Hopcount in the NICE Application Layer Multicast Protocol

Hong Tang[1], Milena Janic[2] and Xiaoming Zhou[3]
[1]Faculty of Information Science and Technology
Beijing Institute of Technology
Beijing, P.R. China
Email: thtyy2004@yahoo.com
[2]TNO Information and Communication Technologies,
P.O. Box 5050, 2600 GB Delft, The Netherlands
Email: M.Janic@telecom.tno.nl
[3]Faculty of Electrical Engineering, Mathematics, and Computer Science,
Delft University of Technology,
P.O. Box 5031, 2600 GA Delft, The Netherlands
Email: X.Zhou@ewi.tudelft.nl

Abstract - Application Layer (AL) multicast protocols emerged, and in large number, as a response to the slow deployment of IP multicast. However, the gain of different AL multicast mechanisms, in terms of resources consumption over unicast, is questionable. In this paper, we investigate the efficiency of AL multicast protocol NICE, in terms of the number of hops a packet traverses on its path toward the destinations. We propose four alternative algorithms for the creation of NICE overlay and compare their efficiency to the efficiency of unicast, IP multicast, and another two previously evaluated AL multicast protocols (MCAN and Scribe). Further, we investigate the influence of the underlying topology awareness on the efficiency. Our analysis has been performed via simulations, as well as via measurements on the PlanetLab network. We show that the NICE structure achieves better results than MCAN and Scribe.

Keywords— **Application Layer multicast, NICE, overlay networks, Hopcount**

I. Introduction

A myriad of multimedia applications that involve multiple simultaneous users has emerged in the last decade (e.g. video streaming, distance learning, and many others). Most of them are considerably bandwidth-demanding, hence the efficient delivery is indispensable. The simplest way to achieve the delivery of group applications is to use unicast. Unicast represents a point-to-point communication between a single sender and a single receiver. When using unicast to realize group communication, a source of multimedia has to send a copy of the packet carrying that content as many times as there are users, wasting the network resources.

The best efficient way of distributing same multimedia to multiple users is through network layer (IP) multicasting [5]. In IP multicast only one copy of packet is sent out, up to the point in the network at which the paths to the destinations split, where it is replicated and sent out to links leading to those destinations. Packet replication and routing are handled by network routers, which need to support multicast routing.

Due to the lack of a widely available IP multicast service and the boom of peer-to-peer (P2P) applications, a possibility of implementing multicast services in the application layer (AL) has been examined [14] [2] [7] [9] [4]. In AL multicast, data packets are replicated at end users. End users self-organize into a logical overlay network (e.g. CAN [8], Chord [12], Pastry [10] and Tapestry [13]), and transfer data along the edges of the overlay network using unicast. The goal of AL multicast protocols is thus to construct and maintain an efficient overlay for data transmission.

AL multicast has several attractive features. The major advantage of AL multicast is that the network infrastructure may remain unchanged, since it uses unicast for data transmission. However, as data is replicated and forwarded by end hosts (users), packets may traverse the same link several times, resulting in an inefficient use of bandwidth. Hence, AL multicast can only make sense if it outperforms unicast in terms of resources utilization.

In the last several years, a vast number of AL multicast protocols have emerged. They all claim to be more efficient than unicast, but due to their scarce and limited evaluation drawing strong conclusions is not justified. This paper examines the efficiency of NICE [1][2]. NICE (recursive acronym of Nice is the Internet Cooperative Environment) is a structured P2P multicast mechanism in which nodes self-organize into a clustered and layered topology. Due to its hierarchical structure, NICE is considered one of the most promising protocols in terms of scalability. However, the clustering and layering mechanism has a direct impact on the efficiency.

Our analysis represents the extension of our study of CAN-based multicast and Scribe presented in [6]. The main purpose of this paper is to evaluate to what extent do the different schemes for the creation of overlay structures (*i.e.* topology-aware or topology-unaware) impact the efficiency. To investigate the influence of the creation schemes of the NICE, we compare their performance with other scalable AL multicast algorithms, CAN-based multicast [9] and Pastry-based Scribe [4]. We further compare these schemes to unicast and IP multicast. As a performance metric, both the hopcount (the number of hops between nodes) and the node degree have been evaluated. The evaluation is carried out both via simulations, and experiments on the PlanetLab[1].

