Link-Aware NICE Application Level Multicast Protocol

Amr Naser, Mohamed Rehan
Intel Labs
Cairo, Egypt
{amrx.naser, mohamed.m.rehan}@intel.com

Dina Helal, Ayman El Naggar
German University in Cairo
Cairo, Egypt
{dina.helal, ayman.elnaggar}@guc.edu.eg

***Abstract*—Multicast is one of the most efficient ways to distribute data to multiple users. There are different types of Multicast such as IP Multicast, Overlay Multicast and Application Layer Multicast (ALM). In this study, we focus on Application Layer Multicast where the multicast functionality is implemented at the end user. We introduce a new ALM protocol, Link Aware-NICE (LA-NICE), which is an enhanced version of the NICE protocol [1]. NICE is a recursive acronym which stands for NICE is the Internet Cooperative Environment. LA-NICE protocol takes into account the fact that different links can have different bandwidths and this fact can be used to improve multicast message delivery and minimize end-to-end delay. OMNeT++ simulation framework [2] was used to evaluate LA-NICE. The evaluation is done through a comparison between LA-NICE and NICE. The simulation results showed that LA-NICE produces an increase in the percentage of successful message delivery ranging from 2% to 10% compared to NICE. Also, LA-NICE has less average delay and less average message hop count than NICE which reduces the overall latency of message delivery.**

***Keywords*—Application Level Multicast, Multicast tree, Overlay networks , Link Aware, Hop count, NICE, Scribe, OMNeT.**

I. INTRODUCTION

As the number of Internet users increases, data delivery over the Internet becomes more challenging as networks get more overloaded and congested. Currently, data exchange through the Internet is mainly based on unicast (point-to-point between two computers). So, if millions of users try to stream an important broadcast event like the world soccer cup, instead of broadcasting the data to all users, the data source sends a copy of the data to each of the users so the source keeps transmitting the same packet a million times. This leads to redundant traffic in the network in addition to overloading the data source resulting in inefficient data delivery and an increase in packet loss. Multicast was introduced as an alternative to unicast in such cases. In multicast, the source sends contents to a sub-server set and each one of those sub-server set forward the content to a different group of users. There are several types of multicast such as IP, overlay, and application level. In IP Multicast, the multicast process is implemented at the IP level during packet transmission. IP multicast provides an efficient multicast technique. However, it was never widely deployed in the Internet due to multiple reasons including the fact that it requires changes at the infrastructural level which slows down the pace of deployment. Also IP multicasting introduces high complexity and serious scaling constraints at the IP layer in order to maintain a state for each multicast group. As a result of the non acceptance of IP Multicast, the Application layer multicast (ALM) approach was proposed. ALM, also called End-System Multicast, was proposed as an alternative implementation of the multicast technique to the IP Multicast implementation. ALM builds a virtual topology on top of the physical Internet to form an overlay network. Each link in the virtual topology is a unicast link in the physical network [3]. Therefore, the IP layer provides a unicast datagram service, while the overlay network implements all the multicast functionality such as dynamic membership maintenance, packet duplication and multicast routing [4].

NICE [1] is an ALM protocol that arranges the set of members in a multicast group into a hierarchical control topology. As new members join and existing members leave the group, the basic operation of the protocol is to create and maintain the multicast tree hierarchy. The NICE hierarchy is created by assigning members to different levels (or layers) as illustrated in layer 0, figure 1. Layers are numbered sequentially with the lowest layer of the hierarchy being layer zero (L_0). Members in each layer are partitioned into a set of clusters [5]. Each cluster is of size between k and 3k-1 members, where k is a constant (usually k=3), and consists of a set of members that are close to each other. Further, each cluster has a cluster leader. The protocol chooses the center of the cluster to be its leader, i.e., the cluster leader has the minimum distance to all other members in the cluster. This choice of the cluster leader ensures that a new joining member is quickly able to find its appropriate position in the hierarchy using a very small number of queries to other members. The leaders in level i are the members of level i+1 in the tree, so all the leaders in L_0 belong to L_1 and their leaders belong to L_2 and so on until there is only 1 leader which is the Rendezvous Point (RP) in the highest level of the tree. Since each cluster in the hierarchy has between k and 3k - 1 members, a host that belongs only to L_0 layer peers with O(k) other hosts for exchange of control messages. In general, a host that belongs to layer L_i and no other higher layer, peers with O(k) other hosts in each of the layers $L_0....L_i$, which results in control overhead O(k*i) for this member. Hence, the cluster-leader of the highest layer cluster peers with a total of O(k*logN) neighbors, which is the worst case control overhead at a member. NICE mainly focuses on minimizing end-to-end delay. This is done by computing the distance between the nodes and constructing the tree such that nodes close to each other get assigned to the same cluster. This technique minimizes end-to-end delays as the cluster leader is always centered in the middle of the cluster where the distance between it and the rest of the cluster members is minimum.

