A Scalable Application-Level Multicast Approach based on Mobile Agents

C. Ragusa, A. Liotta, G. Pavlou
Centre for Communication System Research
University of Surrey
Guildford
Surrey, UK
{c.ragusa, a.liotta, g.pavlou}@eim.surrey.ac.uk

Abstract-- **IP multicast is used to deliver services to groups of users. However, this approach needs an infrastructure support in place, which makes the service not always available. We propose a novel application-level multicast approach based on mobile agents, which does not require any specific layer 3 support and suits the requirements of dynamic networking environments. We present a simulation-based comparative analysis between our approach and DVMRP, focussing on performance and scalability.**

I. INTRODUCTION

The simultaneous provision of services to many different users is nowadays realized through the multicast solution. This approach performs data group distribution at network layer adding more capabilities to the routers. In this way multicast is able to reduce packet replication to the minimum necessary. Although various multicast protocols have been proposed over the last decade, this solution performs well for LAN networks but does not achieve the same performance for inter-domain networks. In [1] a number of problems are identified and discussed. Existing network-level multicast protocols incur significant overheads in the routers; they require specific hardware support in place; they cause sensitive issues behind protocol standardisation; and, finally, they are not suitable for extremely dynamic environments such as the mobile one. These reasons have induced some researchers to find new solutions working above the network level, which are commonly referred to as application-level multicast or overlay multicast. Unlike network-layer multicast, in application-level multicast data is replicated at the end hosts rather than at routers. Those end-systems form, therefore, an overlay network that is used to deliver data packets to end users. The purpose of application-level multicast is, hence, to build and maintain this overlay network.

In this article we present an application-level multicast solution in which end-systems are realised as Mobile Agents (MAs)[1] that are in charge of creating the overlay network and of maintaining it as network conditions evolve. In this context, the task of constructing the overlay network is equivalent to that of optimally placing the end-systems within the network. We divide this task in tree different phases: 1) finding the optimal number of end-systems (*p*); 2) partitioning the multicast group in *p* sub-groups; 3) near-optimally placing those end-systems within their respective partition. The latter phase involves MA migration both at start-up time (creation of overlay network) and during data distribution (maintenance of overlay network). To demonstrate and assess our MA application-level multicast approach, we carried out a simulation-based comparison with the Distance Vector Multicast Routing Protocol (DVMRP)[10], which is used as a representative network-level solution. We focussed on two performance metrics: 1) The DVRMP multicast tree build-up time versus the overlay tree build-up time, and 2) the stress ratio[2] of the overall network. We have prototyped and evaluated this MA-based application level multicast on top of the JavaSim network simulator, a component-based, compositional simulation environment [12]. While we rely on JavaSim for network and protocol behaviour, we have extended it with MA capabilities in order to prototype our system. In this way, we could assess performance, scalability, correctness, validity, robustness, and stability of the system under consideration. The remainder of this paper describes the related work (Section 2), illustrates our approach (Section 3), presents the evaluation methodology and simulations set up used (Section 4), discusses the simulation results (Section 5), and finally draws our conclusion, giving also an indication of future work (Section 6).

II. RELATED WORK

Application-level multicast is aimed at supporting end-systems that are interconnected through networks which do not support multicasting at layer 3 (i.e. the network layer). They create overlay networks which employ the end systems for data forwarding. In contrast, conventional multicast protocols deliver data streams from source to destination through a store-duplicate-and-forward mechanism performed at the routers.

Various application level multicast approaches have been proposed in the literature. Following [2] it is possible to identify three main areas in terms of overlay construction for data distribution:

[1] MAs are software entities that act on behalf of some other software entity, exhibit some degree of autonomy and are particularly featured with migration capability – i.e. they can roam the network and execute in those nodes that can host them. Other properties of a MA include re-activeness, pro-activeness, adaptability and cloning capability. In particular, cloning is the ability of an agent to create and dispatch copies, or 'clones', of itself.

[2] Stress is one of the metrics employed to compare the performance of an application level multicast solution with the traditional one. It is defined as the number of identical packet that a physical link carries. The optimal value for native multicast is 1.