To the best of our knowledge, the only study of the NICE protocols to this end has been provided by Banarjee *et al.* [1][2]. However, our study differs in several aspects: First, we present four alternative methods for NICE overlay construction, in which hopcount and node degree are optimized respectively. Further, we perform our simulations on a large number of different underlying topologies (up to 10^5). Finally, we evaluate our observations via experiments on the PlanetLab network, which, to the best of our knowledge, has not previously been done for the NICE protocol.

One of the major criticisms of AL multicast is that the end users do not possess the IP topology information available to routers, causing even more inefficiencies than unicast. In this paper, two extreme situations have been considered. In the topology-unaware overlay network, the nodes are organized and connected in a random fashion, without taking the underlying IP-layer topology into account. In the other extreme, we assume that the underlying substrate is fully known and this information is optimally exploited for the overlay creation.

The paper is organized as follows: Section II provides a brief description of the NICE protocol and the four alternatives for overlay construction we propose. Section III describes the simulation design, and presents the simulation results. It also provides the performance comparison of NICE, MCAN and Scribe. In Section IV the results of the measurements on PlanetLab are presented, and the performance of NICE, MCAN and Scribe are compared. We conclude in Section V.

II. The Original NICE Protocol and the Variants

This section describes the original NICE protocol together with the variants that we propose.

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

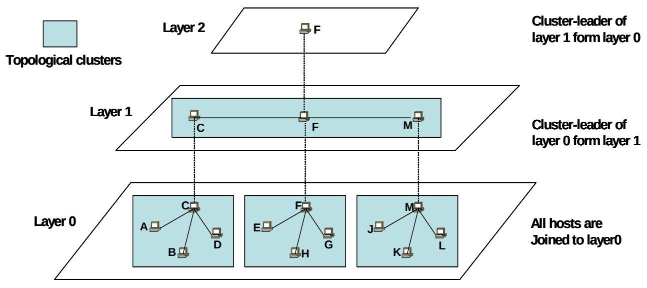

Topological clusters
Layer 2
F
Cluster-leader of layer 1 form layer 0
Layer 1
C
F
M
Cluster-leader of layer 0 form layer 1
Layer 0
A
B
C
D
E
F
G
H
J
K
L
M
All hosts are Joined to layer0

Fig. 1. Hierarchical arrangement of multicast users in NICE (for a multicast group size $m = 12$ and a cluster size $k = 4$)

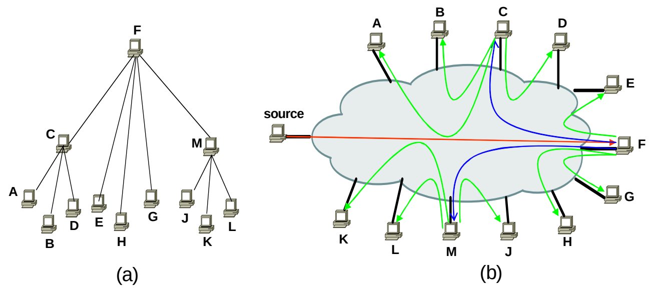


Fig. 2. (a) The data transmission tree corresponding to the NICE overlay given in Figure 1, (b) an exampe of a real-world situation

