A Markov Model for the EpiChord Peer-to-Peer Overlay in an XCAST enabled Network

Mario Kolberg*, Florence Kolberg†, Alan Brown*, and John Buford‡
*Department of Computing Science and Mathematics, University of Stirling, Stirling, UK, {mko,abr}@cs.stir.ac.uk
†Department of Electronic and Electrical Engineering, University of Strathclyde, Glasgow, UK, fkolberg@googlemail.com
‡Avaya Labs, USA, buford@samrg.org

***Abstract*— Structured Peer to Peer (P2P) overlay networks are becoming increasingly popular. Multi-hop systems achieve a successful lookup in O(log N) hops, whereas one-hop systems approach O(1) hops. Both approaches, but especially one-hop overlays suffer from a high number of identical messages being sent to a number of nodes on the overlay.**

Previous work showed that P2P networks benefit from the integration of the overlay network with the underlay network in which multi-destination multicast routing is available. This allows combining identical messages from the same source into joint multi-destination multicast messages to significantly reduce the number of messages. Our experimentation has centered around the one-hop EpiChord overlay. Here the problem is described using a Markov Model for more advanced analysis. The Markov Model is believed to be novel in two aspects: it is the first to investigate one-hop overlays and it is the first to study the performance of multi-destination multicast including the consideration of retransmissions of requests.

I. Introduction

A number of multihop structured P2P overlay systems have been proposed. However, they require O(log N) hops for a message to reach its destination. This affects performance as overlay hops are routed in the underlay (native network) as a number of hops. To remedy this issue one-hop overlay systems have been proposed which only need O(1) hop for routing a message to its destination. One-hop systems trade the number of hops with an increased routing table size and an increased number of maintenance messages to keep the routing tables up to date. EpiChord [1] is a leading one-hop overlay which achieves excellent performance in terms of hop count when contacting peer nodes by using parallel lookup requests.

The message overhead in EpiChord can be significantly reduced by using multi-destination routing [2] but also some multi-hop systems will gain [3].

Multi-destination routing combines identical IP packets to be sent to P destinations, into a single packet which is sent to P destinations. Multi-destination routing enabled routers check the routing decision for each address in a packet. If all addresses are to be routed along the same path, the packet stays together. If not, the packet is split and the subset of destinations for each path are created. If a packet only contains one address, normal unicast routing is employed.

Recently an experimental IP protocol for multi-destination multicast called explicit multicast (XCAST) protocol has been specified [4] and several XCAST testbeds have been deployed. He and Ammar [5] analyze the performance of XCAST. They state that the cost of a multi-destination routing decision is comparable to that of a unicast routing decision, particularly for small address sets. The quantitative benefits of multicast have been expressed in the Chuang-Sirbu scaling law [6] which shows that the efficiency of multicast vs. unicast is about $m^{0.8}$ (where m is the multicast group size).

There has been a previous attempt to model a Peer-to-Peer overlay using a Markov model [7], however that model is for a multi-hop system (P-Grid) and was used for a different purpose.

This paper provides the following contributions:

- the paper provides the first formalisation of the EpiChord retransmission model
- the paper defines a Markov model which captures the dynamics of this model
- the paper uses the Markov model to evaluate the potential for multicast parallelism in EpiChord messaging
- the paper correlates our simulation results of multicast enabled EpiChord with the Chuang-Sirbu [6] law using the Markov model

A. Multidestination Routing vs. Host group Multicast

Oh-ishi et al. have considered the use of Protocol Independent Multicast (PIM) in sparse mode (PIM-SM) [8] and source specific mode (PIM-SSM) [9] to reduce message traffic in peer-to-peer systems. Their analysis focuses on using multicast routes between peers in different ISP networks.

With larger P2P overlays, the use of host-group multicast in DHT creates too much traffic and router overhead if each node maintained multicast addresses for all or many subsets of the overlay network.

To use native host-group multicast to send a message to some set of nodes in order to issue parallel queries, state in the routers must be created first and the receivers brought into the multicast group. 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.

In summary, IP multicast is designed for small numbers of large sets of recipients. However, P2P overlays send messages to a large number of groups each with small numbers of nodes. So IP multicast is not a good choice for use in parallelizing

DHT operations. By contrast, multi-destination multicast routing does not require additional state in routers. Further, due to overlay routing mechanism, destination peer addresses are already known so there is no group join overhead.

B. Using Explicit Multicast with EpiChord

In EpiChord, each peer maintains a full-routing table. Assuming a fully up-to-date routing table, messages can be sent to any other node in one hop. However, achieving a 100% accuracy is notoriously difficult. Even with lower routing table accuracy EpiChord's performance approaches O(1) performance, compared with the O(log N) performance of multi-hop overlays. However, this is at the cost of increased number of routing table update messages and storage requirements.