1. *Mesh-first* approaches construct overlay mesh[3] networks and employ DVMRP-like routing algorithms to deliver the data using mesh information.
2. *Tree-first* approaches configure directly a data distribution tree when a new end-system joins the multicast group and the existing end-system leaves the group.
3. Different protocols such as virtual addressing approaches or other topology construction protocols belong to *implicit approaches*. Virtual addressing approaches assign a virtual address to an end-system using a specific mapping function such as a hash function and flood data according to the virtual address mechanism.

Mesh-First Approaches utilize the mesh topology to create the overlay network needed to deliver the data. The receivers create a mesh or a Complete Virtual Graph (CVG). Then, the sender sends the data to the receivers building a source-spanning tree from the mesh. Finally, each user gets from the mesh group the information needed to find the best path to reach its group members. Thus, since the receivers have access to system information this solution can create optimal multicast routing paths, reducing the delay latency with source specific trees. This allows supporting of many-to-many multicast for real-time application such as audio/video conferencing applications with small-sized group.

Narada, for example, is an application level multicast protocol that employs the CVG or mesh [3]. In this way it is very robust because it relies on mesh or CVG overlay networks but in order to keep this overlay network up to date all the members have to exchange a large amount of packets losing therefore in scalability. Thus, Narada can target audio/video conferencing applications for small-sized groups.

ALMI uses, instead, a centralised control solution to maintain tree consistency and efficiency [4]. The problem of centralized approaches is that they have a single point of failure, in addition of generally being less scalable than their distributed counterparts. It applies a shared-tree solution to build the overlay network used for data distribution, which may result in relatively long delay latencies. Therefore, ALMI is not scalable because of its centralized and mesh-first nature.

Another solution that belongs to the mesh-first area is Scattercast [5], which implements a combination of the overlay approach (over Internet-wide networks) with a localised IP multicast domains. Scattercast makes use of application-aware agents located at strategic positions within the network to adapt the delivery to the specific needs of individual applications and end clients. Agents are located within hosts, not at the router. Thus, the path used for the data will generally incur longer latencies.

Tree-first approaches allow group members to choose the joining location, which includes the possibility that some members do not have knowledge of the overall tree. Because of that, this approach tends to be more scalable than mesh-first. Problems may, however, arise from the appearance of loops caused by the partial knowledge of tree by group members. Therefore, tree-construction algorithms must take appropriate loop-detection or loop-avoidance measures.

An example of tree-first approach is the Tree Building Control Protocol (TBCP), which builds the tree using a recursive approach where the root of the tree behaves as a rendezvous point and the new member contact this root to join the group [6]. The root will accept the new member considering information on its latency from the root children. Because of its recursive nature, the join process is not very fast. However, this approach has the advantage of being scalable. Other solutions as Overcast [7], TAG [8], and Peercast [9] use recursive approaches which cause high latency or low bandwidth, resulting in not fast join procedure.

Finally, within **Implicit approaches** there are solutions such as NICE [14], designed for large groups and low-bandwidth, real-time application, and CAN [15] which uses a virtual addressing approach. The first one uses a recursive joining procedure, which is relatively slow. The second one employs a multicast flooding of the data that does not give guarantee on the delivery.

All of the above approaches try to address different issues that are usually conflicting and are then applicable only to specific cases. Our MA-based solution can be classified among the tree-first approaches. Each agent builds and maintains its part of the tree relying on routing information which is independently maintained by network routing protocols. Thus, the algorithm is scalable, as confirmed by the analysis of section III.

III. Our approach: Mobile Agent based application level multicast

In this section we describe the structure of our system, including the construction and run-time maintenance of the overlay network by the MA system, as illustrated in Fig. 1. Upon being deployed, starting from the data source (dotted line), MAs act as end-systems delivering the group data to the users (tick-line). We can identify two different phases. The first one consists of agent deployment (those agents collectively form the overlay network). It is through a strategic and efficient location process that we rapidly construct effective overlays. We shall next illustrate a progressive deployment process which is linear with network size as well as being near-optimal in terms of agent location.

Once deployed, the agents relay data among themselves as well as distributing it locally. The second phase consists of a combination of data relay with agent self-regulation, aimed at maintaining the overlay in phase of evolving network conditions. The overlay network can be interpreted as source rooted distribution tree where MAs represent the tree leafs. Within their competition zone, MAs deliver data to the users using the unicast protocol. In this phase of the data delivery our approach pays off in terms of stress and adaptivity, with respect to native multicasting.