A. NICE Overlay

As shown in Figure 1, the NICE overlay is created by assigning members to different layers. Layers are numbered sequentially, where the lowest layer of the hierarchy is layer zero (denoted by L_0). All users are part of the lowest layer, L_0. Users in each layer are partitioned into a set of clusters, with the size of each cluster between k and $3k - 1$ (where k is a constant). Further, each cluster has a cluster leader. Each cluster leader of all the clusters in layer L_i represents the cluster in the upper layer L_{i+1}, which is also organized in clusters. An example is shown in Figure 1, where the number of multicast users (multicast group size) is $m = 12$ and the cluster size is $k = 4$. The layer L_0 clusters are $[A, B, C, D]$, $[E, F, G, H]$ and $[J, K, L, M]$. C, F and M are the leaders of their respective clusters in the layer L_0, and they form a cluster in the layer L_1. The leader of the cluster in L_1 is F. Thus, F belongs to layer L_2 as well. Once the overlay is created, the data in NICE is disseminated along the tree, implicitly embedded in this layered structure. The root of the tree is the user in the highest layer. The users in the other layers become the children of their cluster leaders. Figure 2 depicts the data transmission tree.

B. Four Variants of NICE

An end user that wants to join NICE contacts the root to obtain the identity of several users that constitute the highest layer in the hierarchy. The joining user sends probe packets to each of them and selects the "closest" one (in terms of a certain metric). From the selected user the newcomer obtains the list of members of the respective cluster in the first next lower layer. This process continues until the new user finds an appropriate cluster in the lowest layer L_0. It is important to note that, as clusters and layers can be created using different clustering algorithms (based on different metrics), their efficiency can differ significantly. Therefore, we investigate three different topology aware clustering algorithms, denoted as HNICE, DNICE and DHNICE, that we briefly describe in the sequel. In addition, we also evaluate RNICE, a topology unaware algorithm for the overlay construction.

B.1 RNICE Algorithm

In this algorithm, NICE users have no knowledge of the underlying topology and are organized in a random fashion. The clusters and layers are created in such a way to comply to the cluster size boundary k.

B.2 HNICE Algorithm

In the HNICE algorithm, the proximity metric used in creating the clusters is the hopcount between the nodes in underlying network. The node with a minimal hopcount toward the source node and all the other $m-1$ multicast receivers is chosen to be a root (the member of the highest cluster). Other nodes are grouped into clusters from the upper to lower layers in such a way that each cluster leader has the lowest hopcount to all the other cluster members.

B.3 DNICE Algorithm

In this algorithm, the metric used for creating the clusters is the node degree of nodes in the underlying topology. All the nodes are grouped in clusters according to their degrees in a descending order from the upper to the lower layer.

B.4 DHNICE Algorithm

The DHNICE algorithm is a combination of HNICE and DNICE algorithm. The root in the highest layer (L_{max}) has the highest node degree of all, while the other nodes are grouped in clusters according to their hopcount toward their cluster leader in a descending order.

III. Evaluation via Simulations

A. Simulation Setup

In our simulations, we confine ourselves to random graphs of class $G_p(N)$ with N nodes, and independently chosen links with probability p and link weights equal to 1. We first generate a topology consisting of $N \geqslant 100$ nodes, where each node represents a router in the underlying network. For each graph of N nodes, we define the number of multicast users m in the network, such that a ratio $\rho = \frac{m}{N}$ lies in the set $\rho = \{0.025, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.9\}$. For each ($N$, m) pair, 10^5 different topologies have been generated. One of the m randomly chosen nodes is designated as a source node.

In each underlying topology, six different multicast schemes have been implemented: unicast, IP multicast, RNICE, HNICE, DNICE, and DHNICE. In [6], an analysis has been performed on MCAN and Scribe. We use the data and results obtained in [6] for comparing the performance among NICE, MCAN and Scribe. With MCAN we refer to the CAN-based Multicasting. The Content Addressable Network (CAN) [8] is a virtual multi-dimensional Cartesian coordinate space on a multi-torus. A simplest way to achieve multicasting in CAN is to perform flooding over the CAN space, but it is inefficient. Two modifications of the original flooding algorithm have been proposed, one by the Ratnasamy *et al.* in [9] and another by Janic *et al.* in [6], and are referred to in the this paper as MCAN1 and MCAN2, respectively. Scribe [4] is a tree-based AL multicast mechanism built on top of Pastry [10]. In [6], in addition to the evaluation of topology aware and unaware MCAN1 and MCAN2, Scribe1, the Scribe algorithm without topology awareness and Scribe2, in which the topology awareness has fully been integrated, have been evaluated. The evaluation analysis has been performed under identical conditions as the analysis presented in this paper.

