Hopcount in Application Layer Multicast Schemes

Milena Janic, Novi Ineke Cempaka Wangi, Xiaoming Zhou and Piet Van Mieghem
Delft University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
P.O. Box 5031, 2600 GA Delft, The Netherlands
{M.Janic, N.Ineke, X. Zhou, P.VanMieghem}@ewi.tudelft.nl

***Abstract*— Application Layer (AL) multicast emerged as a response to a slow deployment of IP multicast. However, the gain of AL multicast over unicast is questionable. Here, we investigate the efficiency of the two prominent protocols, MCAN and Scribe, in terms of the number of hops. We compare the efficiency of these algorithms to the efficiency of unicast and IP multicast via extensive simulations, as well as via measurements on the PlanetLab network. We introduce modifications to the MCAN algorithm that lead to a reduction in the hopcount. Finally, we demonstrate that the topology unawareness under certain conditions can make these schemes less efficient than unicast.**

I. Introduction

As a response to the slow deployment of IP multicast, the network layer solution to the group communication, numerous Application-Layer (AL) multicast proposals have appeared in the last few years [14][7][1][5]. In AL multicast packet duplication and routing is handled by the participating end hosts, instead of by the network routers in IP multicast.

The reduced complexity of AL multicast with respect to IP multicast comes at the expense of efficiency, because in AL multicast packets may traverse the same link several times. AL multicast only makes sense if its performance is better than that of unicast. Therefore, one of the crucial questions is "how efficient is application layer multicast?" AL multicast algorithms proposed so far all claim to be more efficient than unicast, and slightly less efficient than IP multicast. These claims are usually supported by a few simulation results. In most cases different algorithms are not compared under the same conditions, which complicates conclusions drawn. We take the viewpoint that understanding the performance of existing AL mcast protocols will teach us how to optimize and improve the best among them.

In this paper we focus on AL multicast schemes on top of overlay networks, due to their superior scalability. An overlay network is a virtual network constructed in the application layer. A number of structured peer-to-peer (p2p) overlays (e.g. CAN [6], Chord [11], Pastry [9] and Tapestry [13]) have emerged lately. These overlays can be used to provide AL multicast. AL multicast is then realized either by flooding (MCAN [7]) or tree building (Bayeux [14], Scribe [5]).

There are two main objectives in this paper:

1. To evaluate the performance of two scalable application layer multicast algorithms, CAN-based multicast (MCAN) [7] and Scribe [5]. As a performance metric the number of hops has been used. We compare these schemes mutually, as well as to multiple unicast connections (m-Unicast) and IP multicast. Furthermore, we introduce and evaluate modifications to MCAN that lead to a better performance in terms of the number of hops (duplicate packets). We perform this evaluation both via extensive simulations, as well as via experiments on PlanetLab[1].

2. To investigate the influence of the underlying topology awareness on the performance of application layer multicast. In the early approaches, overlay networks have been created without considering the underlying Internet topology. However, if an overlay network is built in this manner, then the two neighboring nodes in the overlay may be separated by many hops on the IP layer. Consequently, even if optimal in the number of application layer hops, a path between the source and the destination on the overlay may consist of a large number of IP-layer hops. Achieving the congruency of the overlay with the underlying IP-layer network in the optimal way is not a trivial problem. Several methods have been developed for realizing the partial topology awareness, and they can be classified in three groups proximity routing, topology-based *nodeId* assignment, and the proximity neighbor selection. They will be described in the following Sections. In this paper, for each of the implemented algorithms, we have considered two extreme situations: no topology awareness, and absolute topology awareness. Whereas under the no topology awareness conditions nodes in overlay are placed randomly, without taking the underlying IP-layer topology into account, the term absolute topology awareness indicates the ideal situation in which the complete knowledge of the underlying substrate is attained and used in the creation of the overlay.

To the best of our knowledge, the only study similar to ours has been provided by Castro *et al.* [3]. However, our study differs from [3] in several aspects. First, some argue ([8][5]) that for an efficient operation of their protocols, the indications about the position of users suffice, without the need for the exact topological information. In this paper however we do not attempt to estimate the underlying topology. Instead, we assume that the underlying topology is either completely known to the joining overlay nodes (absolute awareness) or completely unknown. In this way, we try to establish the upper and lower boundaries for the hopcount of these two algorithms. Secondly, we use the hopcount as a metric, since it is an important quantity from a network point of view. Third, we perform our simulations on a very large number of different substrates (up to 10^5) which ensures the statistical credibility

[1] http://www.planet-lab.org/

of our results. Fourth, we introduce the modifications to MCAN algorithm that lead to improved results for hopcount compared to the original algorithm. Finally, we evaluate our results via experiments on the PlanetLab network, which to the best of our knowledge has not been done so far.

The paper is organized as follows: Section II provides the brief description of the CAN overlay network and MCAN, and introduces our improvements to the CAN-base multicasting algorithm. In Section III Pastry overlay network and Scribe are described. In Section IV we first explain the simulation designs of five compared schemes: MCAN with and without improvements, Scribe, m-Unicast and IP multicast. The same Section further presents and discusses the simulation results for different types of the underlying topology. In Section V the results of the measurements on PlanetLab have been presented. Finally, we conclude in Section VI.

