Delay Analysis of Application Level Multicast on Content Addressable Networks

Sugata Hazarika, Don Towsley
Department of Computer Science
University of Massachusetts, Amherst
{hazarika, towsley}@cs.umass.edu

Abstract—Two approaches have been proposed for application level multicast on Content Addressable Networks (CANs), implemented through tree-based and flooding-based methods. In this paper, we present simple analytical models for studying these two approaches. Using these models we study the performance achieved by multicast groups as a function of their size, the variability in delays between neighbor nodes and the dimensionality of the underlying CAN. We see that at lower dimensions and lower group sizes the CAN flooding scheme performs better than the tree-based scheme. However as the dimensionality of the CAN increases and the group sizes increase the tree-based scheme performs better. We also propose and study a scheme for reducing delays in the tree based multicast scheme.

I. INTRODUCTION

The recent past has seen the rapid emergence of DHTs (Distributed Hash Tables) like CAN [1], Chord [8], Pastry [4] and Tapestry [7] as fundamental building blocks for distributed applications. Even though originally motivated to make the search process in peer-to-peer networks scalable, today's structured DHTs are being increasingly used to support data dissemination applications.

The lack of deployment of IP Multicast has also in recent years led to an upsurge of interest in application level multicast. Application level multicast is increasingly becoming attractive as an implementable way of building data dissemination mechanisms to members of a large group. There are numerous schemes for doing application level multicast such as NICE [9], Narada [10], and schemes operating on DHTs (Distributed Hash Tables) which include Scribe [5], Bayeux [7] and CAN based multicast [2].

The two major mechanisms in application level multicast are core-based trees and many to many source specific data dissemination. In the case of DHTs, examples of doing multicast using core-based trees are Scribe and CAN based trees. CAN also implements a flooding-based many to many data dissemination approach. However, it needs to be pointed out here that, as application level multicast is still an emerging field there is a lack of studies that give an in depth analytical look at the the various problems and issues involved. This

This research has been supported in part by the NSF under grant awards ANI-0085848, EIA-0087945, EIA-0080119. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

paper is an attempt to fill the vacuum. In particular, we ask the question which mode of doing application level multicast gives the lowest delay corresponding to a multicast group size.

In this paper we develop analytical models for flooding based and tree based multicast on CANs (Content Addressable Networks). The primary reason for choosing the CAN as the basis of analysis is because CANs seamlessly support both types of multicast. This paper attempts to resolve the fundamental issue of which type of multicast is more feasible through analysis. The focus of the paper is to provide simple analytical models that can be used to capture the delays for application level multicast on CANs. Even though we base our analysis on CANs, our results have wide ranging implications for application level multicast on all DHTs. To the best of our knowledge this is the first detailed analytical evaluation of DHT based application level multicast. To compute the average delay for doing tree-based multicast, we introduce a recursive delay equation to calculate the expected end-to-end delay of a search process between any two nodes on the CAN. The relevance of our work arises from the fact that it provides exact values of average delay for the different multicast schemes and paints an unbiased picture of the behaviour of the different schemes. Our work also shows the potential of analytical models to understand better the performance characteristics of DHTs.

We see that at lower dimensions CAN flooding fares better than the tree-based approach but at higher dimensions the tree-based approach gives lower delay. This result is surprising because on an average, the number of hops in the flooding approach is half of the number of hops in tree based multicast. The reason for this is that the routing mechanism in the tree based approach allows the nodes to select the neighbor with lowest delay from multiple possible neighbor choices. This results in multiple possible paths between any two nodes on the CAN. On the other hand, in CAN flooding the path between any two CAN nodes is fixed, the formation of the path being dictated by the CAN flooding rules. At higher dimensions, the multiplicity of possible paths increases and so tree based multicast gives lower delay than CAN flooding. Besides that the average delay in the tree-based approach is independent of the multicast group size and depends only on the size of the CAN overlay while in CAN flooding average delay increases with group size.

The average delay for tree-based multicast depends on the variability of delays between two neighboring nodes on

CAN. As the variance of the distribution increases the delay decreases. In the flooding scheme the delay is independent of the variance of this distribution.

We also present an optimization scheme for reducing delay in tree based multicast on CAN.