 ISBN: 978-1-61208-316-2

The rest of the paper is organized as follows. Section 2 describes Link-Aware NICE protocol. Section 3 presents the simulation design, the implementation and the evaluation and test results. Finally Section 4 presents the conclusion based on the simulation results.

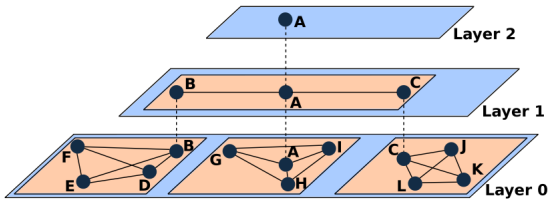

Fig. 1: Hierarchical arrangement of hosts in NICE [6]

II. LINK-AWARE NICE

LA-NICE takes into account the fact that different links can have different bandwidths and uses this fact to improve multicast message delivery and minimize end-to-end delay. As shown in figure 2, LA-NICE doesn't change the cluster structure or how the cluster splits or merges, it focuses on two phases in the tree management which are the member join and the tree maintenance phases.

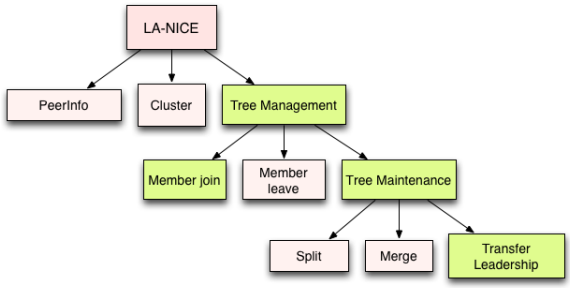

Fig. 2: LA-NICE code structure

A. Multicast Tree Member Join Procedure

When a new node wants to join the multicast tree, in the original NICE algorithm, it contacts the RP with its join query, The RP responds with the hosts that are present in the current highest level of the tree L_i. The joining host then contacts all members in L_i to locate the member closest to it. This member then informs the joining host of its other members in L_{i-1} layer and so on recursively until the joining host finds its position in the lowest level of the tree. Meanwhile, the joining host gets peered temporarily with the RP as soon as it starts communicating with it to get the muticast messages sent until it finds its appropriate position in the tree. This ensures that the joining node gets connected to the tree as soon as it requests to join. Right before joining the tree in the lowest level and after knowing where exactly it will join, the member node requests to be disconnected from the RP and requests to join its appropriate parent in the tree.

LA-NICE modifies the member join procedure of NICE. NICE members join the closest clusters based on round trip time (RTT) measurements. The RTT of sending a message is the time it takes for the message to be sent plus the time it takes for an acknowledgment of that message to be received. The RP gets a list of the clusters leaders ordered by distance and assigns the new host to the closest cluster to it. In LA-NICE, in a tree of i levels, the RP gets a set of potential clusters (PC) that the new node can join using (1) where PC is a set of clusters of size i. Ln is the leader of cluster n. Then the RP checks the bandwidth of the leaders of the potential clusters and assigns the new node to the cluster with the highest ratio of leader bandwidth/number of cluster members as shown in (2).

