 | 630 | 947 |
| 100 | 96 | 130 | 194 |

Multi-destination routing can be used at the initial node in a parallel random walk, and will reduce edge traffic as well as some internal traffic. Further research is needed to determine if parallel forwarding can be efficiently used at other hops in the random walk.

## 6.5 Replication and load-balancing

Beehive [31] is a replication mechanism for prefix-based multi-hop overlays such as Kademlia and Pastry. Assuming object popularity follows Zipf distribution, Beehive uses object access statistics to proactively push objects to sufficient levels in the overlay to meet the required number of hops per query.

Parallel messaging in Beehive occurs in two areas. First, object access statistics are aggregated at each object's home node and propagated to nodes along the access path. Second, each peer locally determines for objects that it currently stores, whether the access level for each object requires increased replication and will push the object to other peers that precede it in the prefix-based routing path.

In both cases, the potential parallelism is determined by the prefix size used in the overlay routing mechanism. For base b in a m-bit address space, the tree has up to b-way branching factor with at most m/b hops from root to leaf nodes. However, few nodes will reach b-way branching due to sparseness of the address space, limiting the benefits of multi-destination routing to parallelize Beehive. However it is possible that peers could push replicas and aggregated statistics to several levels to achieve greater message parallelism.

## 6.6 Overlay multicasting

Several P2P overlays support multicasting. These are referred to as implicit ALM, and include Scribe/Pastry, Bayeux/Tapestry, and NICE. ALM trees suffer from constraints on the in-degree and out-degree of nodes which are using unicast links to connect to parent and children nodes. This increases path length in the tree.

If the network contains hosts which implement the multi-destination multicast routing, overlay nodes sending a multi-destination multicast message send the message to one such host. The host routes the multi-destination messages to other hosts according to the network routing rules. At each such host, duplicate messages containing the subset of destinations for each forwarding path are created and routed. This continues until a message contains only a single address in which case it is converted to a unicast message and is routed to its destination.

Further this integration with ALM and multi-destination routing means that arbitrarily large groups can be created. For example, suppose we limit multi-destination packets to 50 destinations and each node is constrained to say C number of connections. Nevertheless we can form overlay trees of millions of nodes where each node connects to at most C*50 out-going nodes. Each node receiving a single incoming packet forwards it using the set of addresses which is corresponding to its adjacencies.

# 7 Conclusion

We have shown that parallelizing a variety of overlay routing algorithms using multi-destination multicasting instead of parallel unicast messages results in significantly reduced message traffic on both edge and internal links. In structured overlays, this message reduction occurs for a variety of operations

such as joins, routing table maintenance, and application lookups. In general, latency behavior and operational semantics are retained.

We have discussed a number of existing overlay designs to show that multi-destination addressing is generally applicable as a performance enhancement method. The examples include DHT operations like lookup and insert, replication, maintenance, and measurement operations.

The primary limit of this approach is the maximum number of addresses in one packet. About 50 addresses are possible, however, in our experimentation we did not find any applications beyond a level of parallelisms of 10. Hence the limitation does not affect any of the various algorithms we examined. Also larger trees can be supported by using XCAST groups in stages.

We defined criteria for determining whether an overlay message can be parallelized using multicast. These criteria are maximum group size, number of groups, the time to create a new multicast group, and the group formation rate.

We have demonstrated our findings on two overlay maintenance algorithms: EpiChord and EDRA*. EpiChord employs an opportunistic approach to keeping its routing tables accurate by using parallel requests. We have shown that we can use XCAST to reduce the required messages per link and with that the bandwidth on the links. EDRA* is an algorithm for active maintenance of the routing tables. We have shown how the maintenance messages from a node can be combined into an XCAST message.

The Chuang-Sirbu law predicts multicast savings as $1\text{-}m^{-0.2}$. We have shown that the assumptions behind Chuang-Sirbu are appropriate for large-scale peer-to-peer overlays. However, as has been shown by previous work, the savings factor varies depending on the network topology between -0.2 and -0.34. In the examples in the paper, parallelism ranged from 3-way to 10-way or more, meaning message reduction of up to 54%. The actual achieved savings for EpiChord and EDRA* is somewhat lower than this because of unicast and 2-way XCAST retransmissions in EpiChord, and maintenance messages in EDRA* which are sent with a ttl-1 level of parallelism. Thus for 5-way EpiChord we have achieved a message saving of about 30% and for EDRA* of about 35%. At the same time the used bandwidth on the links is substantially reduced. To achieve an identical hop count for lookup messages, EDRA* with XCAST uses about a third of the bandwidth per node than EDRA* using unicast.