II. CAN-BASED MULTICAST

A. CAN overlay network

The nodes in a Content Addressable Network (CAN) are identified in a d-dimensional Cartesian coordinate space on a $(d+1)$-torus. Every node in a CAN holds an individual zone, determined by its coordinates. Each CAN node learns and maintains IP addresses of his neighbors, i.e. nodes whose zones are adjacent to its own zone[2]. The overlay is formed in the following way: a joining node first discovers a node already participating in CAN. Next, it chooses randomly a point in the CAN space. The zone in which this point lies is then split in two: one half is kept by the current holder of that zone, and the other half is allocated to the new node. Both nodes update their routing table and this change is subsequently announced to their neighbors. The routing of the message in CAN is performed in a greedy forwarding fashion: the packet is sent to that neighbor with the coordinates closest to the coordinates of the destination.

B. Topology Awareness in CAN overlay network

Originally, in the CAN overlay nodes were allocated to zones at random, such that there is no relation between node coordinates and the underlying topology. Hence, neighbors on a CAN are not necessarily physically near each other in the Internet. The topology awareness technique implemented in CAN is coined topology-based nodeID assignment. The designers of CAN proposed in [8] a landmark-based placement for accounting for the underlying topology when creating CAN. Each joining node probes a set of landmark machines, estimates network distances by measuring the round-trip-time and *orders* the RTTs in increasing order. The CAN space is then divided into evenly sized bins and the split node is randomly chosen within the bin area. These bins are clusters of nodes with same landmark ordering. Note that even though the underlying substrate is accounted for, the reflectance of the underlying topology on the overlay is partial, due to the incomplete knowledge of the underlying network.

[2]Two nodes are neighbors if their zone coordinates overlap along all but one dimension and abut in one dimension.

C. CAN based Multicast (MCAN)

The simplest way to achieve multicasting in CAN is to perform flooding over the CAN space. Since the flooding induces a large number of duplicate packets, a more efficient forwarding algorithm has been proposed in [7], that we refer to as MCAN1. Nonetheless, even the improved algorithm generates a substantial number of duplicates. Therefore, we introduce the modifications hat lead to a further reduction of duplicates. Here, we describe the resulting modified algorithm, that we refer to as MCAN2. For the description of the original MCAN1 algorithm, we refer to [7]. We explain the MCAN2 algorithm on an example of a two-dimensional CAN, and illustrate it Figure 1.

1) Multicast CAN Modified Forwarding Algorithm (MCAN2)*:*

Origin forwarding rule: The source that generates a message forwards the message to all its neighbors along the x-axis, *but only one neighbor per direction along the y-axis.* In Figure 1(a), A forwards the message to C and N along the x-axis, but only to H and only to E in positive and negative direction of y-axis respectively.

General forwarding rule: A node that receives the message along the y-axis, forwards the message to all the neighbors along the x-axis. However, it will forward the message to only one neighbor in a particular direction along the y-axis (the direction away from the source). If there are several neighbors in that direction, the neighbor to which the message will be sent is chosen randomly. A node that receives the message along the x-axis, forwards it to all the neighbors along the x-axis that lie in the direction *opposite* of that from which the node received the message (e.g. node C and node D in Figure 1(b)).

The halfway rule and the cache suppress rule remain unmodified:

The halfway rule: A node does not forward a message along a particular direction if the message has already propagated halfway across the space from the source coordinates along that dimension. This rule prevents the flooding from looping round the back of the space.

Cache suppress rule: A node caches the sequence numbers of received messages and does not forward the same message more than once.

The corner filter rule: In the modified algorithm, the corner filter rule is applied only on forwarding along the x-axis. In Figure 1(d) M only receives the message from J since M's corner is in contact with J's zone. However, in the y-axis forwarding, even though E's zone does not touch F's corner, E will forward the message to F.

III. SCRIBE

A. PASTRY

In the Pastry overlay network, each node has a unique identifier (*nodeId*). The identifiers are chosen out of the 128-bit circular space and assigned to the nodes. *NodeId*s are presented as a sequence of digits in base 2^b (b is a configuration parameter with typical value 2 or 4).

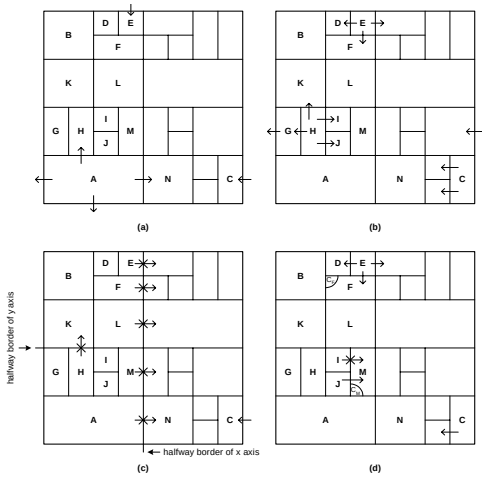

Fig. 1. The modified multicast forwarding algorithm in CAN (MCAN2)

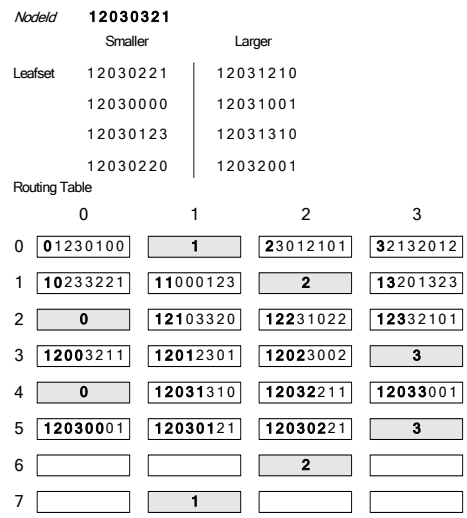

Fig. 2. Leaf set and Routing table for node in Pastry with *nodeId* 12030321. Depending on the destination key, the node choses the next hop from the routing table according to the longest prefix rule. E.g., for a destination keys 31032030 and 12030000, the next hop would be 32132012 (row 0 column 3 in routing table) and 12030000 (in leaf set) respectivelly.