$$PC = min_{RTT}(New\ node, Ln) \quad (1)$$

$$Selected\ cluster = max_{bandwidth/cluster\ size}(PC\ leaders) \quad (2)$$

B. Multicast Group Member Leave Procedure

When nodes leave the tree, they can leave either gracefully or ungracefully. A graceful leave, which is when a host leaves the multicast group after notifying all the clusters it has previously joined that it's leaving. On the other hand, an ungraceful leave occurs when member fails without being able to send out a leave notification to its clusters, the cluster members then manage to detect this departure when they don't receive HeartBeat message from that member. A HeartBeat message is a periodic message sent by every member to the rest of the multicast group informing them that it still exists in the group. If a cluster leader left the group, this could lead to a partition in the tree so a new leader needs to be chosen faster. A new leader of the cluster is chosen depending on who is estimated to be closest to the center among these members.

C. Multicast Tree Maintenance

A cluster leader periodically checks the size of its cluster, and appropriately splits or merges the cluster when it detects a size bound violation. If a cluster exceeds the cluster size upper bound 3k - 1, it gets split into two equal-sized clusters. Given a set of hosts and the distances between them, the cluster split operation partitions them into subsets that meet the size bounds, such that the maximum radius of the new set of clusters is minimized. The centers of the two partitions are chosen to be the leaders of the new clusters and transfers leadership to the new leaders. If these new clusters still violate the size upper bound, they are split by the new leaders using identical operations.

To maintain the tree structure and detect any unexpected partitioning in the tree, each member of a cluster sends periodic HeartBeat message to each of its cluster members. The message contains the distance estimate from the sender to each other member of the cluster. The cluster leader includes the complete updated cluster membership in its HeartBeat messages to all other members. This allows existing members to set up appropriate peer relationships with new cluster members on the control path. The cluster leaders also periodically send their higher layer cluster membership their cluster.

 ISBN: 978-1-61208-316-2

LA-NICE takes the link load into account when maintaining the tree. So instead of only checking if the cluster leaders need to be modified based on the leader's position with respect to the cluster members, LA-NICE checks the bandwidth of the closest 3 nodes to the center of the cluster (given that the cluster has more than 3 members) and assigns the one with the highest ratio of bandwidth/number of cluster members to be the cluster leader. This modification is selecting the cluster leader based on the fact that there is always higher load on the cluster leaders than the other nodes in the cluster since the cluster leaders send the multicast messages to all the cluster members leading to a bottleneck of O(k log N). So, ensuring that the leader has a relatively high bandwidth in addition to being close to all the members reduces the delay and improves multicast message deliver

III. Results and Analysis

A. Experiment Setup

To evaluate LA-NICE, we compared it to NICE protocol and Scribe[7] which is another ALM protocol. The evaluation was done using four different test groups of users. Each group had different number of users. The number of users in those groups were 20, 45, 70, and 100 respectively. The simulation was done using OMNeT++ as shown in figure 3 and it runs for 300 seconds where nodes exchange multicast messages where some users would drop out while other join the group randomly.