An EpiChord peer's routing table is initialized when the peer joins the overlay by getting copies of the successor and predecessor peers' routing tables. Thereafter, the peer adds new entries when a request arrives 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 additional probe messages to selected nodes.

To improve the success of lookups, EpiChord uses p-way parallel requests directed to peers nearest to the target node. All parallel lookups in EpiChord are carried in separate unicast messages. In our approach, these parallel unicast messages are replaced by a single XCAST message [2].

The performance of XCAST enabled EpiChord has been evaluated through simulation using an extended version of the original EpiChord simulator from MIT on SSFNet [10].

In the following an analytical model of XCAST enabled EpiChord using a Markov Chain is discussed. While the Chuang-Sirbu scaling law predicts a saving of about $m^{0.8}$, this does not take into account EpiChord specific retransmissions of messages and timeouts of nodes. The work on the model has been motivated by two reasons:

- It allows for a more flexible and scalable analysis than what is possible with simulation.
- Comparing the model with our simulation results, our simulation results can be verified.

In the next Section the general approach to calculate the saving achieved using multicast rather than unicast is discussed. This is followed by a discussion of node timeouts and message retransmissions in EpiChord and how this is modeled in a Markov Chain. Finally our results and conclusions are presented.

II. Estimating the XCAST Gain

To quantify the multicast efficiency against the unicast case the hop count is used as a metric. The saving can thus be defined like in [11], as:

$$\delta = 1 - \frac{\text{average multicast hops}}{\text{average unicast hops}} \tag{1}$$

The value of δ is in the range [0,1]. When the value equals 0, the use of XCAST has no advantages over unicast. However, as the value increases towards 1, so does the benefit of using multicast.

[12] reformulates Equation (1) as:

$$\delta = 1 - \frac{g_N(m)}{f_N(m)} \tag{2}$$

m represents the multicast group size, N is the number of nodes in the overlay network, $g_N(m)$ is the average number of hops in the multicast distribution tree and $f_N(m)$ is the average number of hops in the unicast case. $f_N(m) = m.E[H]$, where $E[H]$ is the average number of hops in the physical topology from one overlay source node to a destination overlay node. This notation is used in this paper.

A. Multicast Gain without considering re-transmissions

The Chuang-Sirbu Law [6] states that the number of hops in the multicast tree is a function of the multicast group size and a scaling factor k with a value between 0 and 1. Indeed, they approximate $g_N(m)$ by:

$$g_N(m) = E[H] * m^k \text{ where } k = 0.8. \tag{3}$$

Hence the multicast metric can be defined as:

$$\delta = 1 - \frac{E[H] * m^{0.8}}{E[H] * m} = 1 - m^{-0.2} \tag{4}$$

δ returns the value of the savings that multicast offers when compared to unicast transmissions of m lookups. For example, if $m = 5$, the saving obtained by employing multicast rather than unicast is $1 - (5)^{-0.2} \approx 27.5\%$

B. Retransmissions in the EpiChord simulation

For simulating EpiChord with XCAST the SSFNet simulator together with the MIT EpiChord model was used and extended to include XCAST. When a node is sending a request message, two queues are employed, the pending queue and the tried queue. The pending queue is the queue of all nodes whom we await a response from. The tried queue is the queue of nodes that we have sent a request to for this lookup. For a p-way request, p nodes are sent a lookup message and those p nodes are put into pending and tried queues. Hence initially, both queues hold the same entries.

The following discussion concentrates on the pending queue. The tried queue simply keeps track of nodes which a message has been sent to avoid contacting nodes repeatedly for a single request.

The setup of the pending queue may change during a request, depending on node timeouts and negative responses received. Figure 1 shows the possible behaviour of the pending queue during a request. P is the level of parallelism. As stated above, initially a P-way request is sent to the nodes in the queue. If a node times out, the node is tried a further two times using single UDP messages. After the third timeout, the node is removed from the pending queue, and if the queue length equals P, two new nodes are added to the queue and a two-way XCAST request is sent to them. Hence the queue

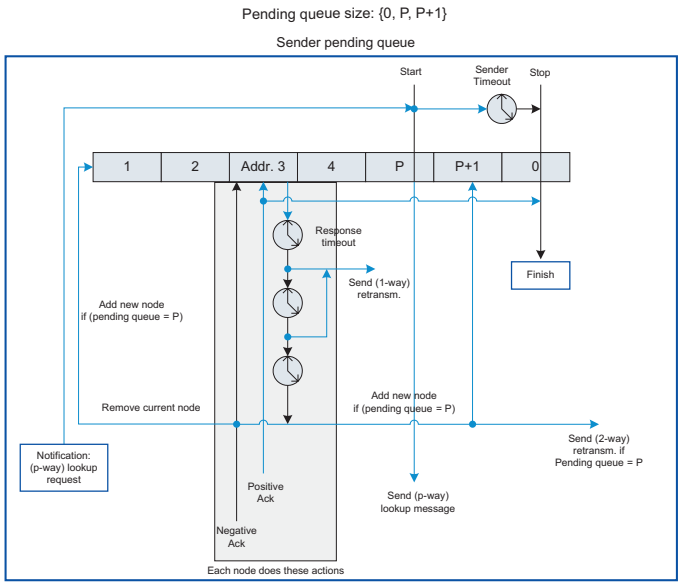

Fig. 1. Behaviour of Pending Queue after Reception of Timeout, Negative and Positive Response

length is now P+1. If the queue length is already P+1, no new message is sent. An example of these cases is shown in Fig 2.