To route a message, a node uses a leaf set and a routing table. A leaf set consists of typically $2b$ or $2 \times 2b$ entries, which are filled with *nodeId*s numerically closest to the current node's *nodeId*. A routing table consists of $128/b$ rows, with $2b$ columns per each row. The routing table is constructed in the following way: the entry in the n-th row and k-th column of a node's routing table contains a *nodeId* that matches the current node's *nodeId* on the first n digits, and its $(n+1)$-th digit is equal to k. If there is no *nodeId* that satisfies this criterion for an entry, the entry is left empty. An example of a routing table for a node with *nodeId* 12030321 is illustrated in Figure 2.

The routing in Pastry is performed as follows: upon receiving a message, a node first searches for the destination key in its leaf set. If the leaf set does not contain that key, the node starts searching in its routing table for the node whose *nodeId* matches the destination key in at least one digit more than the current node does. If such a node is not found in the routing table, the current node will look back into its leaf set and forward the message to a node whose *nodeId* matches the destination key in the same number of digits as the current node, but is numerically closer to the key.

If the number of Pastry users is sufficiently large, there might exist several nodes that meet the criterion for an entry. The node will then be chosen randomly out of all candidates satisfying the criterion. Alternatively, a proximity neighbor selection technique can be applied: an entry in the routing table includes the *nodeId* of the node topologically nearest to the current node, of all possible candidates. In the original Pastry design, Rowstron *et al.* [9] assumed that a function exists that enables each Pastry node to determine its relative "distance" to another node, given the node's IP address.

B. SCRIBE

Scribe is a tree-based application layer multicast mechanism built on top of Pastry. Scribe builds a single multicast tree for the whole group. Each group has a unique *groupId* and a root of the group tree. The tree is created by combining Pastry paths from each group member to the tree root. A node wishing to send a message to the group delivers the message first to the root. From the root, multicast messages are then distributed along the multicast tree.

IV. Evaluation via Simulations

In our simulations, we confine ourselves to random graphs of class $G_p(N)$ [2] with N nodes, and independently chosen links with probability p, and link weights equal to 1. We first generate a graph consisting of $N \geqslant 100$ nodes, representing the routers. For each graph of N nodes, we define the number of multicast users m in the network, such that a ratio $\rho = m/N$ lies in the set $\rho = \{0.05, 0.1, 0.2, 0.5, 0.7, 0.9\}$. For each N, 10^5 different topologies are generated. One node was designated as a source node.

Two different scenarios for the members' location have been considered: in the **scenario a**, some multicast members (or Pastry nodes) may belong to the same router, while in the **scenario b**, each member is attached to a different router.

In each underlying topology, the following six different multicast schemes have been implemented: multiple-connections unicast, IP multicast, the original MCAN with and without the topology awareness (M-CAN1 and M-CAN1top), the modified MCAN with and without topology awareness (M-CAN2 and M-CAN2top), Scribe without the topology awareness (Scribe1) and Scribe with the absolute topology awareness (Scribe2).

For each mechanism and each underlying topology, the number of hops in the path is computed and stored in a histogram. In a multiple-connections unicast, the total hopcount is the sum of the hopcounts along shortest paths from a source to each of the $m-1$ destinations individually. For IP multicast, we assumed that the message is disseminated

along the Shortest Path Tree[3], since the most of the current IP multicast routing protocols forward packets based on The (Reverse) Shortest Path. The total hopcount is equal to the sum of the links in a multicast tree.

For MCAN, in addition to the total number of hops traversed, an effective number of hops has been computed and stored. An effective number of hops is computed by only including the hops traversed in order for all destinations to be reached, without including the forwarding paths of the duplicate messages.

A. MCAN on a random graph

We have assumed for simplicity that all nodes in a CAN form a multicast group, and we created a two-dimensional CAN. In order to evaluate MCAN on top of a random graph, three different sets of simulations have been performed:

(i) Single overlay: in this set of simulations, no topology awareness is included. For each underlying topology and a position of users, the CAN overlay structure is kept the same. Hence, after a CAN overlay consisting of m MCAN members is generated, the m MCAN members are mapped onto each of the underlying topology in the following way: in **scenario a**, each MCAN member is mapped to the one out of N nodes in the underlying network randomly. In this scenario, the same node can be chosen multiple times, i.e. several MCAN members can be attached to a same router. In **scenario b**, m out of N nodes in the underlying network are chosen randomly, and each of the m MCAN members is assigned to one of them. Hence, each member is attached to a different router. For each scenario, and each underlying topology, all 6 schemes have been implemented, and their hopcounts have been stored in histograms. From each histogram, the probability density function of the hopcount was deduced, together with the mean $E[H_N]$ and the variance $var[H_N]$.