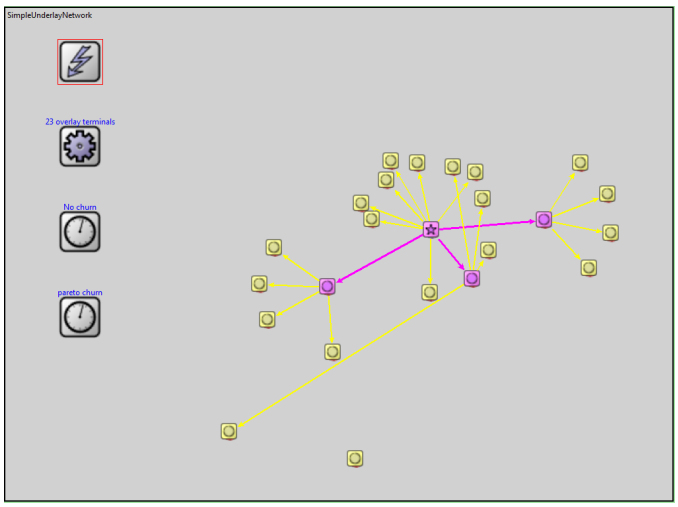

Fig. 3: LA-NICE multicast tree simulation using OMNeT++

B. OMNeT++

OMNeT++ [8] is an object-oriented open-source network simulator that is highly flexible and easy to use. The model's structure is described in OMNeT++ NED language. The Network Description (NED) language facilitates the modular description of a network, which consists of a number of component descriptions (channels, simple/compound module types, gates, etc.). In addition, OMNeT++ provides various statistic collections and visualization tools for results analysis. The simulator supports parallel and distributed simulation with the multiple instances communicating via Message Passing Interface(MPI), as well as support for network emulation through interfaces with real networks and the ability to use real networking code inside the simulator. OMNeT++ simulation models are composed of modules and connections. Connections may have associated channel objects. Channel objects encapsulate channel behavior: propagation and transmission time modeling, error modeling, and possibly others. Channels are also programmable in C++ by the user. Modules and channels are called components. Components are represented with the C++ class cComponent.

C. Implementation

The implementation was done using OMNeT++ framework with oversim [9]. Oversim includes a basic implementation of NICE protocol. They were both used in evaluating LA-NICE performance. The implementation of LA-NICE was built using OMNeT++ oversim framework. In LA-NICE, both the member join and the maintenance methods were modified. The code is implemented in C++ to write the logic of the protocol and OMNeT's .Net language to represent the user interface of the network. Datarate channels were used with different datarates to test various network conditions and bandwidth variations.

D. Results

Message Delivery: The first evaluation criteria for LA-NICE is the percentage of multicast messages delivered, failure in delivery indicates inefficient tree maintenance due to not detecting node departures in a reasonable amount of time or bad bandwidth utilization, which results in bottlenecks that lead to late delivery and sometimes failure in delivery. As seen in table I and figure 4, scribe has lower message delivery percentage. The reason behind the less performance in Scribe is due to the fact that the nodes are assigned random generated IDs. This randomness could lead to a situation where two hosts can be close to each other and yet sending a message from one of them to the other passes by other nodes that are far from them which takes longer time than it should. This is due to the routing table which sends messages to hosts that have the same prefix in their ID (which doesn't always mean that this is the closest node in the table). In addition, messages have a certain time to live (TTL) and then they timeout, so as the delay increases the amount of messages that timeout increase. NICE, on the other hand doesn't have this problem and it sends the messages to the closest nodes graphically without any regard to bandwidth conditions. LA-NICE combines both distance and bandwidth factors.

TABLE I: MESSAGE DELIVERY PERCENTAGE RESULTS

Number of users	NICE	Scribe	LA-NICE		
				% Improvement	
				over NICE	over Scribe
20	90.93	88.57	99.34	8.4%	10.7%
45	96.01	89.65	98.69	2.7%	9.04%
70	90.48	88.71	96.53	6.05%	7.8%
100	93.76	89.62	95.68	1.92%	6.06%

 ISBN: 978-1-61208-316-2

On the other hand, NICE takes the node's proximity to each other into account when constructing the delivery tree. for example, if two nodes are close to each other, the time taken to send exchange messages between them should be less than the time taken to exchange message between nodes that are further away from each other. Our main focus is to build on NICE and enhance its performance by making it link-aware. Link-Aware NICE handles the proximity factor in NICE in addition to the bandwidth factor when a new node joins the tree and when maintaining the tree as well. This is done through measuring the cluster leaders' bandwidth and selecting the cluster with the highest leader bandwidth that is relatively close. This approach produced the highest message delivery percentage as seen in figure 4 compared to NICE and Scribe.

Message Delay and Hop count: Another evaluation criteria is the average hop count of the multicast messages. The hop count of a packet is defined as the number of routers traversed by a packet between its source and destination [10], which is in this case the number of hosts that a message passes from the source to the destination [11]. As seen in figures 5 and 6 and tables II and III when the number of users is small, the average hop count of LA-NICE and NICE, as well as the maximum hop count is almost the same, however as the number of users increase LA-NICE has less number of hops leading to less delay in delivering the messages.

