

Application Layer Multicast For Efficient Peer-to-Peer Applications

Adam Wierzbicki¹

e-mail: adamw@icm.edu.pl

Robert Szczepaniak¹

Marcin Buszka¹

1

*Polish-Japanese Institute of
Information Technology
Chair of Computer Networks
ul. Koszykowa 86, Warsaw, Poland*

Abstract

Application Layer Multicast (ALM) can be useful for many types of Peer-to-Peer (P2P) applications. The paper compares a group of ALM algorithms, that take into account network topology measurements, using Monte-Carlo simulation. A new algorithm, called Fastcast, is introduced. Fastcast allows to control the tradeoff between used traffic and the worst-case length of the application layer path. ALM algorithms for P2P applications can benefit from the possibility of limiting the number of children in the ALM tree, since many nodes could be connected by slow modem connections. The paper studies the impact of such a limitation on ALM algorithm performance.

Keywords:

Peer-to-peer, application layer multicast, overlay network, virtual network.

1. Introduction

Many types of peer-to-peer (P2P) applications can benefit from point-to-multipoint communication. Among the popular file-sharing applications, point-to-multipoint communication could be used for querying a set of nodes for a file. Even in P2P applications that use distributed hash tables, such as Pastry [7], such a functionality is sometimes necessary - for example, when the user wishes to make a keyword query for file contents. In other types of P2P applications, point-to-multipoint communication can be of even more importance. In any P2P application in which one user can affect the state of many other users (such as games, or groupware) it is necessary to communicate the change of state to many receivers.

For these reasons, development of efficient point-to-multipoint communication could facilitate the development of new P2P applications. While IP multicast is not likely to be the answer to the problem of efficient point-to-multipoint communication in the Internet, much research has been devoted to the development of

application layer multicast (ALM). This can be briefly defined as an organization of the participants in point-to-multipoint communications into a delivery tree, in order to increase efficiency. The Rhubarb platform for building P2P applications has been designed to use ALM to provide applications with capacity for point-to-multipoint communications [1].

The performance of ALM algorithms can be measured by two criteria: the amount of traffic required for one message to reach all nodes, and the length of the longest path (in terms of hop-count, for example) that is a measure of the time it takes for a message to reach all nodes. This measure is often modified by dividing the length of the worst-case ALM path between two nodes A and B by the length of the shortest IP routing path between A and B . This value is a more fair measure of performance of the ALM algorithm, and will be used in this paper (referred to as *stretch*).

There exists a tradeoff between the two performance measures, since it is possible to reduce traffic by receiving messages from closer nodes at the expense of increased stretch. This paper introduces a new ALM algorithm that allows to control the tradeoff between total traffic and stretch by the use of a parameter.

2. Application Layer Multicast Algorithms

To simplify the description of ALM algorithms considered in this paper, let $\delta(A, B)$ denote the distance measured in number of hops in the IP network between two nodes A and B . Some ALM algorithms use other measures of distance, but in this paper all considered algorithms will use measures of distance based on hop count (without changing the structure of algorithms proposed in the literature).

The *Yoid* algorithm [5] uses $\min_{B \in V} \delta(A, B)$ as a measure of distance between a node A that wishes to join a group of nodes. The set V is a random subset of the group of nodes; in other words, A will be connected to the

tree as a child of a node P that is the closest node in the random subset V in terms of number of hops. At increasing time intervals, the node A will try to improve its location in the tree by making measurements to another random subset V' . This operation can lead to the creation of loops, which makes it necessary for Yoid to use loop-avoidance procedures (which will not be discussed in the paper). Note that Yoid is more resilient to failures due to its approach that tries to correct tree structure over time, and that Yoid's algorithm is not a purely on-line algorithm: the tree correction procedure can take into account nodes that have joined the tree after A . In this paper, we shall consider a simplified, on-line version of the Yoid algorithm, that chooses the closest node of all nodes currently in the group as a parent for A .

Overcast uses the same measure of distance as Yoid, but has another strategy for reducing the number of measurements. Instead of a random subset, *Overcast* will choose among a chosen subset of all current group members. The subset is chosen by proceeding recursively down the current tree. *Overcast*'s algorithm has a parent candidate PC for the joining node A (at the beginning, the root). In each step, let V be the set of children of PC . If some child $C \in V$ satisfies $\delta(A, C) \leq \delta(A, P)$, then C becomes the new parent candidate, and the algorithm repeats recursively with the children of C . If no child satisfies the condition, then A is connected to the current parent candidate.