The rest of the paper is organized as follows. Section II presents related work. Section III builds the basic analytical model for end-to-end delay between any two nodes on the CAN. Section IV presents an introduction to CAN based application level multicast. Section V analytically compares the modes of doing application level multicast on CAN. Section VI presents experimental results and section VII presents the optimization scheme in tree based multicast. We present our conclusions and future work in section VIII.

II. RELATED WORK

CAN was first introduced in [1]. It is a logical d dimensional hyperspace where every node owns its own zone. Every node maintains a neighbor table with the addresses of the adjacent nodes in the CAN. The process of nodes joining and leaving the CAN, as well as that of routing in the CAN is explained in detail in [1].

Application level multicast on the CAN was first introduced by Sylvia Ratnasamy et al in [2]. In this scheme a bootstrap CAN is used and all the nodes are members of this bootstrap CAN. For every multicast group a separate CAN overlay called the “mini CAN” is set up.

Tree-based application level multicast for DHTs was first introduced in Pastry [5] and Tapestry [7]. Tree based multicast is implemented in the form of core based trees and source specific trees. In both schemes, the receivers of the multicast group search for the core of the group or for the source of multicast using the underlying DHT. The reverse of the search path becomes the data dissemination path of the tree.

In [3], the authors adapted tree-based multicast to operate on a CAN. Instead of forming a separate CAN overlay for each multicast group, tree-based multicast forms a core based tree for each multicast group. The authors also compared tree-based application level multicast with the mini CAN flooding approach through simulation. We will resort to analysis for our purpose as this yields a deeper understanding about the sensitivity of the performance of the two approaches to variations in delay.

III. END TO END DELAY ON CAN

A. Content Addressable Network - CAN

CAN is a d -dimensional hypercube. Each node takes ownership of a specific hyper-rectangle such that the entire space is covered. Each node maintains a neighbor table with the addresses of its immediately adjacent neighbors in the CAN. When a node wishes to join the CAN, it routes a join message to a random point in the hyper-space. The node owning the hyper-rectangle containing that point splits it between itself and the new node. A node routes a message by forwarding it to a neighbor closer to the destination than itself. This process is repeated till the message reaches the destination.

For a d dimensional space partitioned into n equal zones, the average routing path length is $(d/4) \sqrt[n]{n}$ hops. Individual nodes maintain approximately $2d$ neighbors.

CAN has a number of optimizations, that can be used to improve routing performance as detailed in [1]. We include the following CAN optimizations, which the original authors recommend, and which have been proved to perform better.

- **Network Delay Routing (NDR):** Ordinary CAN routing chooses the neighbor closest to the destination in the CAN space. NDR has been proposed as an optimization to improve delay, it chooses the neighbor with least network delay, subject to the constraint that the message moves closer to the destination at each hop. In [3], it was found that NDR always performed better than original CAN routing.
- **Uniform Partitioning:** Uniform partitioning attempts to maintain uniformly sized CAN zones for each node in the CAN hyperspace. In uniform partitioning, when a new node joins the CAN, the node already responsible for that CAN zone does not immediately partition its zone but does a local search among its neighbors to find the node that has the largest zone. This node, then partitions its zone. A similar approach is taken to handle departures, in merging the zone of the departed node. In our model, we assume a perfectly uniform CAN. As we will see in the following sections, a uniform CAN helps illustrate clearly the effect of redundant paths on delay in application level multicast on CAN.

We also make the following additional assumptions. Let $L = \{l_1, l_2, \dots, l_N\}$ denote the set of overlay links, where $\forall j, l_j$ denotes an overlay link between two neighboring nodes on CAN. Let $\{X_k\}_{k \in L}$ be the set of iid random variables denoting delay over the set of overlay links L ; $\forall k, X_k \in \mathbb{R}$. Given a set of iid random variables $\{X_k\}_k^N$ we define $X_{i:j} (1 \leq i \leq j \leq N)$, to be the i^{th} smallest value of j samples drawn from $\{X_k\}$.