Besides operating on a reduced bandwidth requirement, the XCAST savings can also be used to increase the accuracy of the routing table and hence the hop count for lookup messages at no extra cost when compared with unicast transmission.

### 7.1 Future Work

XCAST is currently being deployed on PlanetLab [33,34] and we are planning to integrate P2P overlays with XCAST on PlanetLab. We are also designing a hybrid Application Layer Multicast (ALM)-XCAST overlay [35].

## Acknowledgement

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

## References

[1] J. Risson and T. Moors: Survey of research towards robust peer-to-peer networks: Search methods, Computer Networks, Elsevier Science, Vol. 50, pp 3485 – 3521.

[2] J. Chuang and M. Sirbu Pricing multicast communications: A cost-based approach. In Proceedings of INET 1998.

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

[4] M. Ammar. Why Johnny Can't Multicast: Lessons about the Evolution of the Internet. Keynote - NOSDAV 2003.

[5] R. Boivie, N. Feldman , Y. Imai , W. Livens , D. Ooms, O. Paridaens, Explicit Multicast (Xcast) Basic Specification, draft-ooms-xcast-basic-spec-11.txt, Work in Progress. Jan. 2007.

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

[7] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, S. Lim. A Survey and Comparison of Peer-to-Peer Overlay Network Schemes. IEEE Communications Surveys and Tutorials, Second Quarter 2005, Volume 7, No. 2.

[8] K. Aberer, L. Onana Alima, A. Ghodsi, S. Girdzijauskas, M. Hauswirth, S. Haridi. The essence of P2P: A reference architecture for overlay networks, The Fifth IEEE International Conference on Peer-to-Peer Computing, Konstanz, 31 Aug - 2 Sep 2005.

[9] I. Gupta, K. Birman, P. Linga, A. Demers, R. van Renesse. Kelips: building an efficient and stable P2P DHT through increased memory and background overhead. Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03). 2003.

[10] K. P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, Y. Minsky. Bimodal Multicast. ACM Trans. Computing Systems, 17:2, pp. 41-88, May 1999.

[11] T. Oh-ishi, K. Sakai, H. Matsumura, A. 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.

[12] T. Oh-ishi, K. Sakai, K. Kikuma, and A. Kurokawa. Study of the Relationship between Peer-to-Peer Systems and IP Multicasting. *IEEE Communications Magazine*. Jan. 2003.

[13] G. Phillips, S. Shenker, and H. Tangmunarunkit. Scaling of multicast trees: Comments on the Chuang–Sirbu scaling law. ACM SIGCOMM 1999.

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

[15] R. Chalmers and K. Almeroth, Modeling the Branching Characteristics and Efficiency Gains of Global Multicast Trees, *IEEE Infocom*, Anchorage, AK, USA, April 2001.

[16] S. Fahmy and M. Kwon, Characterizing overlay multicast networks and their costs, *IEEE/ACM Transactions on Networking*, to appear in 2007.

[17] 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.

[18] A. Gupta, B. Liskov, R. Rodrigues. Efficient routing for peer-to-peer overlays. Proceedings of the 1st Symposium on Networked Systems Design and Implementation (NSDI 2004), 2004, pp. 113-116.

[19] L. Monnerat and C. Amorim. D1HT: A Distributed One Hop Hash Table. In *Proc of the 20th IEEE Intl Parallel & Distributed Processing Symposium (IPDPS)*, April 2006.

[20] J. Buford, A. Brown, M. Kolberg. Parallelizing Peer-to-Peer Overlay with Multi-Destination Routing. IEEE Consumer Communications and Networking Conference (CCNC 2007).

[21] J. Buford, A. Brown, M. Kolberg. Multi-Destination Routing and the Design of Peer-to-Peer Overlays, IEEE Consumer Communications and Networking Conference (CCNC 2007), Workshop on Peer-to-Peer Multicasting (P2PM).

[22] J. Buford, A. Brown, M. Kolberg. Analysis of an Active Maintenance Algorithm for an O(1)-Hop Overlay. Submitted to IEEE Globecom 2007.

[23] M. Kolberg, F. Kolberg, A. Brown, J. Buford. A Markov Model for the EpiChord Peer-to-Peer Overlay in an XCAST enabled Network. IEEE International Conference on Communications (ICC) 2007.

[24] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In Proc of IPTPS02, Cambridge, USA, March 2002.