(ii) Multiple overlays: For each underlying topology $r = 1000$ different samples of a CAN overlay, each consisting of m members, have been generated. For each of the overlay instances, the identical procedure as described in (i) has been applied.

(iii) Topology aware overlay: For each m and N, and both scenarios a and b, in each of the generated underlying topology, the identical nodes (routers) to those chosen in (i) have been selected. CAN overlay is here constructed using the information about the distances in the underlying network. Each newcoming node learns the hopcount to all the other nodes already in CAN using the Dijkstra algorithm. The nearest node is then chosen to be the split node.

In Table I we summarize the parameters used in simulations. In the remainder of this Section we present the results of our simulations.

[3]A Shortest Path Tree is the union of the shortest paths between the source and all m destinations.

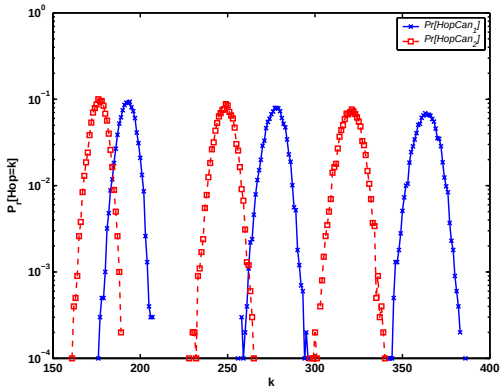

Fig. 3. The pdfs of MCAN1 and MCAN2 in a single CAN overlay (schemes 4-6 and 10-12), $N = 200$, $\rho = 0.5, 0.7, 0.9$, scenario a

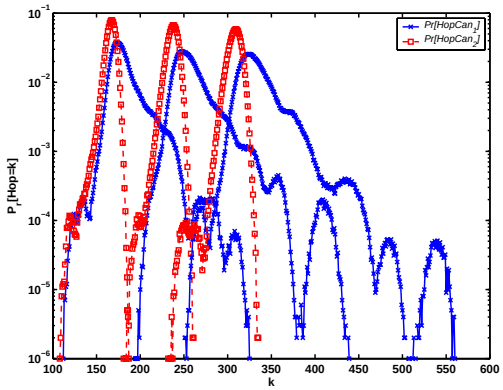

Fig. 4. The pdfs of MCAN1 and MCAN2 in multiple CAN overlay (schemes 16-18 and 22-24), $N = 200$, $\rho = 0.5, 0.7, 0.9$, scenario a

1) The pdf of hopcount in MCAN: Figure 3, Figure 4 and Figure 5 present the probability density functions (pdf) of the hopcount of both M-CAN1 and M-CAN2 algorithms in a single (schemes 1-12), multiple (schemes 13-24) and a topology aware (schemes 25-36) CAN structure respectively, for $N = 200$, scenario a, and high values of ratio $\rho = m/N$. Figure 3 suggests for simulations on a single overlay for both MCAN1 and MCAN2 a similar, bell-shape form of pdf. However, the average hopcount of MCAN2 is up to 11% lower than that of MCAN1 with the increase of m (ρ). In Figure 4 and Figure 5 we observe another interesting phenomenon: a "clustering" effect for the hopcount in MCAN1. These Figures, particularly Figure 4, indicate that the values of hopcount of MCAN1 in multiple overlays (schemes 13-18), for both scenario a and b, concentrate around several dominant values. This suggests that there seems to exist a finite number of "groups" of CAN structures.

This behavior is observed in the topology-aware CAN as well (scheme 25-30), however the number of values around which hopcount concentrates is lower. This phenomenon is not reflected in the hopcount of MCAN2, as can be observed from same Figures. The pdf of MCAN2 is bell-shaped, with a variance much lower variance of MCAN1. The average hopcount of MCAN1 is again 10% higher than that of MCAN2.

sch.	MCAN	m/ρ	$1/2$	**sch.**	MCAN	m/ρ	$1/2$	**sch.**	MCAN	m/ρ	$1/2$
1	single	$10/0.05$	1	**13**	multiple	$10/0.05$	1	**25**	top. aw.	$10/0.05$	1
2	single	$20/0.1$	1	**14**	multiple	$20/0.1$	1	**26**	top. aw.	$20/0.1$	1
3	single	$40/0.2$	1	**15**	multiple	$40/0.2$	1	**27**	top. aw.	$40/0.2$	1
4	single	$100/0.5$	1	**16**	multiple	$100/0.5$	1	**28**	top. aw	$100/0.5$	1
5	single	$140/0.7$	1	**17**	multiple	$140/0.7$	1	**29**	top. aw.	$140/0.7$	1
6	single	$180/0.9$	1	**18**	multiple	$180/0.9$	1	**30**	top. aw.	$180/0.9$	1
7	single	$10/0.05$	2	**19**	multiple	$10/0.05$	2	**31**	top. aw.	$10/0.05$	2
8	single	$20/0.1$	2	**20**	multiple	$20/0.1$	2	**32**	top. aw.	$20/0.1$	2
9	single	$40/0.2$	2	**21**	multiple	$40/0.2$	2	**33**	top. aw.	$40/0.2$	2
10	single	$100/0.5$	2	**22**	multiple	$100/0.5$	2	**34**	top. aw.	$100/0.5$	2
11	single	$140/0.7$	2	**23**	multiple	$140/0.7$	2	**35**	top. aw.	$140/0.7$	2
12	single	$180/0.9$	2	**24**	multiple	$180/0.9$	2	**36**	top. aw.	$180/0.9$	2