Fig. 1. Search in a 2 dimensional CAN

B. Multiple Possible Paths

The two dimensional CAN in Figure 1 illustrates how network-delay routing (NDR) works. In the figure we ignore the fact that in a CAN messages wrap around as the CAN is a d dimensional torus. NDR forwards the message

to the neighbor with the least network delay as long as the message moves closer to the destination. In the context of Figure 1, suppose node A wishes to route a message to node F. According to NDR, node A routes the message to one of its two neighbors, node B or node C, whichever has lower delay. This rule is applied by node C if the message reaches it. In case of nodes B,D and E there exists only one link to forward the message on. Node B has to forward the message to node D, and nodes D and E have to forward to node F. That is because, if nodes B, D and E forward the message to any other node then the message will move farther away from the destination than its current location. This example clearly illustrates that NDR leads to multiple possible paths between a source and a destination. In our example the possible paths are ABDF, ACDF, ACEF. This redundancy of paths arise due to the fact that some nodes have a choice of more than one neighbor to forward the message to. In a perfectly uniform CAN every CAN node has $2d$ neighbors. The number of neighbors increase as the dimensionality of the CAN increases and so the choice of nodes to which the message can be passed also increases with increasing dimension.

C. Recurrence Relation for End-to-End Delay

Consider the transfer of a message between any two nodes on a d dimensional CAN, so that the source has coordinates $(0, 0, \dots, 0)$ and the destination has coordinates $(i_1, i_2, \dots, i_d)$. Let $R(i_1, i_2, \dots, i_d)$ be the random variable which denote the end-to-end delay between $(0, 0, \dots, 0)$ and $(i_1, i_2, \dots, i_d)$. Let $M \subset L$, $M = \{1, 2, \dots, d\}$ denote the set of overlay links from $(0, 0, \dots, 0)$ to its neighbors in the direction of $(i_1, i_2, \dots, i_d)$. Let $\bar{R}(i_1, i_2, \dots, i_d)$ be the expected value of $R(i_1, i_2, \dots, i_d)$. Here $I(P)$ indicates an indicator random variable.

$$\begin{aligned} \bar{R}(i_1, i_2, \dots, i_d) &= E[X_{1:n(i_1, i_2, \dots, i_d)}] + \\ &\quad \frac{1}{n(i_1, i_2, \dots, i_d)} \sum_{j=1}^d \\ &\quad (\bar{R}(i_1, \dots, i_{j-1}, \\ &\quad (i_j - 1)^+, i_{j+1}, \dots, i_d) \quad (1) \\ n(i_1, i_2, \dots, i_d) &= \text{number of non-zero} \\ &\quad \text{arguments towards} \\ &\quad (i_1, i_2, \dots, i_d) \\ &= \sum_{j=1}^d I(i_j \neq 0) \\ k^+ &= \max(0, k) \\ \bar{R}(0, \dots, 0, i_j, 0, \dots, 0) &= i_j E[X_k] \end{aligned}$$

The recurrence equation developed in this section forms a building block to evaluate the performance of different application level multicast techniques. Note that this recurrence relationship is completely general in nature and works with any distribution for $\{X_k\}$.

IV. APPLICATION LEVEL MULTICAST ON CAN

The different modes of doing application level multicast on CAN are:

A. Core Based Trees

CAN can be used as an infrastructure for doing application level multicast using core-based trees. In the core-based tree, henceforth referred to as the tree-based approach of doing multicast, a multicast tree is formed for each multicast group. Associated with each group is a node which acts as the core of the multicast tree for that group. The core node is the node whose nodeid corresponds to the value of the hash function applied to the group name. Any node that wants to subscribe to a particular multicast group hashes the group name and then searches for this hashed value in the CAN. The search terminates at the CAN node whose nodeid is numerically closest to this hashed value and this node becomes the core for that multicast tree. The path of the search forms the path from the subscriber to the core, and the reverse of this path, the path of the distribution tree from the core to the subscriber. The collection of all these search paths form the distribution tree. Any message that a subscriber sends first travels to the core and then the core distributes it over the tree. A multicast tree is illustrated in a 2 dimensional CAN in Figure 2. In the figure we ignore the wrap around in the CAN.

Fig. 2. Core Based Tree in a 2d CAN

B. CAN Flooding

Can flooding is the alternative approach of doing multicast on the CAN. In the CAN flooding approach the members of the CAN that want to join a particular multicast group first form a specific “mini CAN” which is a separate CAN overlay containing only the members of that multicast group. Multicasting is done by flooding over the mini CAN.