The delay factor is related to the hop count. Figures 7 and 8 and tables IV and V show a comparison between LA-NICE, NICE in terms of delay. The delay and message hops are usually proportional, therefore, with the increase of the hops along the tree, the delay increases. It is clear that LA-NICE outperforms NICE in terms of delay and hop count after taking the links' load factor into account.

TABLE II: AVERAGE HOP COUNT RESULTS

Number of Users	LA-NICE	NICE
20	1.55	1.48
45	1.73	1.77
70	1.79	2.1
100	1.78	2

TABLE III: MAXIMUM HOP COUNT RESULTS

Number of Users	LA-NICE	NICE
20	2	2
45	4	4
70	4	7
100	3	6

TABLE IV: AVERAGE DELAY RESULTS (S)

Number of Users	LA-NICE	NICE
20	0.923	0.91
45	0.83	0.9
70	1.54	1.58
100	1.21	1.19

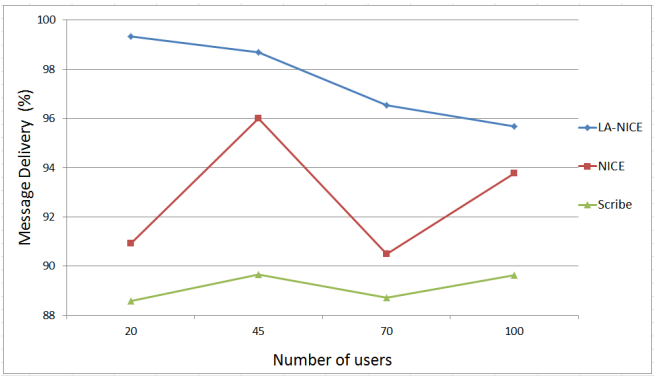

Fig. 4: Percentage of Message Delivery of LA-NICE, NICE and Scribe against different number of users

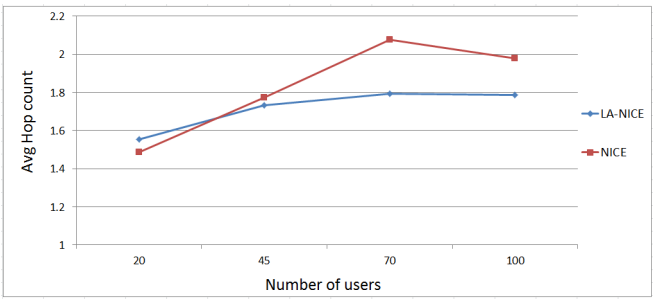

Fig. 5: Average hop count in LA-NICE and NICE against different number of users

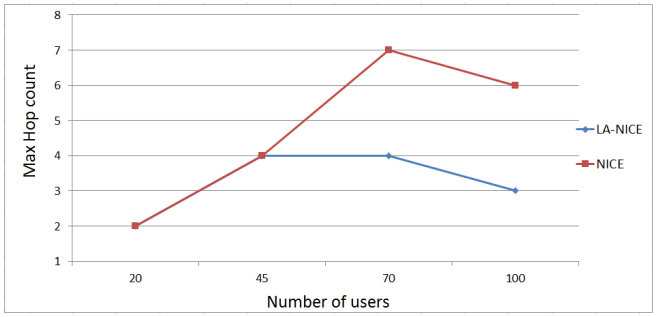

Fig. 6: Maximum hop count in LA-NICE and NICE against different number of users

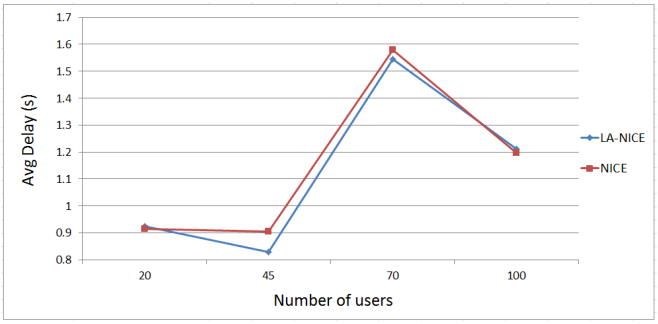

Fig. 7: Average Delay in LA-NICE and NICE against different number of users

TABLE V: MAXIMUM DELAY RESULTS (S)