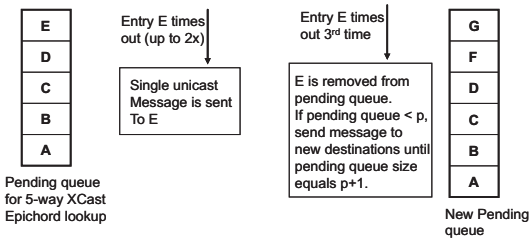

Fig. 2. Timeout scenario for a 5-way XCAST EpiChord request.

If a node responds with a negative message (does not have the data), it is also removed from the queue, and again depending on the current length of the queue a new two-way request is sent to new nodes. This process continues until a node responds with the data looked for (positive response). As above, XCAST is used for this request. An example scenario is shown in Figure 3. Thus the behaviour after a negative response and a third node timeout is identical.

Retransmissions stop, as soon as a success message from any tried node is received. In the next section is is shown how (4) must be modified to take retransmissions into account.

C. Multicast Gain when considering re-transmissions

To take retransmissions into account, the cost of unicast and the cost of multicast, C_u and C_m, respectively, can be defined with the Equations (5) and (6). The cost of multicast can be derived as the sum of the cost of sending m-way lookups, (m-1)-way lookups, ..., 1-way lookups because the multicast group size varies according to the size of the pending queue.

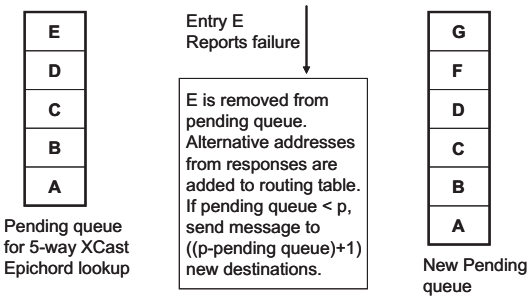

Fig. 3. Failure scenario for a 5-way XCAST EpiChord request.

$$\begin{aligned} C_u &= f_N(m) + f_N(re-transmissions) \\ &= E(H) * m + E(H) * E(X) \end{aligned} \quad (5)$$

where X is the number of retransmissions and $E(X)$ is the average number of retransmissions for a request.

$$\begin{aligned} C_m &= g_N(m) + E(m-1) * g_N(m-1) + \\ &\quad \ldots + E(1) * g_N(1) \\ &= E(H) * m^k + E(m-1) * E(H) * (m-1)^k + \\ &\quad \ldots + E(1) * E(H) \end{aligned} \quad (6)$$

where $E(m-1)$ is the average number of (m-1)-way retransmissions.

In the simulation of EpiChord, only certain types of messages can be sent: a single p-way request initially, 2-way XCAST messages after a negative response from a tried node or the third timeout of a node, and single UDP message after a node times out once or twice. Thus (5) and (6) can be simplified as shown in (7) and (8). The Chuang-Sirbu law would suggest a value of 0.8 for k.

$$C_u = E(H) * m + E(2) * 2 * E(H) + E(1) * E(H) \quad (7)$$

$$C_m = E(H) * m^k + E(2) * E(H) * 2^k + E(1) * E(H) \quad (8)$$

Here E(2) is the expected number of 2-way retransmissions per lookup and E(1) is the expected number of unicast retransmissions per lookup. Thus only the cost of the initial request plus two-way retransmissions plus UDP retransmissions, until the requesting node has obtained a positive acknowledgment, are considered.

The pending queue size changes depending on the type of response the sender receives. The probabilities of receiving each type of response is known from our simulations, hence the pending queue size can be calculated and thus the average number of 2-way and 1-way retransmissions the sender issues for each lookup. To do so, the behaviour of the pending queue has been modeled as a Markov chain.