Overcast's algorithm can be thought of as an organized local search among the current group members. The number of measurements made by *Overcast* is smaller than the simplified Yoid algorithm, but it cannot be as effective since simplified Yoid chooses among all current group members. Note that the local search method used by *Overcast* does not depend on the measure of distance (as a matter of fact, *Overcast* uses the available bandwidth as a measure of distance). Further on, other measures of distance based on hop count will be proposed, all of which can be used with the *Overcast* algorithm.

Both Yoid and *Overcast* create ALM trees that reduce the traffic for a multicast message, but can have a large stretch: the paths in the ALM tree can be much longer than the shortest paths in the IP routing topology. To change this property, it is necessary to use a measure of distance that takes into account the length of the entire path in the ALM tree from the root. Let B be a node in the current ALM tree, and A be a joining node that considers becoming a child of B . Let the path in the ALM tree from the root R to the node B be composed of the nodes C_1, C_2, \dots, C_k . The length of this path, $\delta^{ALM}(B, A) = \delta(R, C_1) + \delta(C_1, C_2) + \dots + \delta(C_k, B) + \delta(B, A)$ can be used as a measure of distance by an ALM algorithm that intends to minimize stretch. The algorithm that uses this measure and chooses as a parent for A the node P that has the smallest distance among all nodes currently in the topology, will be referred to as *Localcast*.

Intuitively, it seems that *Localcast* should always have a stretch of 1, since it can always choose $P=R$ and cannot choose a node P' that has $\delta^{ALM}(P', A) > \delta^{ALM}(R, A)$. However, if it is possible to find a node P' that satisfies $\delta^{ALM}(P', A) = \delta^{ALM}(R, A)$, then that node can be chosen instead of R . Still, *Localcast* is very limited in its choice of parent nodes.

Instead of using the distance $\delta^{ALM}(B, A)$, an algorithm can consider only a part of that distance, or discount the length of the paths high up in the tree against the length of paths lower in the tree. To do so, an algorithm could use weights a_1, a_2, \dots, a_{k+2} . Using the introduced notation, the new measure of distance can be calculated as $a_{k+2}\delta(R, C_1) + a_{k+1}\delta(C_1, C_2) + \dots + a_2\delta(C_k, B) + a_1\delta(B, A)$.

To discount the length of the path closer to the root against the length of paths closer to A , the weights should be decreasing. One possibility is to use exponentially decreasing weights given by the formula: $a_1=1, a_{k+1}=a_1 q^k, 0 < q < 1$. Let the distance measure that uses the above formula and exponentially decreasing weights with parameter q be denoted by $\delta^{ALM}(B, A, q)$. The algorithm that will select the closest node in the current group using this measure as a parent for A will be called *Fastcast*. Notice that *Fastcast*'s behavior will depend on the parameter q , for example, if $q=1$ then *Fastcast* becomes equivalent to *Localcast*. On the other hand, if $q=0$ then *Fastcast* becomes equivalent to the simplified Yoid.

Note that the *Fastcast* algorithms require the same number of network measurements as Yoid. The distances in the higher parts of the tree can be obtained from the parent candidates, who have measured them when they joined the tree.

3. Comparison of ALM Algorithms

To compare the discussed ALM algorithms, we have used Monte-Carlo simulation. The simulation consisted of the following stages:

1. Generate an example network topology.
2. Choose a group of nodes in the topology as the multicast group and an order of arrival for the nodes.
3. Choose a subset of the nodes that will limit their number of children.
4. Use an ALM algorithm to construct a tree, taking into account constraints on number of children.

The first stage of the simulation use the GT-ITM network topology generator [6]. The transit-stub model was used, together with the locality model for the transit and stub networks, to generate topologies that resemble realistic internetworks. The average size of the topology was 200 nodes. 30 random graphs were used in the simulation.