[3] A mesh network consists of a topology where multiple paths exist between a pair of group members.

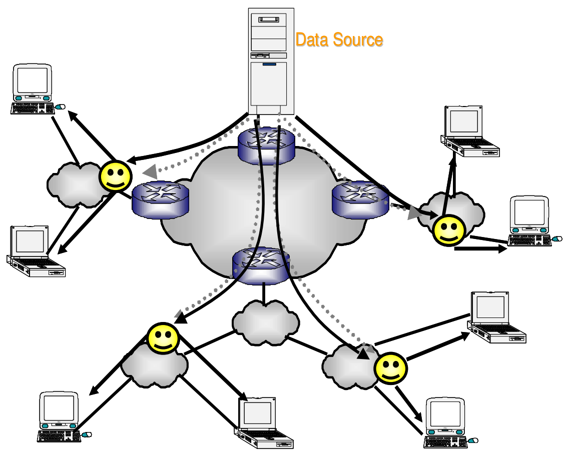


Fig. 1 System structure. MAs act as end-systems to deliver group data from the source to the users.

These two phases are described below in more detail.

A. *Deployment phase*

In this phase we calculate the number and location of agents needed for a given network and set of recipients. Agents of such a system need to have three important abilities: mobility, ability to perform agent cloning, and read-access of the network routing tables. Routing information provides agents the necessary input for the processing of the deployment and self-adaptation processes. Because of that, the efficiency of the MA overlay is sensitive to the accuracy of the routers routing tables which is, in turn, maintained by routing protocols operating independently of the MA system.

Fig. 2 illustrates the MA deployment algorithm. The algorithm deploys the end-systems by partitioning the network through an MA "clone and send" process, which starts at the source host. Initially, the first MA is created at a node that is MA-enabled, namely an MA host. This MA is given the list of group users (*U*) and the operations to be performed on them (data delivery). This first MA builds an estimate of the total cost associated to the current location by summing up the individual routing costs (extracted from the local routing table) related to the involved users' locations. Having estimated the total cost for performing the task from its current location, the MA will then consider alternative configurations (i.e. MA locations) using partial costs attached to the current neighbor nodes.

For each neighbor, partial costs are found by adding the routing costs involved only by those target users that are reached through the specific neighbor node. After that, a simple heuristic function is employed in order to decide whether or not to clone new agents to be sent to the neighbors. For each neighbor node (i), the heuristic function computes the probability (i.e. the advantage) of sending an MA to the node, by using the following formula