Number of Users	LA-NICE	NICE
20	1.67	1.72
45	1.76	1.57
70	2.48	2.48
100	2.01	2.05

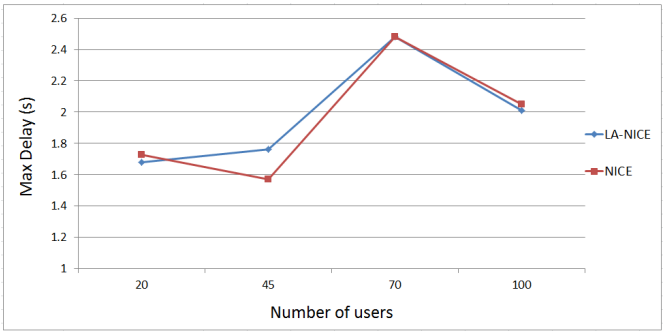

Fig. 8: Maximum Delay in LA-NICE and NICE against different number of users

IV. CONCLUSION

We presented LA-NICE which is an enhancement version of the ALM based NICE protocol. The added enhancement resulted in improved multicast message delivery and lower end-to-end delay. LA-NICE takes into account the fact that different links can have different bandwidths. The original member join, member leave and the maintenance functions in NICE algorithm were modified to include the link information. OMNeT++ simulator was used to evaluate LA-NICE and compare it against NICE as well as Scribe protocols. Simulation results showed that LA-NICE produced higher percentage of successful message delivery and less delays in data forwarding multicast messages compared to NICE and Scribe protocols. The study of the behavior of the three algorithms has shown that as the number of users increased and the network became more congested, the successful message delivery decreased and the delay increased.

REFERENCES

[1] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," vol. 32, no. 4. New York, NY, USA: ACM, Aug. 2002, pp. 205–217.

[2] A. Varga and R. Hornig, "An overview of the omnet++ simulation environment," in *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, ser. Simutools '08. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, pp. 60:1–60:10.

[3] K. Ke and C. Huang, "Performance evaluation of multisource application layer multicast (alm): Theoretical and simulative aspects," vol. 57, no. 6. New York, NY, USA: Elsevier North-Holland, Inc., Apr. 2013, pp. 1408–1424.

[4] L. Lao, J.-H. Cui, M. Gerla, and D. Maggiorini, "A comparative study of multicast protocols: top, bottom, or in the middle?" in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 4, 2005, pp. 2809–2814 vol. 4.

[5] X. Li, X. Zhang, W. Luo, and B. Yan, "A clustering scheme in application layer multicast." vol. 30, no. 2, 2011, pp. 335–355.

[6] W. Xijuan, J. Ruisheng, L. Guang, and Y. Xianghong, "Research on p2p-based application layer multicast technology for streaming media," vol. 1. Los Alamitos, CA, USA: IEEE Computer Society, 2010, pp. 341–345.

[7] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron, "Scribe: A large-scale and decentralized application-level multicast infrastructure," vol. 20, 2002, pp. 1489 – 1499.

[8] A. Varga, "Omnet++ discrete event simulation system user manual," 2011.

[9] I. Baumgart, B. Heep, and S. Krause, "Oversim: A scalable and flexible overlay framework for simulation and real network applications," in *Peer-to-Peer Computing, 2009. P2P '09. IEEE Ninth International Conference on*, 2009, pp. 87–88.

[10] C. Hübsch, C. P. Mayer, and O. P. Waldhorst, "User-perceived performance of the nice application layer multicast protocol in large and highly dynamic groups," in *Proceedings of the 15th international GI/ITG conference on Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance*. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 62–77.

[11] S. Rizvi, M. Khan, and A. Riasat, "Deterministic formulization of bandwidth efficiency for multicast systems," in *Computer, Control and Communication, 2009. IC4 2009. 2nd International Conference on*, 2009, pp. 1–6.

 ISBN: 978-1-61208-316-2