There is a bootstrap CAN which contains all the nodes belonging to the CAN. When a particular node wants to join a particular multicast group it hashes the group name and searches for this hashed value in the bootstrap CAN. The search terminates at the CAN node whose nodeid is closest to this hashed value. This node then acts as the rendezvous node for that group. The rendezvous node for the group creates a separate CAN for that group, and then directs the incoming

node to this separate mini CAN. The incoming node then searches for its zone in the mini CAN.

Any of the nodes on the mini CAN can flood to all other nodes on it. In [2], the authors suggest an approach to flooding that reduces duplicates by a large amount. In fact, in a perfectly uniform CAN, the number of duplicates is zero.

V. DELAY COMPARISON: CORE BASED TREES AND CAN FLOODING

In this section we compare the average delay of the two types of CAN based application level multicast described in the previous section. The set of iid random variables $\{X_k\}_{k \in L}$ denote overlay link delays and was introduced in section III.

A. Core Based Trees

The recurrence relation given by equation (1) allows us to calculate the expected end-to-end delay between any pair of nodes in the CAN. As we have seen in the previous section, the path of the multicast tree from each of the subscribers to the core is formed by a search for the core from that subscriber on the CAN. Assuming that this path is symmetric, then the expected end-to-end delay from the subscriber to the core is the same as that from the core to the subscriber and can be calculated using equation (1). To calculate this expected delay, the recurrence relation can be applied to every subscriber, assuming that the subscriber is the source and the core is the destination, or the core is the source and the subscriber is the destination. In Figure 2, the expected delays from the source S1 to each of the receivers S2, S3 and S4 is the sum of the expected delay from S1 to the core C and the expected delays from the core C to S2, S3 and S4. The average of the expected delays from the source to all the receivers yield the average expected delay on that multicast tree. As can be seen in Figure 2, group nodes share common path segment to the core and, thus their delays are correlated. However, since we are only interested in the expected end-to-end delay, this can be ignored.

Let us consider a multicast group of size m in an n node CAN. There are C_m^n possible multicast trees with m receivers in the n node CAN. Let $T = \{T_1, T_2, \dots, T_k\}$, $k = C_m^n$, define the family of multicast trees having group size m in the CAN. $\forall T_i \in T$, let $R_i = \{r_{i1}, r_{i2}, \dots, r_{im}\}$ denote the set of receivers in T_i . We define $Y_{i[j]}$, $j = \{1, 2, \dots, m\}$ as the set of random variables denoting end-to-end delay from the core to the j^{th} member of the receiver set R_i . We define $\bar{Y}_i$ as the average of $Y_{i[j]}$, that is $\bar{Y}_i = \sum_{j=1}^m Y_{i[j]} / m$. The average delay of multicasting on tree T_i is $2\bar{Y}_i$. We define $\bar{Y}$ as the average delay of multicasting for T , the family of trees having group size m . $\bar{Y} = \sum_{i=1}^{C_m^n} 2\bar{Y}_i / C_m^n$. We define $\bar{X}$ as the average end-to-end delay between any two nodes on the CAN. The instantaneous end-to-end delay between any two CAN nodes i and j is defined as d_{ij} .

Theorem 1: The average delay of multicasting, averaged over all trees having group size m , $\bar{Y}$, is equal to $2\bar{X}$, that is twice the average end-to-end delay between any two random nodes on the CAN.

Proof:

$$\begin{aligned} \bar{Y} &= \frac{\sum_{i=1}^{C_m^n} 2\bar{Y}_i}{C_m^n} \\ &= 2 \frac{\sum_{i=1}^{C_m^n} \bar{Y}_i}{C_m^n} \\ &= 2 \frac{\sum_{i=1}^{C_m^n} \sum_{j=1}^m Y_{i[j]}}{m C_m^n} \\ &= 2 \frac{\sum_{i=1}^{C_m^n} \sum_{j=1}^m d_{ij}}{m C_m^n} \end{aligned}$$