State	Number of timeouts	Probability type	Event	Next State	Transition probability
$\{k,i,j\}$ $0 \le i+j < k$ $1 \le i,\ 1 \le j$	0	$(k-i-j)/k$	neg. response	$\{\hat{k},i,j\}$	$(k-i-j)/k \times p_-$
	0	$(k-i-j)/k$	timeout	$\{k,i+1,j\}$	$(k-i-j)/k \times p_t$
	1	i/k	neg. response	$\{\hat{k},i-1,j\}$	$(i/k) \times p_-$
	1	i/k	timeout	$\{k,i-1,j+1\}$	$(i/k) \times p_t$
	2	j/k	neg. response/timeout	$\{\hat{k},i,j-1\}$	$(j/k) \times (p_- + p_t)$
$\{k,i,j\}$ $0 \le i+j = k$ $1 \le i,\ 1 \le j$	1	i/k	neg. response	$\{\hat{k},i-1,j\}$	$(i/k) \times p_-$
	1	i/k	timeout	$\{k,i-1,j+1\}$	$(i/k) \times p_t$
	2	j/k	neg. response/timeout	$\{\hat{k},i,j-1\}$	$(j/k) \times (p_- + p_t)$
$\{k,i,j\}$ $i = k$	1	1	neg. response	$\{\hat{k},i-1,j\}$	p_-
	1	1	timeout	$\{k,i-1,j+1\}$	p_t
$\{k,i,j\}$ $j = k$	2	1	neg. response/timeout	$\{\hat{k},i,j-1\}$	$(p_- + p_t)$
$\{k,i,j\}$ $0 \le i+j < k$ $1 \le i,\ j = 0$	0	$(k-i)/k$	neg. response	$\{\hat{k},i,j\}$	$(k-i)/k \times p_-$
	0	$(k-i)/k$	timeout	$\{k,i+1,j\}$	$(k-i)/k \times p_t$
	1	i/k	neg. response	$\{\hat{k},i-1,j\}$	$(i/k) \times p_-$
	1	i/k	timeout	$\{k,i-1,j+1\}$	$(i/k) \times p_t$
$\{k,i,j\}$ $0 \le i+j < k$ $i = 0,\ 1 \le j$	0	$(k-j)/k$	neg. response	$\{\hat{k},i,j\}$	$(k-j)/k \times p_-$
	0	$(k-j)/k$	timeout	$\{k,i+1,j\}$	$(k-j)/k \times p_t$
	2	j/k	neg. response/timeout	$\{\hat{k},i,j-1\}$	$(j/k) \times (p_- + p_t)$
$\{k,i,j\}$ $0 \le i+j < k$ $i = 0,\ j = 0$	0	1	neg. response	$\{\hat{k},i,j\}$	p_-
	0	1	timeout	$\{k,i+1,j\}$	p_t

TABLE I
TRANSITION PROBABILITIES WITH $\hat{k} = P+1$ if k = P and P if k = P + 1

III. A MARKOV MODEL FOR RETRANSMISSIONS IN EPICHORD

For the model we assume that a node which sent out requests to other nodes in its pending queue, can receive three different types of responses: positive response, negative, and timeout. This is shown in Fig. 4.

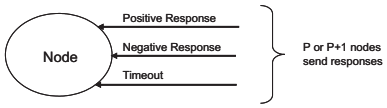

Fig. 4. Responses received by a node

The probabilities for a positive response (p+), negative response (p-), and timeout (pt) can be calculated as shown in (9). Initially the number of messages as observed in our simulations can be used, however, other values can be inserted allowing for *what-if* analysis.

$$\begin{aligned} p+ &= \frac{\text{number of positive responses}}{\text{total number of responses}} \\ p- &= \frac{\text{number of negative responses}}{\text{total number of responses}} \\ pt &= \frac{\text{number of timeouts}}{\text{total number of responses}} \end{aligned} \tag{9}$$

Figure 5 shows the resulting Markov chain for a request with P=3. In the diagram all transient states are shown, however, the absorbing final state representing the reception of a positive response {0,0,0} has been omitted for clarity. The absorbing state can be reached from any state when a positive response is received.

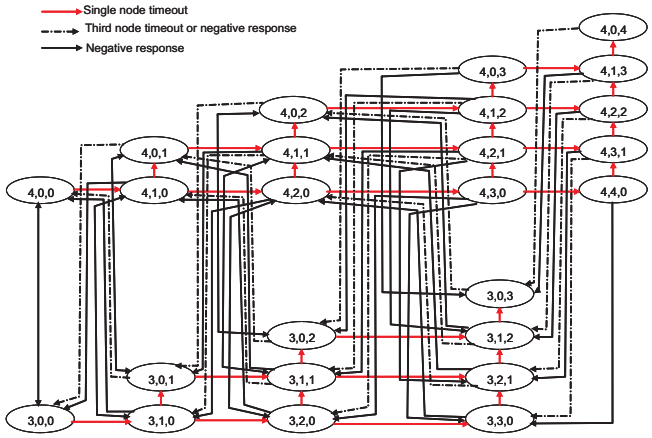

Fig. 5. Markov Chain Model - State Diagram for the Pending Queue

The name of each state contains three numbers $\{k,i,j\}$: k is the current length of the queue, i is the number nodes in the queue which have timed out once, and j is the number of nodes in the queue which have timed out twice. The initial state of the queue is {P,0,0}, where the pending queue has a size P and 0 timeouts have occur. The retransmission process ends when the sender receives a positive response, thus the pending queue state becomes {0,0,0} (not shown).