One drawback of the network topologies generated by GT-ITM was that they create a backbone network with few access nodes (nodes that have only one neighbor in the topology). To be realistic for the considered application of ALM (P2P applications that use computers

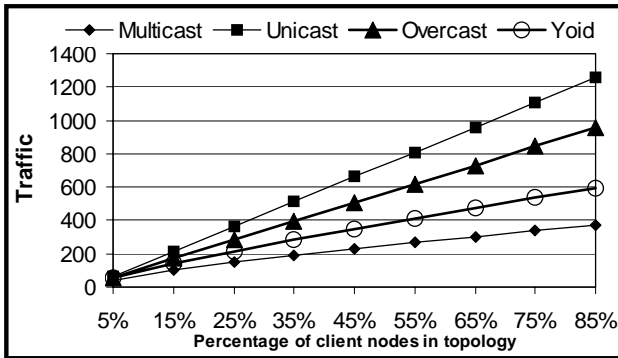


Figure 2 Traffic of Overcast and Yoid

at the edge of the Internet), the graphs generated by GT-ITM were modified to include more access nodes.

The second step was a selection of a group of nodes that were the members of the multicast group. This step depended on the value of a parameter, $k \in \{5\%, 15\%, 25\%, 35\%, 45\%, 55\%, 65\%, 75\%, 85\%\}$, that can be interpreted as the percentage of nodes in the topology that were members of the multicast group. In other words, if the topology had exactly 200 nodes and $k=15\%$, then 30 nodes were chosen as members of the multicast group.

For each topology, 15 random multicast groups were generated, first for $k=5\%$. Next, for each larger value of k , the multicast groups were increased, keeping the nodes chosen in the previous step. The ordering of the nodes in the group was determined by the random order of adding nodes to the group. In this way, for each value of k , there were 15 different groups. However, for the same initial group of $k=5\%$, groups with larger k had the same root.

While the group members were chosen randomly, the choice was constrained to ensure that in all cases 90% of group members were access nodes. This constraint ensured that the group membership reflected the location of P2P users in a network.

The third step was used to investigate the effect of slow modem nodes that could limit their number of children. These nodes were specially chosen so that they were always access nodes. A parameter, $l \in \{10\%, 20\%, 30\%\}$ was used to determine how many modem nodes were present in a group. A second parameter, $c \in \{1, 2\}$ determined how many children each of these nodes agreed to have. In other words, there could be for example 10% modem nodes that all agreed to have at most 1 child in the ALM tree. This step in the simulation was optional and could be skipped.

The final step was to construct a distribution tree using an ALM algorithm. For comparison, a tree was constructed using shortest paths in the underlying topology to simulate IP multicast routing algorithms. The IP multicast (IPM) tree was created by overlapping the shortest paths, and could include nodes that were not members of the multicast group. A flat ALM tree was

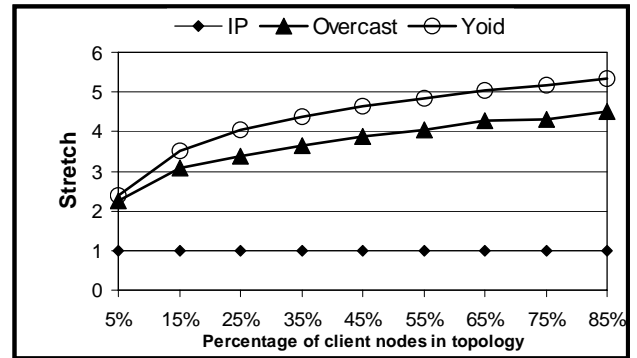


Figure 1 Stretch of Overcast and Yoid

also created to simulate direct distribution from the root using unicast. Each tree was evaluated using the two measures discussed above: traffic needed to deliver one multicast message to the entire group, and stretch. The results for all combinations of random topologies and multicast groups were averaged to eliminate random influence.

The use of Monte-Carlos simulations that consisted of several steps resulted in an exponentially increasing number of computations. For example, if stage 1 generates 30 random graphs, step 2 generates 15 random multicast groups, each group is selected for 9 different values of k , step 3 generates 15 different subsets of nodes that limit their number of children for 3 different values of l and 2 different values of c , and step 4 simulates 8 different ALM algorithms, then together we needed 2916000 separate computations. For that reason, we chose to use average graph sizes of 200 nodes. Another reason for this choice was that the performance of ALM algorithms depends mainly on the density of receiving nodes in the network, and not on topology size. Increasing the average graph size to 500 nodes for a reduced subset of simulations did not affect the results. However, simulations that use very large graphs (order of 10000 nodes) are part of our future work.