For each of the above listed mechanisms and each underlying topology, the number of hops in the data transmission path is computed and stored in a histogram. From each histogram, the probability density function (pdf) of the hopcount (H_N) was deduced, together with the mean $E[H_N]$ and the variance $var[H_N]$. In unicast, the total hopcount is computed as the sum of the hopcounts on each path segment along shortest paths from a source to each of the m destinations individually. For the IP multicast, the messages are disseminated along the shortest path tree [2], since the most of the current IP multicast routing protocols forward packets based on the (reverse) shortest path. For the NICE, MCAN, and Scribe overlay algorithms, the total hopcount is the sum of the underlying hopcounts along the overlay data transmission path using unicast.

In order to facilitate the understanding of the simulation scenario, in Table I we summarize the parameters used in the simulations.

In order to compare the performance of different NICE schemes, we introduce a relative different hopcount

[2] A shortest path tree is the union of the shortest paths between the source and all m destinations.

scheme	$m/\ \rho$	Cluster size k
1	5/0.025	4,8
2	10/0.05	4,8,16
3	20/0.1	4,8,16,32
4	40/0.2	4,8,16,32,64
5	50/0.25	4,8,16,32,64
6	60/0.3	4,8,16,32,64
7	70/0.35	4,8,16,32,64
8	80/0.4	4,8,16,32,64,128
9	100/0.5	4,8,16,32,64,128
10	120/0.6	4,8,16,32,64,128
11	140/0.7	4,8,16,32,64,128
12	180/0.9	4,8,16,32,64,128

TABLE I
THE SIMULATED SCHEMES 1-12. THE NUMBER OF NODES N=200

(RDH), and define it as:

$$RDH = \frac{E[H_{N-Uni}] - E[H_{N-xNICE}]}{E[H_{N-Uni}]}$$

where $E[H_{N-Uni}]$ is the average value of hopcount when using unicast, $E[H_{N-xNICE}]$ is the average value of hopcount of different NICE schemes, where xNice stands for different NICE overlay creation algorithms (i.e. RNICE, HNICE, DNICE, DHNICE). RDH quantifies the hopcount difference between unicast and other NICE overlay algorithms.

A.1 Effect of the Topology Unawareness

Figure 3 displays the pdf of hopcount using topology unaware algorithm RNICE for different number of multicast destinations m ($m = 10, 20, 40, 60, 80, 100, 140, 180$) where the cluster size $k = 4$. The RDH of the RNICE algorithm is between -3.03% and 1.68%. When the same experiment was conducted with unicast, we found that the hopcount in these two schemes had similar values.
Figure 4 shows the average hopcount $E[H_N]$ of unicast, NICE, MCAN1, MCAN2, and Scribe1 in a topology unaware overlay, as a function of the number of multicast receivers m. We observe that MCAN1 has the worst performance of the four overlay algorithms, even worse than unicast. The hopcount in other three algorithms, RNICE (cluster size $k = 8$), MCAN2 and Scribe1, is comparable to that of unicast.

A.2 Effect of the Topology Awareness

In Figure 5 the average hopcount $E[H_N]$ as a function of the number of multicast receivers m (when $k = 8$) has been given. This figure indicates that among the NICE algorithms, HNICE achieves the best performance, DHNICE is slightly behind, whereas DNICE performs worse than DHNICE, but better than unicast. These results suggest that topology-aware NICE is most efficient when hopcount is used as proximity metric.