$$Pi = \frac{|Ui|}{|Ut|} \times \frac{|Ci|}{|Ct|}$$

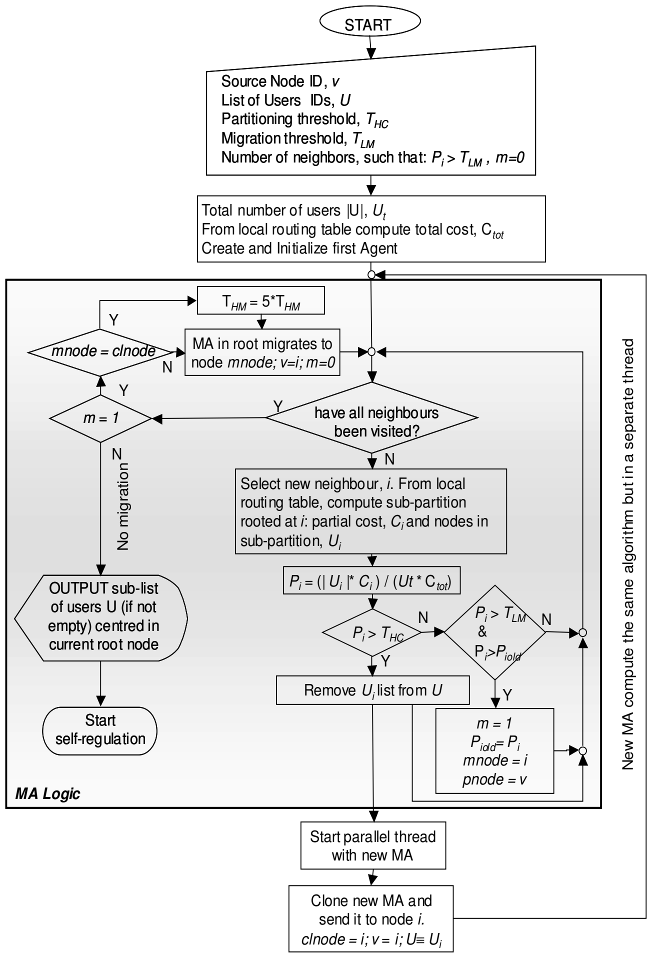

START
Source Node ID, v
List of Users IDs, U
Partitioning threshold, T_{HC}
Migration threshold, T_{LM}
Number of neighbors, such that: $P_i > T_{LM}$, $m=0$
Total number of users |U|, U_t
From local routing table compute total cost, C_{tot}
Create and Initialize first Agent
$T_{HM} = 5*T_{HM}$
Y
$mnode = clnode$
N
MA in root migrates to node $mnode$; $v=i$; $m=0$
Y
$m = 1$
Y
have all neighbours been visited?
N
N
No migration
Select new neighbour, i. From local routing table, compute sub-partition rooted at i: partial cost, C_i and nodes in sub-partition, U_i
$P_i = (| U_i |* C_i) / (Ut * C_{tot})$
OUTPUT sub-list of users U (if not empty) centred in current root node
$P_i > T_{HC}$
N
$P_i > T_{LM}$ & $P_i > P_{iold}$
N
Y
Remove U_i list from U
Y
Start self-regulation
$m = 1$
$P_{iold} = P_i$
$mnode = i$
$pnode = v$
MA Logic
Start parallel thread with new MA
Clone new MA and send it to node i. $clnode = i$; $v = i$; $U \equiv U_i$
New MA compute the same algorithm but in a separate thread

Fig. 2 Deployment algorithm.

Where $|U_t|$ is the total number of users targeted by the agent, $|U_i|$ is the number of users targeted through node *i*, C_t is the total cost and C_i the partial cost. The cloning decision is made by comparing this probability with two appropriate cloning threshold values T_{HC} T_{HM} and which have also an impact on the number of MAs that is finally cloned and deployed. When P_i lays between T_{HM} and T_{HC}, the agent triggers migration. Otherwise, agent deployment is finished, multicasting starts and the agent system switches to the self-regulation phase (described in the next section). A migration flag (m) is activated when P_i exceeds T_{HM} but the actual migration is only activated after all the alternative locations have been examined. P_{iold} is a variable that allows the algorithm to remember the value of P_i for which migration has been set. If another neighbor having a higher P_i appears, migration will be set for that neighbor, therefore ensuring that migration to the neighbor with maximum P_i is triggered. The variable *mnode* identifies the MA destination node. This value is finally compared with *pnode*, which denotes a previously visited node (by the MA). If *mnode* and *pnode* are equal there is the risk that the MA oscillates between two locations leading the MA system to become unstable. This is avoided by increasing the T_{HC} threshold according to our preference.

B. Self-Regulation phase

The second phase is to make sure that the agents are following the dynamics of the network. Node failures, link congestion and users leaving the multicast session result in changes in the network conditions. In order for the agent system to be adaptive, it has to periodically access relevant routing tables and reconsider the MA locations. Fig. 3 shows the algorithm employed for self-regulating the system. This follows the same approach adopted during MA deployment, with the exception for cloning which is disabled in order to avoid uncontrolled agent proliferation.

The parameters of the algorithm are the rooting table look up period (Π), the difference between the monitoring costs associated to two successive retrieves (DC_t), and an MA migration threshold (T_{H1}). Each MA autonomously identifies the neighbor associated to the local minimum multicast cost (this related to the MA partition only). As for deployment, migration is controlled through the use of second threshold (T_{H2}) and a flag, which prevents instability (i.e. agent oscillation between two adjacent nodes).

Next, we evaluate the methodology and simulations design used to carry out our results.

IV. Evaluation Methodology

This section illustrated the methodology used to compare our solution with the Distance Vector Multicast Routing Protocol (DVMRP) [10] [11]. DVMRP is a multicast routing protocol for dense topologies, which means that group members are densely distributed across the network. It forwards packets using a multicast tree that is built by applying the Reverse Path Multicast (RPM) algorithm.