This equation can be rewritten in another way. A particular receiver, say j can be taken as fixed, and the average delay to that receiver from the cores of all possible trees of size m in which it appears is computed. Node j is equally likely to appear in C_{m-1}^{n-1} trees.

$$\begin{aligned} \bar{Y} &= 2 \frac{\sum_{i=1}^n \sum_{j=1}^{C_{m-1}^{n-1}} d_{ij}}{n C_{m-1}^{n-1}} \\ &= 2\bar{X} \end{aligned} \quad (2)$$

B. CAN Flooding

Let us consider a mini CAN having group size m . Let s be the source of multicast on the mini CAN and $R = \{r_1, r_2, \dots, r_{m-1}\}$ be the set of receivers. Let h_i be the number of hops from s to r_i , $\forall i \in R$. Let $\bar{Y}$ be the average delay of multicasting on the mini CAN. We define $\bar{X}$ as the average end-to-end delay between any two nodes on the CAN.

In a perfectly uniform CAN, the CAN flooding approach does not create any duplicates. That means that between s and r_i , $\forall i \in R$ there is only one path, dictated by the flooding rules. If there is only one possible path, NDR cannot be used. Let s have coordinates $(0, 0, \dots, 0)$. Let us consider a receiver r_i with coordinates $(i_1, i_2, \dots, i_d)$. Then h_i for that receiver is given by,

$$h_i = i_1 + i_2 + \dots + i_d \quad (3)$$

The average delay of multicasting on the mini CAN is defined by,

$$\bar{Y} = \sum_{i=1}^{m-1} h_i \bar{X} \quad (4)$$

VI. EXPERIMENTAL EVALUATION

In order to evaluate average delay for the two different multicast algorithms, we need to assume an overlay link delay distribution, X_k . The random variable denoting overlay link delay, is either drawn from the gamma distribution where $f_{X_k} = \text{GAMMA}(c, \gamma, \beta)$, or from a two stage hyperexponential distribution where $f_{X_k} = \text{HYP}(2, \lambda)$.

Gamma Distribution: The mean, variance and coefficient of variation (CV) of the gamma distribution are γ/β , γ/β^2 and $\gamma^{-\frac{1}{2}}$. We always set $\beta = \frac{1}{\gamma}$ to ensure that the expected value is normalized to 1. The pdf of the distribution is

$$\text{GAMMA}(c, \gamma, \beta) = \frac{1}{\beta \gamma \Gamma(\gamma)} c^{\gamma-1} e^{-\frac{c}{\beta}} \quad (5)$$

Fig. 3. Dimensionality Walk: For dimension 3, 6, 7 and 10

Note: $\text{GAMMA}(c, \gamma, \beta)$ with $\text{CV} = 1$ and expected value = 1 is the exponential distribution with an expected value of 1. In equation (1), the term $E[X_{1:n}]$ has a closed form if X_k is drawn from an exponential distribution. For an exponential distribution with rate λ ,

$$E[X_{1:n}] = \frac{1}{n\lambda} \quad (6)$$

Two stage Hyperexponential: A two stage hyperexponential distribution can be used to generate any coefficient of variation (CV) greater than 1. With a probability of α X_k is chosen from the first stage, and with a probability of $\bar{\alpha} = 1 - \alpha$ it is chosen from the second stage. By varying the values of α and $\bar{\alpha}$ the CV can be varied. In case of this distribution, we can obtain a closed form for $E[X_{1:n}]$.

$$E[X_{1:n}] = \sum_{i=0}^n C_i^n \alpha^i \bar{\alpha}^{n-i} \frac{1}{i\mu_1 + (n-i)\mu_2} \quad (7)$$

where

μ_i = rate of exponential stage i .

A. Sensitivity to dimensionality

In Figure 3 we plot the average delay of multicasting, $\bar{Y}$ for tree-based multicast and CAN flooding as a function of group

size and dimensionality of the CAN. We generate CANs of sizes ranging from 100 to 10000.

In the dimensionality walk we plot the following quantities,

- *treeaverage*: average delay for the tree-based scheme; $\bar{Y}$.
- *treemax*: maximum expected delay of all trees of a given group size. For a group size m , *treemax* is $\bar{Y}_{m:m}$.
- *treemin*: minimum expected delay of all trees of a given group size. For a group size m , *treemin* is $\bar{Y}_{1:m}$.
- *flood*: average delay of the CAN flooding algorithm; $\bar{Y}$.