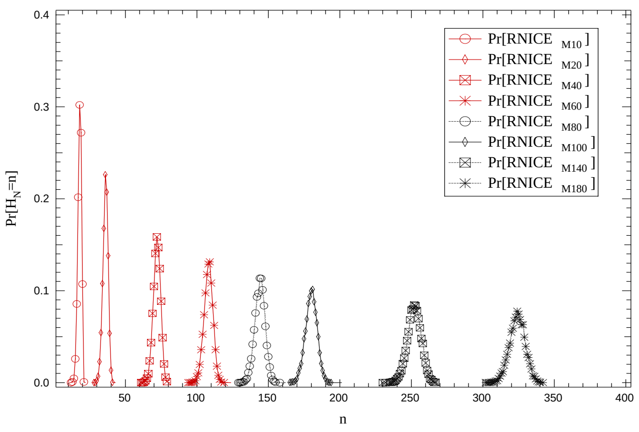


Fig. 3. The pdfs of hopcount in unicast and topology unaware overlay algorithm RNICE. m=10,20,40,60,80,100,140,180 (from left to right).

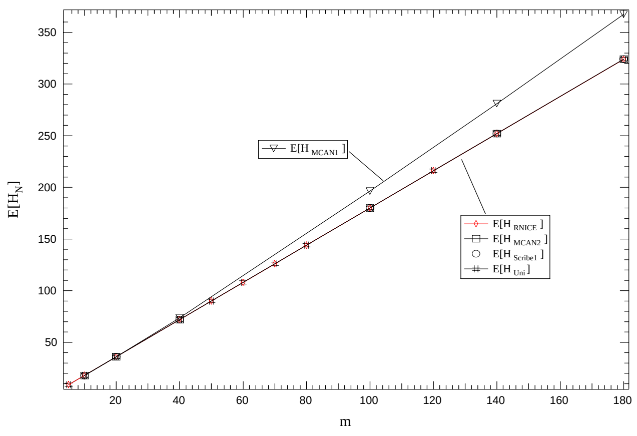


Fig. 4. Performance comparisons of unicast, RNICE (cluster size k=8), MCAN1, MCAN2 and Scribe1 in the topology unaware overlay.

Figure 6 demonstrates the performance comparison of unicast, IP multicast, HNICE, DHNICE, DNICE, RNICE, MCAN1, MCAN2, and Scribe2 in a topology aware overlay, with cluster size $k = 8$. We observe that performance levels can be roughly grouped into two areas: area A consists of MCAN1, unicast, RNICE, DNICE and MCAN2, while area B consists of DHNICE, HNICE, Scribe2 and IP multicast. These results show that IP multicast performs best, as expected, while Scribe2, HNICE and DHNICE produce similar hopcount. Scribe2 performs better than the other AL multicast schemes, and only a little worse than IP Multicast. MCAN2 and DNICE demonstrate poor performance, but better than uni-

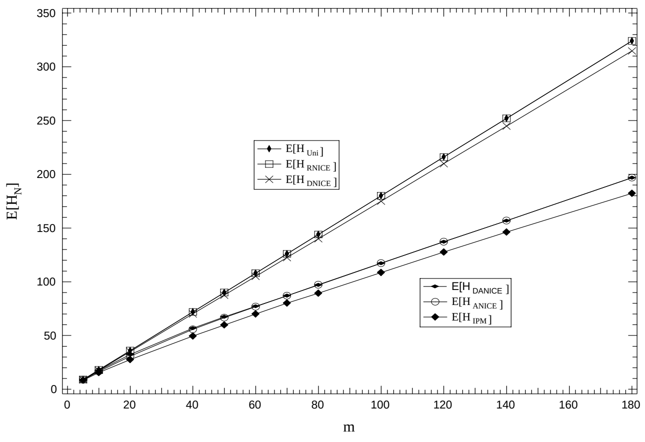


Fig. 5. The average hopcount as a function of number of users m for k=8 in a topology-aware overlay.