When designing the model, three assumptions have been made to achieve a good balance between accuracy and complexity of the model.

Assumption 1: In this paper, an embedded discrete (homogeneous) time Markov Chain model was used to describe the pending queue operation. This means the transitions probabilities do not change over time (homogeneous) and the model is expressed in terms of transitions probabilities rather than rates. The time the queue is in each state is ignored and the probabilities of making transitions from each state to all other states when a transition occurs are the only concern.

Assumption 2: A transition occurs after one and only one response was received by the queue. Hence transitions consider only the change of state of any one node in the pending queue.

Assumption 3: It is equally likely for a node to timeout once, twice, or three times. The probabilities of timing out and to receive a negative response is independent of the state.

The transitions between states indicate the reception of a negative response or a node timeout. Taking a closer look at the model, it can be seen that it is split into two halfs. The bottom half models the states for the queue length = P (3 in this example), whereas the top half includes the states for queue length = P+1 (4 in this example). Moving from the start state to the right, increasing numbers of first node timeouts are modeled (up to P). Moving up line by line until the middle of the diagram, the second timeouts of nodes are modeled (up to P). The same is repeated in the top half of the model, however for up to P+1 nodes. The two halfs are joined by transitions indicating third node timeouts and negative responses. Negative responses and the third timeout of a node trigger the same transition in the diagram, hence arrows indicating these are combined for brevity. The transition probabilities of the Markov model are shown separately in Table I.

The number of transient states n in the Markov Chain can be calculated using Equation (10). The first half calculates the number of states for queue length equal P, whereas the second half calculates the number of states when the pending queue size equals P+1.

$$n = (P+1) * (P+2) * 0.5 + (P+2) * (P+3) * 0.5 \quad (10)$$

The probabilities for timeouts and negative responses have been obtained using our SSFNet XCAST-EpiChord simulator [2].

The Markov state diagram above is then used to calculate the expected number of retransmissions per lookup. For this, the transition matrix needs to be derived. Let T be the transition matrix. T is composed of the transient states and the one absorbing state.

$$T = \begin{pmatrix} Q & R \\ 0 & I \end{pmatrix} \quad (11)$$

Here the transition matrix for the transient states, Q, is of size $(n \times n)$. n as calculated above is the number of transient states in the model. The absorption matrix R is of size $(n \times 1)$, the matrix of zeros O is of size $(1 \times n)$, and finally the Identity matrix I is of size (1×1). So in total, T is of size $(n+1 \times n+1)$.

Q can now be used to calculate the fundamental matrix N. The entry N(i,j) shows the expected number of visits to transient state j before absorption, starting in state i.

N depends only on the square sub-transition matrix Q. Hence the total number of visits is the sum of the expected number of visits on the different steps and that the expected number of visits to j on the n-th step, starting i is Q(i,j). Or written differently: $N = I + Q + Q^2 + Q^3 + Q^4 + \ldots$. Here I is the Identity matrix.

Now N can be calculated using the relatively simple Equation (12). If both the left and the right side are multiplied by $(I - Q)$, this results in $N * (I - Q) = I$. Hence N can be calculated as (inv is the matrix inverse function):

$$N = inv(I - Q) \quad (12)$$

Using N, the expected number of 2-way retransmissions can be derived. It represents the sum of N(i,j) when the queue state goes from P to P+1. The expected number of unicast retransmission can be calculated in a similar fashion. It is the sum of N(i,j) when a first or second timeout occurs in queue state P and P+1. Matlab was used to calculate the fundamental matrix and the values of expected number of retransmissions. These can then be used in (7) and (8). For the following experimentation k=0.8 according to the Chuang Sirbu law has been assumed. With these two values the saving of using XCAST can be calculated using (13).

$$saving = 1 - \frac{C_m}{C_u} \quad (13)$$

IV. Experimentation

XCAST reduces the number of messages sent per lookup and hence the number of messages per link. Thus achieving a bandwidth saving on each link used by the messages.

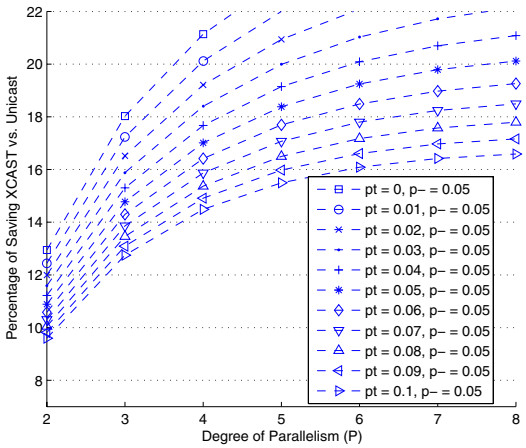

Fig. 6. Saving achieved using XCAST for increasing levels of node timeouts for different degrees of parallelism (lookup intensive).