Without loosing generality and in order to achieve reasonable simulation times, we assume that all the nodes in the network are group members. The metrics used for the comparison are the Tree Build up Time and the Stress Ratio that are plotted against topology size. The former is defined for the DVRMP as the period of time elapsed between the initiation of the members' joining procedure and the construction of data distribution tree. In the MA system, the tree build time is the time to finish the MA deployment phase.

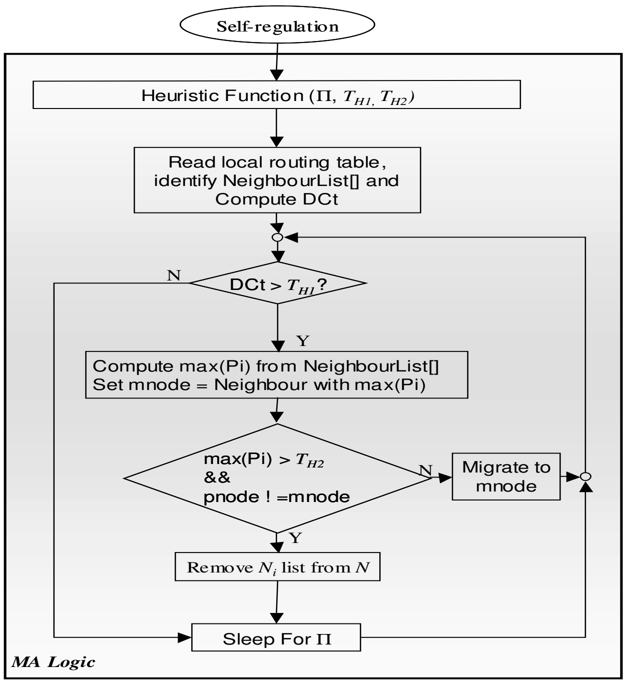
Stress Ratio is defined as:

$$StressRatio = \frac{StressOverlay}{SressMcast}$$

which gives the variation of the overlay network from the optimal value. Since we assume that all nodes are group members, the optimal stress value is 1 packet per link, corresponding to 1 packet per member. For example, a topology of 25 nodes may have a stress of 24 (1 node is the data source) for the DVMRP algorithm, and 33 (calculated as the total number of packets generated) for the MA-based one. In that case, the stress ratio would be 1.32, which means that the overlay network generates on average 1.32 packets per node[4].

[4] Note that the leaf' members receive only 1 packet, but the end-systems can receive more than 1.

In order to carry out this comparison, extensive simulations under random network environments have been performed. It was very important to run the algorithm under realistic network topologies composed of routers, links and hosts. For this reason we used the Javasim network simulator [12], which allows the simulation of a variety of scenarios.



Self-regulation
Heuristic Function (Π, T_{H1}, T_{H2})
Read local routing table, identify NeighbourList[] and Compute DCt
DCt > T_{H1}?
N
Y
Compute max(Pi) from NeighbourList[]
Set mnode = Neighbour with max(Pi)
max(Pi) > T_{H2} && pnode ! =mnode
N
Migrate to mnode
Y
Remove N_i list from N
Sleep For Π
MA Logic

Fig. 3 Self-regulation algorithm.

TCP/IP networking and transport, including the Distance Vector (DV) routing protocol, is used as the bases for our simulations. We have also enhanced the simulator with MAs capabilities, including agent generation, cloning, migration, termination and so on.

The GT-ITM topology generator was used to create realistic, Internet-like, transit-stub topologies [13]. The simulations have been executed for topologies of 25, 50, 75, 100, 150, 200 and 300 nodes. To ensure statistical significance, simulations have been repeated 20 times for each of the above network sizes. That is, we have randomized the simulation process, generating families of 20 topologies at a time, characterized by comparable topological features (e.g. average node degree, number of nodes, etc). In this way we could assess the sensitivity of our metrics to network size, using the number of node as network size parameter and isolating other effects.

In the next section we present and analyze the results of the simulations.

V. Results

Fig. 4 shows the average tree build time for each of the two approaches under evaluation. We can see that the MA-based approach performs slightly worse than DVMRP. This was expected due to the difference in the very nature of the two approaches. Being an application-based solution, our MA approach is bound to incur larger overheads which are countered by a larger flexibility and by its ability to work without layer 3 multicasting support.