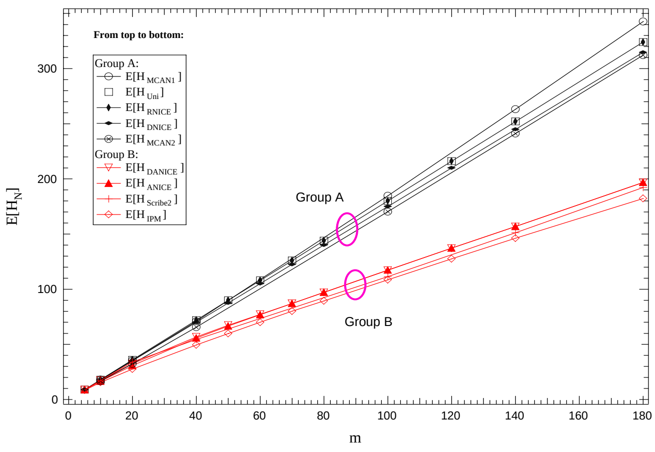


Fig. 6. Performance comparison of unicast, IP multicast, HNICE, DHNICE, DNICE, RNICE, MCAN1, MCAN2, and Scribe2 in the topology aware overlay. NICE cluster size k=8.

cast. MCAN2 achieves a slightly lower hopcount than DNICE. MCAN1 again performs worst, even worse than unicast.

IV. 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 selected one node per PlanetLab site, resulting in totally 79 nodes. Each of these nodes represents a multicast group member or source. Among the 79 nodes, traceroutes [11] were collected, and the hopcount between each pair of nodes has been recorded. Based on this traceroutes collection, the underlying router-level topology has been created. This topology consisted of 4204 nodes and 7041 links. No alias resolution technique has been implemented. We refer to this PlanetLab underlying router-level topology as TPL.

With the knowledge of this topology, NICE, MCAN and Scribe overlays have been created and implemented, and subsequently, their hopcount has been computed. The hopcounts of unicast and IP multicast have been computed as well. In our NICE algorithm measurement, 10^3 different overlays are created for each m (number of multicast destinations), the source node is also chosen randomly, and they all belong to the 79 measurement end-hosts of PlanetLab.

The results of our PlanetLab experiments are given in Figure 7 and Figure 8. Figure 7 shows the trend of $E[H_N]$ as the m increases when cluster size $k = 8$. This figure corresponds to Figure 5 obtained from the simulations. We observe that the results of measurements match the simulation results remarkably well. Again, our results reveal that IP multicast achieves the lowest hopcount. IP multicast is followed by HNICE. DHNICE is slightly poorer than HNICE. Topology unaware algorithm RNICE has the similar performance of unicast. The RDH of the HNICE algorithm is between 14.23% and 36.83%, DHNICE is between 10.38% and 35.40%, DNICE is between 3.77% and 34.3%, RNICE is between -2.78% to 2.59%.

Figure 8 presents the effect of the cluster size k on the average hopcount $E[H_N]$ for $m = 60$. As k increases, the performance of DNICE, HNICE and DHNICE improves. A possible explanation is that TPL topology has a low link density, hence, the hopcount between 2 nodes may be large (in our experiment, the average hopcount between a source and a random multicast destination is about 15 hops). If the size of the cluster is too small, when the cluster i is full, any new coming user, even if the best candidate for joining is the leader of the cluster i, has to join another cluster j ($j \neq i$) in the same layer. This can result in a higher total hopcount. We illustrate this in an example. Figure 9 shows a sparse network topology with 12 routers. To each of them an end user is attached. We create one HNICE overlay for k=6 and k=12 respectively. The total hopcount in HNICE built overlay with $k = 6$ is 27, larger than that the hopcount of 24 for k=12. When k is small, if node C and node E join after the cluster of cluster leader F is full, they will be designated to another cluster leader, e.g. D.

In the extreme situation, when k is larger than m (in Figure 10, k=64), there is only one cluster. DHNICE and DNICE have the same root node and same overlay, so they have the same performance. However, even though there is only one cluster, there are several nodes that might have the same, highest, node degree. Hence, HNICE can choose a node with better performance as its overlay tree root, and consequently, achieve a better performance than DHNICE.