4. Results of the comparison

The first comparison of ALM algorithms was made to evaluate the organized local search of Overcast. To do so, Overcast and Yoid were compared to unicast and IP multicast. The results of this comparison are shown on Figure 1 and Figure 2. On Figure 1, the traffic of each distribution method was shown. Notice that Overcast performs worse than Yoid. This indicates that a global search among all current group members for the closest node in terms of hop count is an efficient method of reducing traffic. However, this method can also lead to very long application layer paths, which can be seen on Figure 2 and 4, where the stretch of Yoid is worst of all the considered algorithms (IP stands for both Unicast and Multicast that have stretch equal to 1).

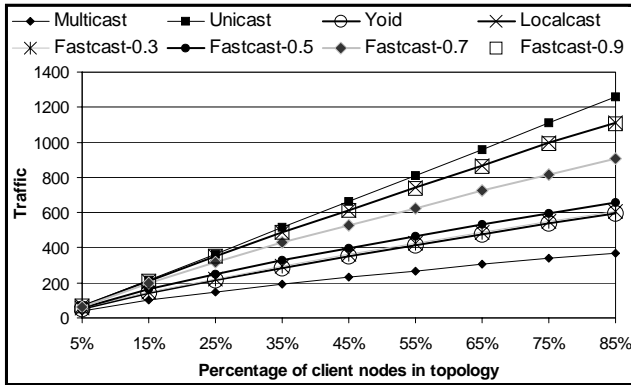


Figure 4 Traffic of Fastcast algorithms

Overcast performs a constrained, organized local search using the same distance measure as Yoid. It turns out that this method leads to a lower stretch. This implies that Yoid is able to find solutions that have lower traffic at expense of higher stretch, but a constraint that does not allow Yoid to reach this solution could lead to lower stretch and higher traffic. Such a constraint is the limitation the number of children of slow modem nodes. We shall return to this subject at the end of this section.

The second part of the comparison deals with the performance of the Fastcast algorithms that allow to control the tradeoff between stretch and traffic by the means of the parameter q . Recall that q controls the speed of the decrease in the exponential weights of Fastcast, and therefore the relative importance of the length of the paths closer to the root as compared to the length of the paths closer to the joining node. For $q=0$, Fastcast performs like Yoid, and for $q=1$, Fastcast performs like the Localcast algorithm that always has stretch equal to 1. Therefore, as q increases, the Fastcast algorithm should have better stretch, but worse traffic.

These observations are confirmed by the results of the comparison shown on Figure 3 and Figure 4. The Figures show the performance of Fastcast for $q \in \{0, 0.3, 0.5, 0.7, 0.9, 1.0\}$. As discussed above, Fastcast-0 is equivalent to Yoid, and Fastcast-1 is equivalent to Localcast. On Figure 3, the curves for Localcast and Fastcast-0.9 overlap. This is due to the high percentage of access nodes in the multicast group (90%). Note that Localcast does not save much traffic when compared to unicast. However, its stretch is always equal to 1, which means that it overlaps with the IP curve on figure 4.

For good stretch and traffic, the choice of Fastcast-0.5 or Fastcast-0.7 seems most appropriate. Note that Fastcast-0.7 has a very low stretch, while providing some traffic savings. Fastcast-0.5 has a significantly larger stretch, but also much larger traffic savings. However, one can argue that usually the percentage of client nodes in a topology will not be very high. For low values of k ,

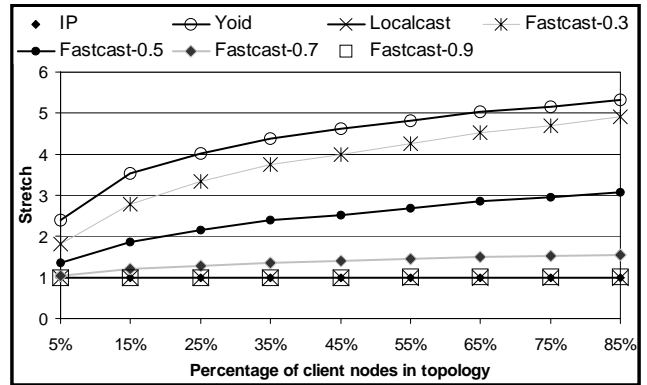


Figure 3 Stretch of Fastcast algorithms

the traffic used by Fastcast-0.7 is much better than direct unicast communication.