TABLE I

THE SIMULATED SCHEMES 1-36. THE NUMBER OF NODE N=200. THE COLUMN 1/2 STANDS FOR TYPE OF MCAN ALGORITHM: MCAN1 IS THE ORIGINAL, MCAN2 IS THE MODIFED. EACH SCHEME 1-36 HAS BEEN SIMULATED IN BOTH SCENARIO A AND B. (SOURCE)

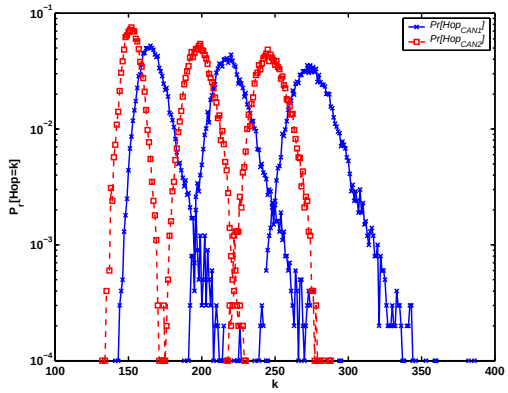

Fig. 5. The pdfs of MCAN1 and MCAN2 in topology aware CAN overlay (schemes 28-30 and 34-36), $N = 200$, $\rho = 0.5, 0.7, 0.9$, scenario a

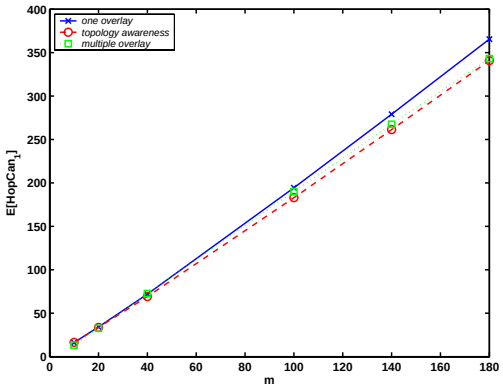

Fig. 6. Effect of overlay creation on hopcount of MCAN1 ($N = 200$, scenario b)

2) Effect of the topology awareness on MCAN1 and MCAN2: The average hopcount in MCAN1 and MCAN2 as a function of m, in a single, multiple and topology aware CAN (schemes 1-36), has been plotted in Figure 6 for scenario b.

Remarkably, in scenario b the topology awareness does not seem to impact the hopcount significantly. Moreover, for a small ratio ρ ($\rho \leq 0.2$), the hopcount of MCAN1 obtained in a topology aware CAN (schemes 25-30) is higher than the hopcount in multiple overlays (schemes 13-18). A possible explanation is that when a new user joins the group, it chooses the nearest node in CAN as the split node, which consequently may separate two nodes already neighboring each other in CAN. This phenomenon is illustrated in Figure 7. We consider a small portion of CAN overlay, where two users A and B, close to each other in the underlying network, are neighbors in the CAN space. Node C lies in another part of the underlying network. If a newcomer C is close to A on the underlying substrate, it chooses A as the nearest node already in CAN. A splits its zone and assigns half of it to C. However, this partitioning can be performed in such a way that the addition of C in the existing CAN structure will separate A and B,

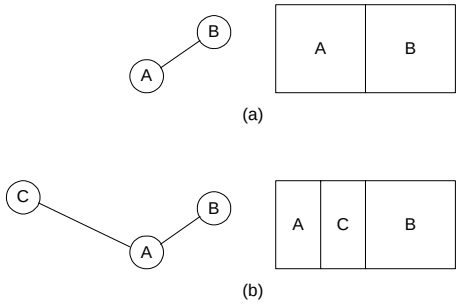

Fig. 7. Joining of the node C increases the distance between A and B.

resulting in a higher hopcount among them (as exemplified in Figure 7).

3) The total and effective hopcount of MCAN1 and MCAN2: The average total and effective hopcount of MCAN1 and MCAN2 in scenario b has been displayed in respetively Figure 8 for multiple CAN overlays and Figure 9 for topology aware CAN overlays (schemes 13-36). The effective hops are the hops a message traverse until it reaches each of the destinations for the first time, and do not include the forward-

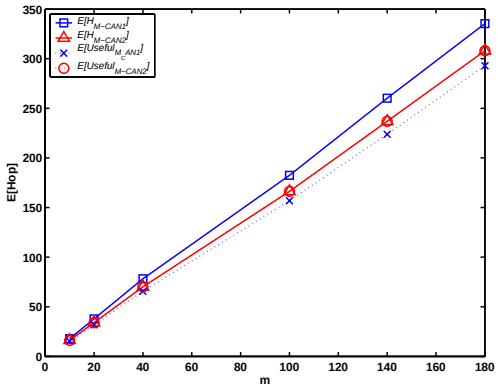

Fig. 8. Comparison of total and effective hopcount in MCAN1 and MCAN2 for multiple CAN overlays (scheme 13-24)