The following figures illustrate the saving that can be achieved for varying probabilities of receiving a negative acknowledgment (p-) or timeout responses (pt). The values

chosen were guided by the values from our EpiChord simulations. Two sets of graphs are presented, one with p- and pt modeled on a lookup intensive behaviour in the overlay (nodes join the network at a rate of 2 per second and issue on average 2 lookups per second), and the other on a churn intensive behaviour (15 nodes join the overlay network per second and issue on one lookup every 10 seconds). This setup follows the original EpiChord simulations by MIT.

Figure 6 shows the XCAST savings which can be achieved for varying levels of node timeouts at different degrees of parallelism. The probability of negative responses was taken from our simulations under a lookup intensive workload. As can be seen the same saving of about 18% can be achieved by a 3-way lookup with p-=0.05 and no timeouts, and a 6-way request with the same value for p- and pt=0.07, or a 8-way request with the same p- and pt=0.08.

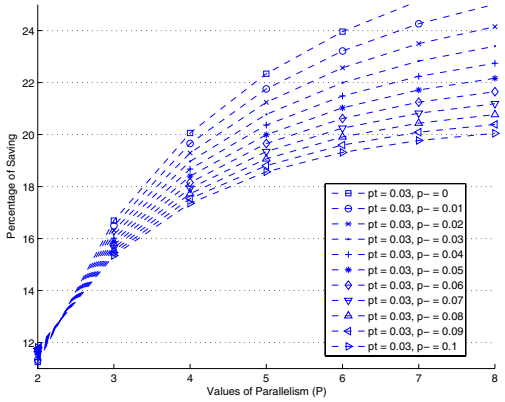

Fig. 7. Saving achieved using XCAST for increasing levels of negative responses for different degrees of parallelism (lookup intensive).

Figure 7 shows the XCAST savings which can be achieved for varying levels of negative responses at different degrees of parallelism. The probability of timeouts was taken from our simulations under a lookup intensive workload.

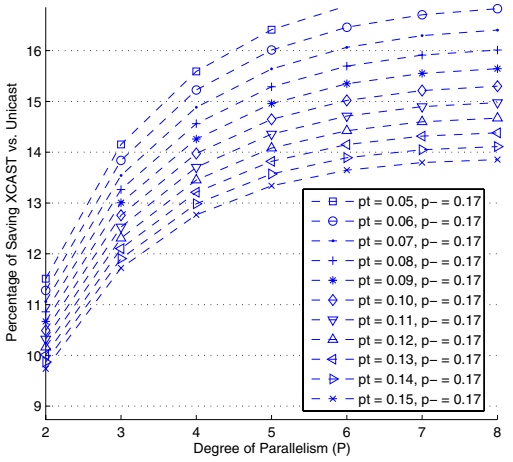

Fig. 8. Saving achieved using XCAST for increasing levels of node timeouts for different degrees of parallelism (churn workload).

Figure 8 shows the XCAST savings which can be achieved for varying levels of node timeouts at different degrees of parallelism. Here the probability of negative responses was taken from our simulations under a high churn workload.

Figure 9 shows the XCAST savings which can be achieved for varying levels of negative responses at different degrees of parallelism. The probability of timeouts was taken from our simulations under a high churn workload.

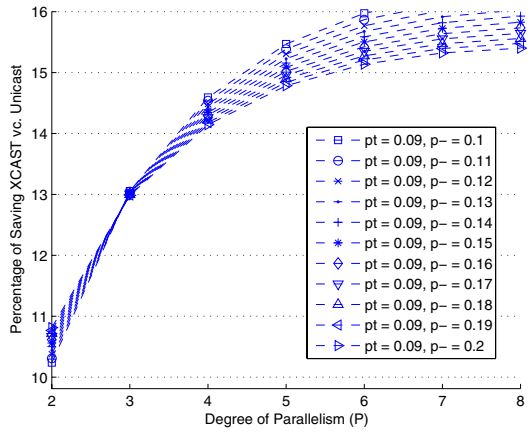

Fig. 9. Saving achieved using XCAST for increasing levels of negative responses for different degrees of parallelism (churn workload).

The figures show that for increasing numbers of node timeouts and negative responses and hence retransmissions the XCAST saving decreases. However, the saving decreases faster for increasing node timeouts than for increasing negative responses. This is because node timeouts result in UDP messages, whereas negative responses result in 2-way XCAST messages. In fact for 2-way EpiChord, the saving goes up slighty when considering negative responses. This is because the initial request is also only sent 2-way.