[25] B. Wong, A. Slivkins and E. G. Sirer. Meridian: A Lightweight Network Location Service without Virtual Coordinates. In Proc. of *SIGCOMM Conference*, Aug 2005.

[26] J. Li, J. Stribling, R. Morris, and M. F. Kaashoek, Bandwidth-efficient Management of DHT Routing Tables, NSDI 2005.

[27] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, S. Shenkar. Making Gnutella-like P2P Systems Scalable. In Proc. ACM SIGCOMM. 2003.

[28] R. Morselli, B. Bhattacharjee, A. Srinivasan, and M. A. Marsh. Efficient lookup on unstructured topologies. In Pro of the 24th Annual ACM Symposium on Principles of Distributed Computing PODC '05.

[29] Q. Lv, P. Cao, E. Cohen, K. Li, S. Shenker: Search and replication in unstructured peer-to-peer networks. In: Proc. of ACM Int'l Conf. on Supercomputing (ICS'02), 2002.

[30] B. Cooper. An Optimal Overlay Topology for Routing Peer-to-Peer Searches. Middleware 2005.

[31] V. Ramasubramanian and E. Sirer. Beehive: O(1) Lookup Performance for Power-Law Query Distributions in Peer-to-Peer Overlays. In Proc. of Networked System Design and Implementation (NSDI), March 2004.

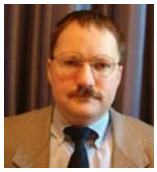
[32] SSFNet. http://www.ssfnet.org.

[33] N. Kawaguchi. XCAST PlanetLab Integration, IEEE Consumer Communications and Networking Conference (CCNC 2007), Workshop on Peer-to-Peer Multicasting (P2PM).

[34] E. Muramoto, Y. Imai, S. Sakurai, F. Kan, N. Kawaguchi, D.Matsui, Y.Shinoda. Experimental Deployment Method for Router Supported ALM using PlanetLab. IRTF. Work in Progress. draft-muramoto-irtf-sam-exp-testbed-00.txt. June 2007.

[35] J. Buford. Hybrid Overlay Multicast Framework, draft-irtf-sam-hybrid-overlay-framework-01.txt, IRTF, Work in Progress, Jan 2007.

## Vitae



**John Buford** is a Lead Scientist at the Panasonic Princeton Laboratory, Princeton, New Jersey, USA. Previously he was VP of Software Development at Kada Systems, Director of Internet Technologies at Verizon, and Chief Architect-OSS at GTE Laboratories. Earlier he was tenured Associate Professor of Computer Science at the University of Massachusetts Lowell, where he also directed the Distributed Multimedia Systems Laboratory. He has authored or co-authoered seventy refereed publications including the book Multimedia Systems. He is an IEEE Senior Member and is co-chair of the IRTF Scalable Adaptive Multicast Research Group. He holds the PhD from T. U. Graz, Austria, and MS and BS degrees from MIT.

**Alan Brown** gained his Software Engineering BSc (Hons) from the University of Stirling in Scotland where he is now pursuing his PhD on Service Discovery in Peer-to-Peer Networks. His research interests in Peer-to-Peer include Service Discovery, Routing Efficiency and network simulations. Alan is involved with a joint research project with Panasonic (USA), researching Peer-to-Peer overlays. He is also interested in Home Networking and Personal Area Networks, with extensive experience with OSGi, SIP and UPnP.

**Mario Kolberg** studied computer science at the HTWS Zittau/Goerlitz in Germany. After spending one year in the computer science department at Humboldt University in Berlin he joined the Communications Division in the Department of Electronic and Electrical Engineering at the University of Strathclyde in June 1997. In September 2000 he moved to the Department of Computing Science and Mathematics at Stirling University where he is a member of academic staff. His research interests include Home Networks, Feature Interaction, Service Creation, and IP Telephony. He has extensive experience of home networking protocols and of implementing services on the OSGi platform. He is leading a project funded by Panasonic (USA) investigating Peer-to-Peer overlays. He is also involved with the MATCH project, focusing on integrating different technologies for care in the home. He was leading an effort providing a proof-of-concept demonstrating the integration of digital pen and paper with networked appliances. Previously he has worked on an EU funded project (TOSCA) on issues related to the rapid development of distributed telecommunications services. On invitation by Telcordia Technologies, a leading presence in home automation, he has visited their laboratories in the USA for an extended period. He holds a PhD in Electrical Engineering from the University of Strathclyde.