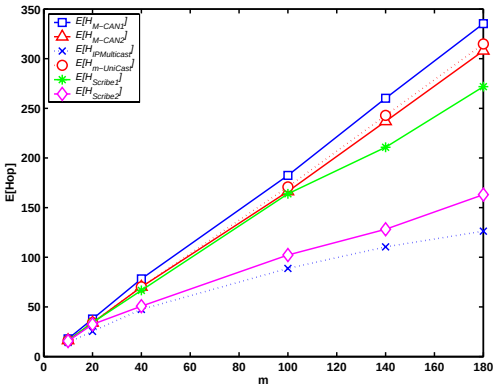

Fig. 10. Comparison of MCAN in multiple-overlay (schemes 13-24) with other mechanisms ($N = 200$, scenario a)

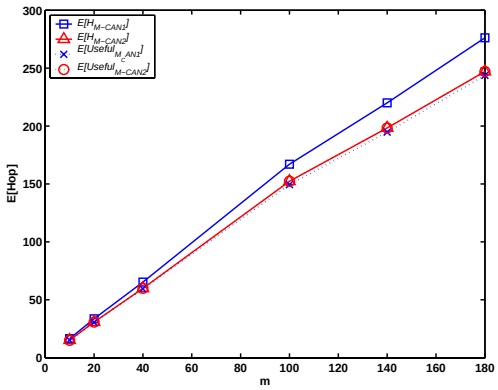

Fig. 9. Comparison of total and effective hopcount of MCAN1and MCAN2 in topology aware overlays (scheme 25-36)

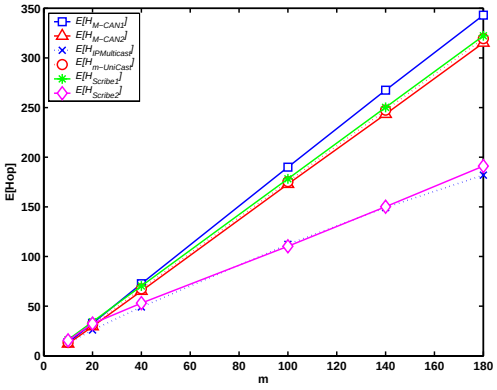

Fig. 11. Comparison of MCAN in multiple-overlay (schemes 13-24) with other mechanisms ($N = 200$, scenario b)

ing paths of the duplicate messages. For both multiple and topology-aware CAN (schemes 18-24, and 30-36), the ratio of the total and the effective hopcount of MCAN2 is around 1, implying that the modifications we introduce eliminate most of the duplicate messages. In a topology aware CAN (schemes 25-36), the number of effective hops is the same for both algorithms, indicating that the total hopcount of MCAN2 is approximately the same as the effective hopcount of MCAN1. In multiple overlays (schemes 13-24), the effective hopcount of MCAN1 is up to 5% lower than the effective hopcount of MCAN2, suggesting, as expected, that the message forwarded with MCAN1 will reach all users in a smaller number of hops. However, MCAN2 optimizes the total number of hops traversed by the message.

B. Scribe on a random graph

For the simulations of Scribe, exactly the same random graphs as those used in the analysis of MCAN have been generated. In order to simulate Scribe, we first define a number of Pastry nodes N_{Pastry}.Random *nodeIds* in the range $\left[0, 2^{128} - 1\right]$ are assigned to those nodes. Since we only simulated a small number of users ($N_{Pastry} \ll 2^{128}$), we confined ourselves to $b = 2$. The leaf set of each Pastry node consists of 8 entries. Again two different scenarios have been simulated: scenario a, in which several Pastry nodes may belong to the same router, and scenario b, where each Pastry node is attached to a different router. Subsequently, a subset of m Pastry nodes has been chosen, representing Scribe participants. Also, a root for the group tree has been determined. A Scribe tree is built by combining Pastry paths from each Scribe member to the root.

Two different sets of simulations: Scribe1, the Scribe algorithm without topology awareness and Scribe2, in which the absolute topology awareness has been integrated.

1) Comparative analysis on a random graph: In Figure 10 and Figure 11 the average hopcount in Scribe1, Scribe2, m-Unicast and IP Multicast, together with MCAN1 and MCAN2 in multiple CANs (schemes 13-24) has been plotted for both scenarios a and b, respectively. The results correspond to $N_{Pastry} = m$. As anticipated, IP Multicast achieves the best performance. These Figure further reveals that the lowest number of hops is reached with Scribe2, and the highest one with MCAN1. Furthermore, the hopcount of MCAN1 in both scenarios is higher than that of m-Unicast. The hopcount of MCAN2 is in both scenarios comparable to the hopcount of m-Unicast. In scenario b, Scribe1 performs slightly worse than MCAN2 but comparable to m-Unicast. In scenario a, for $\rho \geqslant 0.5$, the hopcount of Scribe 1 decreases compared to MCAN2 and m-Unicast. These Figures indicate a great

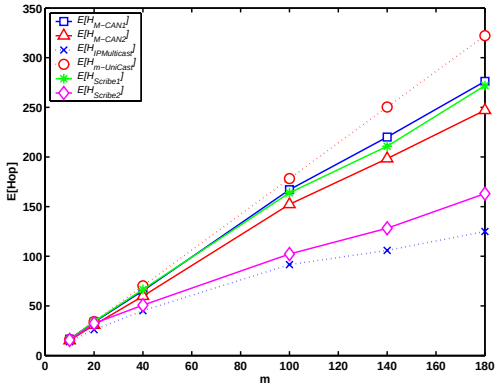

Fig. 12. Comparison of MCAN in topology aware-overlay (schemes 25-36) with other mechanisms ($N = 200$, scenario a)