An important aspect of both algorithms is linearity with respect to network size. This is an essential feature, which makes the MA approach viable even in the case of large number of nodes/users. Linearity during MA deployment is obtained because that process is carried out in parallel upon each cloning instance. As soon as being injected into the system, each new agent continues the deployment process within its sub-partition (defined by its parent MA), autonomously from any other agent that act in parallel.

Because of that, the overall agent deployment time is proportional to the depth of the agent deployment tree, i.e. to the network size. The factor dominating the linear deployment process is the time needed to migrate between two nodes. We have taken measures in real MA systems and we have found that this factor is in the order of hundred of milliseconds [16].

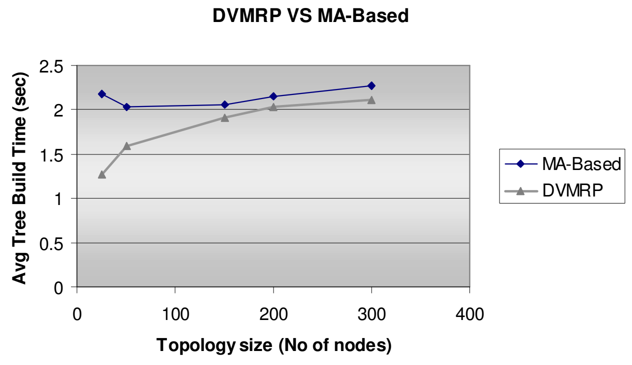


Fig. 4 DVMRP Tree build time Vs MA-based Tree build time

Our findings on the Average Stress Ratio are depicted in Fig. 5. Again we find linearity with respect to number of network nodes/users. In particular, we see that the distance from optimality is relatively small (considering that we have an application-level algorithm compared to a layer-3 solution). The relatively small slope of the MA line (0.049 degrees) demonstrates a negligible sensitivity to network size that can be controlled by changing the number of MAs injected in the system.

When the ratio between number of MAs and number of nodes/users is reduced, the time to deploy those agents (i.e. the time to build the overlay network) tends to decrease because there are less agents to be located. However, in that case the size of the network partitions (i.e. number of nodes/users per MA) increases and, hence, the cost of unicasting within those partitions increases. On the other hand, if the MA to nodes ratio increases, we observe the reverse phenomenon. We can therefore conclude that due to its linearity and order of magnitude of the dominant factors, the MA solution is scalable with respect to network size and number of users and represents a viable alternative to native, layer-3 multicasting protocols.

In this paper we have reported some key problems encountered by the important area of multicast, including layer 3 multicast and other application-level multicast approaches.

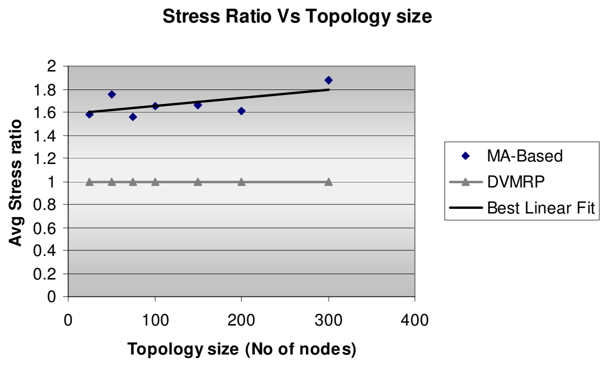


Fig. 5 Stress Ratio Vs Topology size.

VI. CONCLUSIONS & FUTURE WORK

Because there seems to be increased scope for application-level solutions, we have embarked the study of a new approach based on mobile software agents. The idea was that, in order to find adaptable solutions in the context of dynamic and mobile networking environment, the natural way was to use code mobility as a means to address the necessary requirements.

We have therefore come up with an algorithm that would be distributed (in order to pursue scalability), adaptive (in order to keep the pace with changing networking conditions), infrastructure-less (in order to be easily deployed), and efficient (in order to be represent a viable alternative to existing solutions). It is through agent mobility and cloning that we have addressed those requirements.