V. Comparison: Analytical and Simulation Results

A. Discussion of the number of retransmissions

As stated above, the probability of having a transition due to negative response or timeout in the Markov chain model have been calculated using results from the SSFNet simulations. The simulations returned the average number of negative and timeout responses before absorption per lookup. By dividing these values by the number of states in the Markov chain, the probability of having a negative response or a timeout can be calculated. Indeed, starting from any state in the pending queue, the probability that any node replies with a negative response is equally likely to occur. Similarly, the probability that any node times out, when the pending queue is in any state, is equally likely to occur.

In the simulations, values of 2.54 for the expected number of negative responses and 1.77 for the expected number of timeouts per request were obtained for P = 5 (degree of parallelism). Using (10) the number of transient states in the Markov chain model for P=5 can be calculated as 49. Thus, the

probability of receiving a negative response starting from any state is $\frac{2.54}{49} = 0.052$ and the probability of timeout starting from any state is $\frac{1.77}{49} = 0.036$.

The results from the Markov chain model for P=5 return values of 1.35 for the expected number of 2-way messages and 1.18 for the expected number of unicast retransmissions. The results obtained by simulation were 1.22 for the expected number of 2-way messages and 1.20 for the expected number of unicast retransmissions. Thus the model yields results quite close to the simulation values. Table II shows the values for P = 3, 4, and 5 for a simulated network of 1200 nodes under a lookup intensive workload (join rate of 2 nodes per second and two lookups per second).

P	Neg Resp. per lookup	Timeouts per lookup	Xcast (model)	Xcast (simul)	unicast (model)	unicast (simul)
3	1.44	1.3	0.77	0.75	0.87	0.9
4	1.98	1.54	1.06	1	1.02	1.05
5	2.54	1.77	1.35	1.22	1.18	1.2

TABLE II

LOOKUP INTENSIVE RESULTS FOR PARALLELISM 3, 4 AND 5

Table III show the comparison of the results obtained from the Markov model vs. simulation results for a network of 9000 nodes under high churn (join rate of 15 nodes per second and one lookup every ten seconds).

P	Neg Resp. per lookup	Timeouts per lookup	Xcast (model)	Xcast (simul)	unicast (model)	unicast (simul)
3	6.1	3.16	3.19	3.05	2.2	2.22
4	7.27	3.67	3.81	3.45	2.52	2.6
5	8.49	4.23	4.49	3.92	2.88	3.03

TABLE III

CHURN INTENSIVE RESULTS FOR PARALLELISM 3, 4 AND 5

The slight deviation between the model results and simulation results can be explained by the three assumptions discussed earlier.

The model could be made more accurate by using a transition rate matrix, rather than a transition matrix. By doing so, transitions probabilities are replaced by transitional intensities that would capture the time dependency between the reception of responses and the network state (e.g. a timeout occurs because a node has disappeared or because of congestion). In the current model, the probabilities are modeled on the average length of a request, i.e. the average time until absorption. However, taking into account transitional intensities to model probabilities will make the model more accurate. The three probabilities for the transitions pt, p-, and p+ would change with an increasing numbers of timeouts or negative responses. For instance, if many timeouts are encountered, the probability for p- will decrease. Such a model would not only capture single requests, but many requests. For this a new transition, indicating a new request, from the state $\{0, 0, 0\}$ (absorbing state) to $\{P, 0, 0\}$ (initial state for a request) needs to be included. This would change the Markov Chain to be ergodic.

Further, for this model we have assumed that only one response triggers a transaction to another state. Hence it could happen that more than one response is received during the interval $[t; t + \Delta k]$ (discrete interval) and thus the number of transitions in the proposed model should be higher, enabling transitions between more states (e.g. a transition from (P,0,0) to state (P,2,0) would indicate that two nodes timed out).

The simulation using SSFNet does not fully simulate busy links and routers as a source of lost messages. This makes the disappearance of nodes from the overlay the most likely event to cause timeouts. This results in the behaviour that it is *very* likely for a node which has timed out already once, to timeout further two times. In the model we have not assumed this behaviour, but a more realistic view that the probability of a node timing out once, twice, or three times is identical, and hence it is equally probable to receive p+, p-, and pt in any state. However, this approach could be fine-tuned. For instance, the probability of a timeout could be increased by a factor α with the number of timeouts occurring. As the probability of timeouts occurring increases, the probability of receiving a response decreases. Hence $p+$ and $p-$ would decrease by the same factor. These changed probabilities can be implemented in the current discrete Markov model using the same transitions. This change addresses the same issue as the use of transitional intensities. However, it is easier to implement.

B. Comparison of the achieved Multicast Saving

While the Chuang Sirbu law is widely accepted as a good estimation of the multicast saving for random networks, some work reports slightly different results where the factor is closer to 0.7. Mieghem et al [12] discuss the C-S law and state that the factor 0.8 is not constant but is dependent on the size of the network especially for small multicast group sizes. Chalmers and Almeroth [13, 14] found that for their networks modeled on realistic topologies they achieve a factor between 0.65 and 0.7. Fahmy and Kwon [15] confirm these results.