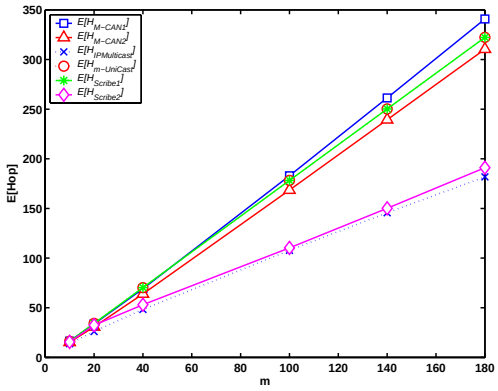

Fig. 13. Comparison of MCAN in topology aware-overlay (schemes 25-36) with other mechanisms ($N = 200$, scenario b)

influence of topology awareness on the hopcount in Scribe. The difference in average hopcount in Scribe1 and Scribe2 raises dramatically with ρ. For $\rho \geq 0.1$ the average hopcount is up to 40% lower than that of Scribe2.

Figure 12 and Figure 13 display the average hopcount in Scribe1, Scribe2, m-Unicast, IP Multicast, together with MCAN1 and MCAN2 in a topology-aware CAN (schemes 25-36) in both scenarios a and b, respectively. These Figures seem to indicate that in scenario a the hopcount achieved with MCAN1 is lower than that of m-Unicast, that performs the worst. In both scenarios, MCAN2 performance is better than that of Scribe1. In scenario b, the results are the same as for the multiple CANs, MCAN1 obtaining the highest hopcount of all, with a slightly improved performance of MCAN1 and MCAN2.

V. Evaluation via Measurements on PlanetLab

In addition to simulations, we evaluated the performance of AL multicast protocols under realistic conditions, by performing experiments on PlanetLab. Our experiments have been executed on November 10th 2004. At that moment, there were 445 PlanetLab nodes running on locations in USA, Asia and Europe. We have performed two sets of measurements.

1) For the measurements corresponding to the scenario b, we selected one node per PlanetLab site, resulting in totally 79 nodes. Each of these nodes represents a multicast group member.

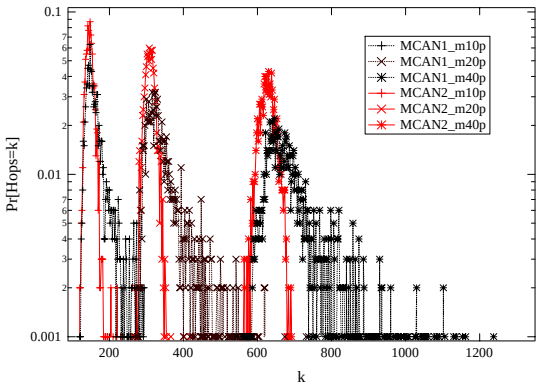

Fig. 14. The pdfs of MCAN1 and MCAN2 in multiple CAN overlay derived from experiments on PlanetLab

2) For the 79 selected nodes Scribe and MCAN without topology awareness have been implemented. One node has been designated as a source. Among the neighbors on the overlay, traceroutes [10] were collected. The hopcounts of MCAN, Scribe, multicast and m-unicast have been computed.

3) Based on the traceroutes collected among these 79 nodes, the underlying router-level topology has been created. This topology consisted of 4226 nodes and 7171 links. No alias resolution technique has been implemented.

4) With the knowledge of this topology, topology aware MCAN and Scribe have been created and implemented, and subsequently, hopcount has been computed.

Figure 14 and Figure 15 show the results of our PlanetLab experiments. Figure 14 gives the pdf of the hopcount in multiple MCAN1 and MCAN2. Also this Figure reveals the clustering effect in the hopcount of MCAN1 as we observed for simulations of multiple CAN. In Figure 15 we have plotted the average values of hopcount for each of the schemes. This Figure resembles the corresponding Figure 11 and 13 obtained from simulations. Again, IP multicast achieves the lowest hopcount. The values of hopcount of MCAN, Scribe1 (without topology awareness), seem all to be approximately equal to unicast. The hopcount of Scribe 2 (with absolute topology awareness) is only slightly lower than topology aware MCAN2, but lower than unicast. Strikingly: the hopcount of topology aware MCAN1 is even larger than that of unicast! One possible explanation is illustrated in Figure 16. The sequence of Figures 16(a) to 16(e) shows the process of nodes joining CAN network. First, only node 1 is in CAN, and then, one by one, other nodes join. Let us assume that node 2 is the nearest node to the newcomer 5. Node 5 then chooses node 2 as the split node. However, even though they will be neighbors in CAN, due to the way MCAN algorithm operates (as described in Section II), they do not send messages to each other. Hence, even though the nodes will be neighbors in CAN, the hopcount among them will not be diminished. In addition, as illustrated earlier in Figure 7 as well, the addition of node 6 (close to, e.g., node 3) may separate nodes 3 and 4, which are nearby in the underlying topology.