Calculation of $\bar{Y}_{m:m}$ and $\bar{Y}_{1:m}$: The following procedure is used to calculate $\bar{Y}_{m:m}$ and $\bar{Y}_{1:m}$ for tree-based multicast.

- In an n node CAN, choose a core randomly.
- Let $\bar{D}_i$ be the expected end-to-end delay of the node which has the i^{th} minimum expected end-to-end delay from the core.
- Let the group size be m .
- Then,

$$\bar{Y}_{1:m} = \frac{m\bar{D}_1 + \sum_{i=1}^m \bar{D}_i}{m} \quad (8)$$

$$\bar{Y}_{m:m} = \frac{m\bar{D}_n + \sum_{i=n-m+1}^n \bar{D}_i}{m} \quad (9)$$

The main conclusions that can be drawn are,

- Average Delay for tree-based multicast is a constant depending only on the size of the CAN and is independent of group size.
- As regards the comparison of CAN flooding with tree-based multicast, CAN flooding performs better than tree based multicast at small dimensions, e.g. $d = 2, 3$. In these dimensions all groups have lower average delay for flooding than tree-based multicast. However as the dimensionality increases the number of possible paths increase as well, and so NDR helps the tree based scheme to achieve lower average delay than CAN flooding. At dimensions 6 and 7, for more than 50% of the groups, trees fare better than CAN flooding. At higher dimensions, like 9 and 10, trees fare better than CAN flooding for more than 90% of the groups. However it is worth noting that in case the group sizes are small, that is they are of an order of a 1000 or less, CAN flooding is still a viable alternative, even at dimensions 9 and 10.

B. Variability of Overlay Link Delays

Little is known regarding the variability present in the Internet end-to-end delay. In [13], Amarnath Mukherjee observed that round trip times follow a gamma distribution. This belief was extended in [14], where in a recent study the authors found that the one way delay on fixed internet paths follow a gamma distribution. However, the exact parameter space of the gamma distribution is not known. Mukherjee in his 1992 study found that the coefficient of variation (CV) varied from close to 0 to as high as 6. In the absence of conclusive evidence, we examine how the average delays of the different approaches is affected by CVs. In this section we will look at a range of CV from .2 to 4.5.

Fig. 4. Tree based multicast against CAN flooding: CV less than 1

Figures 4 and 5 plot average delays of tree-based multicast and CAN flooding as a function of group size for different values of the coefficient of variation (CV) of the overlay link delay distribution for $d = 6$. The effect is the same at any d . The first observation is that average delay for CAN flooding

Fig. 5. Tree based multicast against CAN flooding: CV greater than 1

is not affected by the variance of the link delay distribution. This is because, this scheme is not affected by NDR. From both the figures, it is observable that as the CV increases the average delay for tree-based multicast decreases. At higher values of CV trees perform much better than flooding, and at lower values they perform worse than flooding. This is because, tree-based multicast is affected by NDR and equation (1), and $E[X_{1:n(i_1, i_2, \dots, i_d)}]$ decreases as CV increases.

VII. DELAY OPTIMIZATION ON CORE BASED TREES

A. Optimization

In this section, we present a technique for optimizing delay on core based trees on the CAN. It was originally suggested for IP Multicast by Ballardie et. al in [15]. In application level multicast a packet travels from the source to the core and then the core distributes the packet over the tree. This can be optimized so that as a packet traverses towards the core the forwarding nodes can directly copy it and forward it to receivers that are attached to them. This reduces the number of hops taken by the packet to reach these receivers. We evaluate this optimization below.

B. Results

We continue with the assumption that X_k , the random variable denoting overlay link delay is drawn from a $GAMMA(c, \gamma, \beta)$ distribution. We simulate the routing process and the setting up of multicast trees in a CAN simulator that we developed. It is difficult to analyze the delay of multicasting in this case using the recursive delay equation because some packets do not travel to the core, and so we resort to simulation. For every multicast group size the experiment is repeated 100 times and then averaged over all of them. Figure 6 presents the average delay for multicasting at $d = 10$. The trend of the result is similar for all values of d . It is seen that the optimized fares better.