Figure 10 exhibits the average values of hopcount for

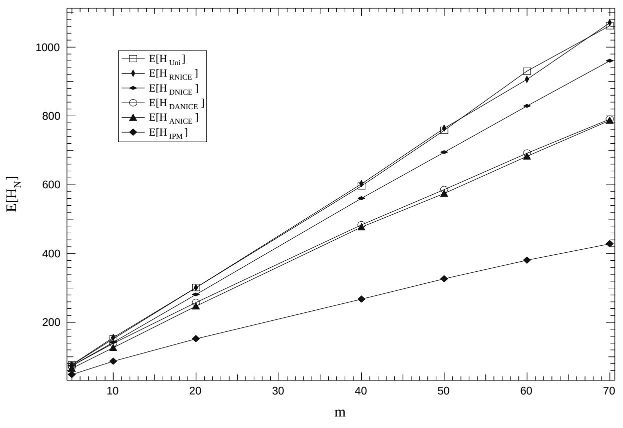


Fig. 7. Average hopcount as a function of number of users m for $k = 8$ (PlanetLab experiments).

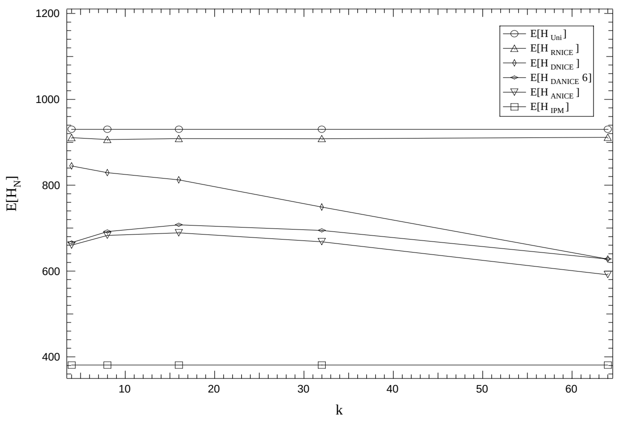
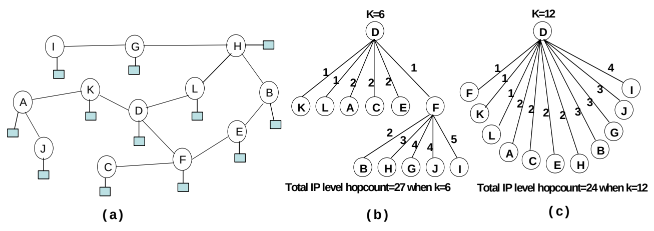


Fig. 8. Average hopcount as a function of the cluster size k for $m = 60$ (PlanetLab experiments).

each of the schemes. This figure corresponds to Figure 6 obtained from simulations. The performance levels of these mechanisms can roughly be classified into three groups: group A consists of MCAN1, MCAN2, Scribe1, Scribe2 and RNICE; group B consists of topology-aware MCAN2 and DNICE; while group C consists of DHNICE and HNICE. Among these three groups, group C performs the best, followed by the group B. The results show that the topology aware algorithms HNICE, DHNICE and DNICE achieve better performance than MCAN and Scribe. The values of hopcount of RNICE, MCAN, Scribe1 (topology unaware), and Scribe 2 (topology aware) demonstrate similar performance as unicast. DNICE performs similarly to topology-aware MCAN2. The results further confirm that there is a significant influence of topology awareness on the hopcount of NICE: topology-unaware algorithms for the creation of NICE overlay structure cause poorest performance of NICE. Clustering algorithm based on hopcount makes NICE perform best; while algorithm based on degree makes NICE performs slightly worse.