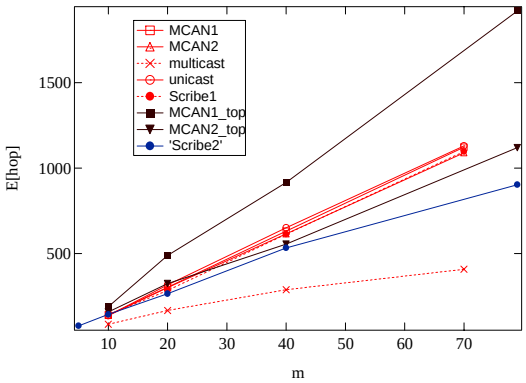

Fig. 15. The average hopcount of all schemes of interest, derived from experiments on PlanetLab

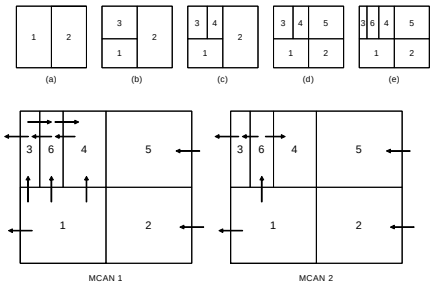

Fig. 16. The node 5, although neighbor of 2, does not receive messages from him..

VI. Conclusions

Our goal was to examine the performance of several scalable Application Layer multicast mechanisms, MCAN and Scribe, under the same conditions, with the hopcount as a performance metric. In addition, we aimed at establishing the upper and lower boundaries for the hopcount of these two algorithms, as a function of the network awareness. Here, we summarize our observations:

1) We observe a hopcount "clustering" effect in MCAN.
2) The number of duplicate messages in MCAN is fairly eliminated with the modifications to forwarding algorithm we introduced.
3) There is no significant influence of topology (un)awareness on the hopcount of MCAN in scenario b (all members attached to different routers).
4) For a small number of multicast users compared to N ($\rho = m/N \leq 0.1$), the hopcount obtained in a topology aware overlay is slightly **higher** than the hopcount in topology unaware overlays, in scenario b.The same phenomenon observed in the measurements results. One possible explanation is the way topological information has been integrated in CAN.
5) In Scribe, we observe a large influence of topology awareness on the hopcount. The difference in hopcount can reach 40%.
6) When the underlying topology is completely known, and each user is attached to a different router, Scribe achieves the same number of hops as IP multicast. The hopcount of a complete topology aware Scribe when the users may connect to the same router remains higher than that of IP multicast. The performance of Scribe without topology awareness is comparable to m-Unicast, if each user is attached to a different router. It does achieve lower hopcount than m-Unicast if users may be attached to the same router. MCAN seems to outperform m-Unicast only if CAN is completely topology-aware, and users may attach to the same router multiple times. Under all the other circumstances, the number of hops of MCAN1 is higher than m-Unicast. The modified CAN algorithm outperforms m-Unicast. It outperforms the topology unaware Scribe as well, unless MCAN2 is applied on a topology unaware CAN and users may join the same router several times.
7) The results of experiments on PlanetLab match our simulation results.

We can conclude that all the schemes perform poorly when underlying topology is unknown. However, in case of CAN, the topology awareness can even lead to an increase in the hopcount.

References

[1] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. "Scalable application layer multicast", Proceedings of ACM SIGCOMM 2002 Pittsburg, PA, August 2002.
[2] B. Bollobas, *Random Graphs*, Academic Press, 1985.
[3] M. Castro, M. Jones, A-M Kermarrec, A. Rowstron, M. Theimer, H. Wang, A. Wolman, "An Evaluation of Scalable Application-level Multicast Using Peer-To-Peer overlays", Proceedings of IEEE INFOCOM 2003, San Francisco, CA, April 2003.
[4] S. Deering, "Multicast routing in a datagram internetwork", PhD thesis, Stanford University, Palo Alto, CA, December 1991.
[5] P. Druschel, M. Castro, A.-M. Kermarrec, and A. Rowstron. "Scribe: A large-scale and decentralized application-level multicast infrastructure", IEEE Journal on Selected Areas in Communications (JSAC), 20(8), October 2002.
[6] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network", Proceedings of ACM SIGCOMM 2001, San Diego, CA, August 2001.
[7] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Application-level multicast using content-addressable networks," Proceedings of the Third International Workshop on Networked Group Communication, November 2001.
[8] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Topologically-aware overlay construction and server selection," in Proceedings of IEEE INFOCOM, 2002.
[9] A. Rowstron, P. Druschel, "Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems", 18th IFIP/ACM Conference on Distributed Systems Platforms, Heidelberg (D), November 2001.
[10] W. Richard Stevens, *TCP/IP Illustrated*, Addison-Wesley, 1994.
[11] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. "Chord: A scalable peer-to-peer lookup service for Internet applications", Proceedings of ACM SIGCOMM 2001, San Diego, CA, August 2001.
[12] P. Van Mieghem, G. Hooghiemstra, R. van der Hoftstad, "On the Efficiency of Multicast", IEEE Transactions on Networking, 9(6), 719 - 732 , December 2001.
[13] B.Y. Zhao, J. D. Kubiatowicz and A. D. Joseph, "Tapestry: An infrastructure for fault tolerant wide-area location and routing". Tech. Rep. UCB/CSD-01-1141, UC Berkeley, April 2001.
[14] S. Q. Zhuang, B. Y. Zhao and A. D. Joseph, "Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination", In 11th ACM/IEEE NOSSDAV, New York, June 2001.