Figure 7 presents the percentage reduction in average delay for the optimized tree as compared to the original tree-based multicast. The reduction in average delay in the optimized tree

Fig. 6. Core Based Tree Optimization

Fig. 7. Core Based Tree Optimization

is sensitive to the dimensionality of the CAN. It is seen that this reduction in delay is maximum at small d , e.g. $d = 3$. However, it decreases as d increases. At $d = 10$, it is only about 7 percent. This lowering of reduction in average delay with increasing d can be explained. As d increases the depth of the tree decreases, and so the number of forwarding nodes decrease. As a result of this, forwarding nodes that are on the path from the source to the core have fewer receivers attached to them, so fewer receivers receive the packet as it travels to the core. This leads to the reduction in benefit in delay.

VIII. CONCLUSION AND FUTURE WORK

In this paper, we showed analytically at how the delays for application level multicast behave in the context of the CAN. We see that at higher dimensions tree based multicast performs better than CAN flooding. This is because the network delay routing metric of CAN does not help in improving the delay of CAN flooding, as there are no redundant paths. Last we present an optimization scheme in tree based multicast that reduces delay.

In our future work, we will extend this framework to other DHTs. In section III, what we do is essentially model the search process on the CAN. The DHT search is at the heart

of many DHT applications. We have a very similar recursive model for Chord[8]. Tree-based multicast on Chord can be set up in a manner very similar to CAN. We have also developed a CAN-like zero-duplicate flooding algorithm on Chord. The next logical step, is to do a comparison of application level multicast between Chord and CAN. Last, the basic recursive model can be used to do other feature comparisons of DHTs.

REFERENCES

- [1] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, Scott Shenker, "A Scalable Content-Addressable Network", in Proceedings of *ACM SIGCOMM* 2001.
- [2] Sylvia Ratnasamy, Mark Handley, Richard Karp, Scott Shenker, "Application-level Multicast using Content-Addressable Networks", in Proceedings of *NGC* 2001.
- [3] M. Castro, M. B. Jones, A-M. Kermarrec, A. Rowstron, M. Theimer, H. Wang and A. Wolman, "An Evaluation of Scalable Application-level Multicast Built Using Peer-to-peer overlays", in proc. *Infocom* 2003, San Francisco, CA, April, 2003.
- [4] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems", in proc. *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, Heidelberg, Germany, pages 329-350, November, 2001.
- [5] A. Rowstron, A-M. Kermarrec, M. Castro and P. Druschel, "SCRIBE: The design of a large-scale event notification infrastructure", in proc. *NGC2001*, UCL, London, November 2001.
- [6] Ben Y. Zhao, John Kubiawicz and Anthony Joseph, "Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing", UCB Tech. Report UCB/CSD-01-1141
- [7] Shelley Q. Zhuang, Ben Y. Zhao, Anthony D. Joseph, Randy H. Katz, John Kubiawicz, "Bayeux: An Architecture for Scalable and Fault-tolerant Wide-Area Data Dissemination", in proc. *ACM NOSSDAV* 2001.
- [8] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications", in proc. *ACM SIGCOMM* 2001, San Deigo, CA, August 2001, pp. 149-160.
- [9] Suman Banerjee, Bobby Bhattacharjee, Christopher Komareddy, "Scalable Application Layer Multicast", in proc. *ACM Sigcomm*, August 2002.
- [10] Yang-hua Chu, Sanjay G. Rao, Srinivasan Seshan and Hui Zhang, "Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture", in proc. *ACM SIGCOMM*, San Deigo, CA, August 2001.
- [11] www.kazaa.com
- [12] www.gnutella.com
- [13] Amarnath Mukherjee, "On the Dynamics and Significance of Low Frequency Components of Internet Load", Technical Report, MS-CIS-92-83/DSL-12.
- [14] C.J Bovy, H.T Mertodimedjo, G. Hooghiemstra, H. Uijterwaal, P. Van Mieghem, "Analysis of End-to-end Delay Measurements in the Internet"
- [15] Tony Ballardie, Paul Francis, Jon Crowcroft "Core Based Trees : An Architecture for Scalable Inter-Domain Multicast Routing"