K=6
K=12
Total IP level hopcount=27 when k=6
Total IP level hopcount=24 when k=12
(a)
(b)
(c)

Fig. 9. (a) a sparse network topology with 12 routers, each router attaches an end host; (b) and (c) are the overlay data transmission trees of HNICE for the topology (k=6 in (b), k=12 in (c)). The number on the edge is the hopcount between the 2 nodes

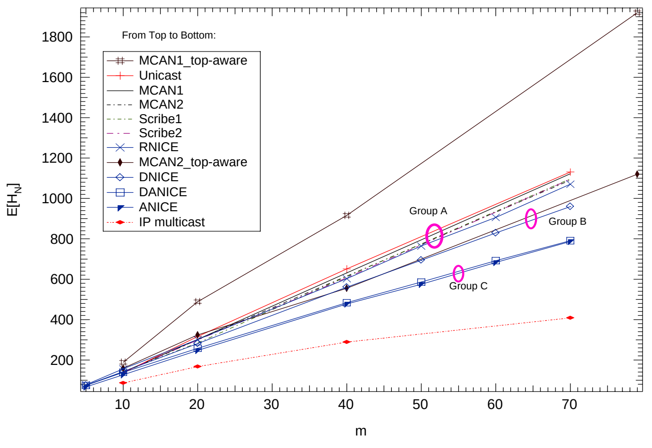


Fig. 10. The average hopcount of all schemes, derived from experiments on PlanetLab

V. Conclusion

We evaluated four different clustering algorithms in NICE, and compared their efficiency to the efficiency of MCAN and Scribe, and IP multicast and unicast, in terms of the resources consumption. Here, we summarize our observations:

• The data transmission efficiency of NICE is between that of unicast and IP multicast. In the topology unaware situation, NICE has the similar performance as unicast. Among the topology aware NICE overlays, HNICE, with hopcount as optimization metric, achieves the minimum hopcount, followed by DHNICE and DNICE.

• There is a significant influence of topology awareness on the hopcount of NICE. All topology-unaware schemes perform similarly as unicast, where MCAN1 is even poorer. In the topology-aware situation, the NICE algorithms perform always better than unicast and MCAN, and better than Scribe in the PlanetLab experiments.

• For the creation of NICE overlay structure, using hopcount as a metric can achieve a better performance than using node degree. Our hopcount-based algorithm HNICE demonstrates better performance than the node degree-based algorithm DNICE. The degree and the hopcount combination algorithm DHNICE performs better than DNICE, however worse than HNICE.

• The effect of the cluster size (k) on the efficiency in NICE has been investigated. The simulation results in PlanetLab suggest that all topology aware NICE schemes obtain lower total hopcount with the increase of k.

Acknowledgement: This work was performed while Hong Tang joined the NAS group at the Delft University of Technology.[3] We would like to thank Professor Piet Van Mieghem for providing facilities for conducting this work. We also thank Novi Ineke Cempaka Wangi for providing relevant data on MCAN and Scribe, and for her feedback on our analysis. Part of this work has been funded by the NWO SAID project.

REFERENCES

[1] Suman Banerjee, Bobby Bhattacharjee, "Analysis of the NICE Application Layer Multicast Protocol" UMIACS TR-2002-60 and CS-TR 4380, Department of Computer Science, University of Maryland, College Park, June 2002

[2] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast", Proceedings of ACM SIGCOMM 2002 Pittsburg, PA, August 2002.

[3] B. BOLLOBAS, Random Graphs, Academic Press, 1985

[4] 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.

[5] S. Deering, "Multicast Routing in Datagram Internetworks", Ph.d., Stanford University, May 1991.

[6] M. Janic, N. Ineke, X. Zhou and P. Van Mieghem, "Hopcount in Application Layer Multicast Schemes", Proc. of 4th IEEE International Symposium on Network Computing and Applications (IEEE NCA05), Cambridge, MA, USA, July 27-29, 2005

[7] Liebeherr J., Nahas M., Weisheng Si, "Application-layer multicasting with Delaunay triangulation overlays", IEEE Journal on Volume 20, Issue 8, Oct. 2002 Page(s):1472 – 1488

[8] 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

[9] 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.

[10] 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, November 2001.

[11] W. RICHARD STEVENS, TCP/IP Illustrated, Addison-Wesley, 1994.

[12] 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.

[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.

[3] At the time this work was conducted Milena Janic has also been affiliated with Delft University of Technology.