Our simulation uses a network model of the American academic network identical to what was used for the original MIT EpiChord simulations. Clearly, this is not a random network and our simulation results show a saving of 29% for a lookup intensive workload and 25% for a churn intensive workload for $m = 5$. These results indicate a savings factor k of just below 0.7. This matches the results cited above. To verify this we used varying values for k in the Markov model.

Figure 10 shows the saving for EpiChord without any retransmissions (solid lines), with retransmissions as expected under a lookup intensive workload (dotted-dashed lines), and with retransmissions under a churn intensive workload (dashed lines), for varying values of k. The figure shows that for $m = 5$, and $pt = 0.036$ and $p- = 0.052$ (lookup intensive), and $k = 0.7$ the saving is about 27%. For $m = 5$ and $pt = 0.086$ and $p- = 0.17$ (churn intensive), and $k = 0.7$ the saving is about 22%. This matches our simulation results.

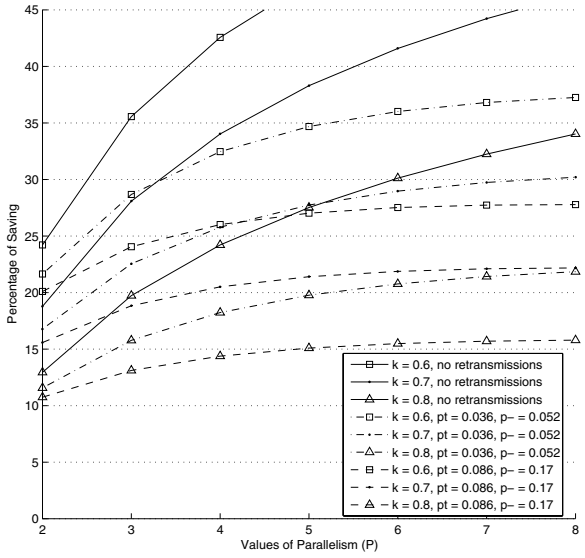

Fig. 10. Saving achieved using XCAST for varying factor k.

VI. Conclusions

In this paper, a relatively simple finite state discrete time Markov model was used to describe the dynamic behaviour of sending requests in EpiChord in an unambiguous way. Modeling EpiChord in this way is novel. The model can be parameterised in a number of ways: the probabilities for a node timeout, the probability of a negative response from a node, the level of parallelism used for EpiChord requests, the level of parallelism used for retransmissions, and the multicast savings factor k.

The presented model allows to study the effects of varying levels of retransmissions and node timeouts without lengthy simulations. When using EpiChord together with XCAST, the message saving taking into account retransmissions which can be achieved can be calculated. Here the Chuang-Sirbu law has been assumed for these calculations. For 5-way EpiChord including retransmissions a saving of about 20% can be achieved. However, as discussed, the Chuang-Sirbu law does not provide an accurate prediction for all types of networks. For networks which exhibit a smaller factor k, the savings will be higher.

The model allows to investigate the relationship between the level of parallelism used for lookups and the probabilities of timeouts and negative responses on the message saving.

More complicated finite state continuous time Markov models would express transitional rates in real time and hence also capture the behaviour of the message transmissions.

Acknowledgements

The authors would like to thank Ben Mestel and Sergei Zuyev for their help developing the model.

References

[1] 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.

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

[3] J. Buford, A. Brown, and M. Kolberg, *Multi-Destination Routing and the Design of Peer-to-Peer Overlays*, IEEE Consumer Communications and Networking Conference (CCNC) 2007 - 1st International Workshop on Peer-to-Peer Multicasting (P2PM'07), Las Vegas, USA.

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

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

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

[7] K. Aberer, A. Datta, M. Hauswirth: *Route maintenance overheads in DHT overlays*, WDAS 2004. The 6th Workshop on Distributed Data and Structures, EPF Lausanne, Switzerland, July 8-9, 2004.

[8] T. Oh-ishi, K. Sakai, H. Matsumura, and 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.

[9] 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.

[10] SSFNet. http://www.SSFNet.org.

[11] R. Chalmers, K. Almeroth, *Developing a Multicast Metric*, 1998.

[12] P. Van Mieghem, G. Hooghiemstra, R. Van Der Hofstad, *On the Efficiency of Multicast*, IEEE/ACM Transac. on Networking, Vol.9, No.6, December 2001.

[13] R.C. Chalmers and K.C. Almeroth, *Modeling the Branching Characteristics and Efficiency Gains in Global Multicast Trees*, IEEE Infocom 2001.

[14] R.C. Chalmers and K.C. Almeroth, *On the Topology of Multicast Trees*, IEEE/ACM Transac. on Networking, Vol.11, No.1, February 2003.

[15] S. Fahmy and M. Kwon, *Characterizing Overlay Multicast Networks and their Costs*, IEEE/ACM Transac. on Networking, 2007, in print.