At the same time, we have gone through the exercise of developing a suitable assessment methodology that would be acceptable to the networking community, providing an insight on the MA-based solution. To achieve that, we started from a widely used network simulator (JavaSim is a Java version of the NS network simulator), enhancing it with the necessary MA capabilities.

Although our initial results show that our approach offers performance that are not too worse than DVMRP, the intention was not to make the case to replace layer-3 solutions with application-level ones. DVMRP has been selected mainly because its implementation was already available in the adopted simulator. Other more efficient layer-3 solutions do exist and it would probably be unwise to pursue the avenue to replace them with better application-layer counterparts. Therefore, we would see network- and application-level multicast as complementary rather than competing solutions, capable of addressing complementary requirements.

Our assessment of the potential of the MA approach is not completed. We shall also seek to improve its performance and reduce instability problems that occasionally arise. But the main focus of the work to be done next is an evaluation of MA-based multicast in the context of mobile networks and mobile users. Having put into place the basic simulation infrastructure we can now look at the behaviour of our system in a number of more complex scenarios, assessing for instance its sensitivity to different routing protocols (our approach relies on routing table information). Finally, our priority will be to refine run-time self-regulation under very dynamic conditions.

ACKNOWLEDGMENT

The work reported in this paper has formed part of the WA1 area of the Core 2 Research Programme of the Virtual Centre of Excellence in Mobile & Personal Communications, Mobile VCE, www.mobilevce.com, whose funding support, including that of EPSRC, is gratefully acknowledged. More detailed technical reports on this research are available to Industrial Members of Mobile VCE.

REFERENCES

[1] A. El-Sayed, V. Roca, L. Mathy, "A survey of Proposals for an Alternative Group Communication Service", IEEE Network magazine, Special issue on "Multicasting: An Enabling Technology", January/February 2003.

[2] Korikang, Kimsh, "Survey on Application Level Multicast", CDS&N (Collaborative Distributed System and Network) laboratory.

[3] Y-h. Chu, S. G. Rao, and H. Zhang, "A Case for End System Multicast," ACM SIGMETRICS 2000, June 2000.

[4] D. Pendarakis, S. Shi, D. Verma and M. Waldvogel, "ALMI: An Application Level Multicast Infrastructure," the 3rd Usenix Symposium on Internet Technologies & Systems (USITS 2001), March 2001.

[5] Y. Chawathe, "Scattercast: An Adaptable Broadcast Distribution Framework," In special issue of the ACM Multimedia Systems Journal on Multimedia Distribution, 2002.

[6] L. Mathy, R. Canonico, and D. Hutchison, "An Overlay Tree Building Control Protocol," NGC 2001, November 2001.

[7] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole Jr., "Overcast: Reliable Multicasting with an Overlay Network," USENIX Symposium on Operating Systems Design and Implementation, October 2000.

[8] M. Kwon and S. Fahmy, "Topology Aware Overlay Networks for Group Communication," Proceeding of NOSSDAV'02, May 2002.

[9] M. Bawa, H. Deshpande, and H. Garcia-Molina, "Transience of Peers & Streaming Media," HotNets-I, October 2002.

[10] IP Multicast and DVMRP, FutureSoft, www.futsoft.com.

[11] Seiji Ueno, Toshihiko Kato, Kenji Suzuki: Analysis of Internet Multicast Traffic Performance Considering Multicast Routing Protocol. ICNP 2000: 95-104.

[12] The JavaSim simulator, www.javasim.org

[13] Source code of GT-ITM, available as http://www.cc.gatech.edu/projects/gtitm/.

[14] Suman Banerjee, Bobby Bhattacharjee, and Christopher Kommareddy, "Scalable Application Layer Multicast," *Proc. of ACM SIGCOMM'02,* August 2002.

[15] Sylvia Ratnasamy, Mark Handley, Richard Karp, Scott Shenker, Application-level Multicast using Content-Addressable Networks, In Proceedings of Third International Workshop on Networked Group Communication (NGC'01).

[16] C. Bohoris, A. Liotta, G. Pavlou, Evaluation of Constrained Mobility for Programmability in Network Management, in Services Management in Intelligent Networks, Proceedings of the 11th IEEE/IFIP International Workshop on Distributed Systems: Operations and Management (DSOM '00), Austin, Texas, USA, December 2000