If the application would require lower traffic at the expense of higher stretch, it could consider Yoid or Fastcast-0.3 that both have the lowest traffic. Of the two, Fastcast-0.3 has a lower stretch, and is therefore a preferable choice.

A third comparison was made to evaluate the effect of the possibility P2P nodes can limit the number of children in the ALM tree. This possibility was introduced in order to limit the amount of traffic over slow or expensive modem links.

The comparison was made by repeating all simulations in an environment where l percent of group users were connected by modem nodes, and limited their number of children to c . The limitation did not have a significant effect on stretch for any algorithm. The comparison of Yoid and Overcast seemed to suggest that a constrain of Yoid's algorithm could lead to a reduction of stretch at the expense of increased traffic. However, this is not the case for the considered simulations; the only observed effect was an increase of traffic.

To simplify the presentation of the results, the traffic of the algorithms was scaled in such a way that it represents the relative position of the algorithm between the results of IP multicast and unicast. Let the traffic of an ALM algorithm be T . Let, for the same values of parameters l and c , the traffic of IP multicast be M , and the traffic of unicast be U . Then, the scaled traffic of the ALM algorithm is given by the formula: $(T-M)/(U-M)$. If the scaled traffic is close to zero, then the traffic used by the ALM algorithm is close to multicast traffic; if the scaled traffic is close to one, then the ALM algorithm uses about the same amount of traffic as unicast.

The results of the comparison are shown on Figure 5 for the Yoid and Fastcast-0.5 protocol. Other protocols showed a similar behavior. The effect of introducing more nodes that limit their number of children is a deterioration of the traffic used by the ALM distribution without significantly affecting stretch. However, the limitation of

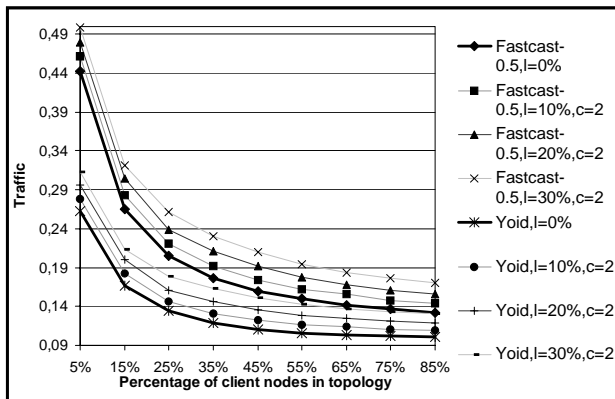


Figure 5 Effect of limiting number of children

the number of children is still a very useful feature, since it allows to control the traffic on slow/expensive access links.

Observe that all of the considered curves are decreasing, indicating that for larger percentages of the multicast group in the topology, ALM algorithms become more efficient. This is natural when one considers that the performance of unicast deteriorates very quickly when the percentage of multicast group members increases. The scaling is affected by unicast performance, and therefore the ALM algorithm performs better when compared to unicast.

Observe that all of the discussed Fastcast algorithms can be used with a form of organized or random local search that limits the number of measurements required by the algorithm. A simple modification of the Fastcast algorithm would be to substitute Fastcast's distance

measure in the Overcast algorithm. Another possibility is to use Yoid's repeated random local search.

5. Conclusion and Future Work

This paper introduces the Fastcast algorithm that belongs to the class of root-based, on-line and topology-aware ALM algorithms. Fastcast is controlled by a parameter, and by changing this parameter it is possible to control the tradeoff between used traffic and stretch of the ALM tree. Fastcast does not increase the number of network measurements; rather, it uses measurements already made by nodes that previously have joined the tree.

The paper has developed a method of comparison of ALM algorithms using Monte-Carlo simulation. This method can be used to study the effect of changing location of the members of a multicast group on ALM algorithm performance. The paper has shown that the increase of the number of multicast group members that have slow modem access and can limit their number of children can increase the total traffic used by ALM algorithms, but does not have a significant effect on stretch.

Among future work, the authors plan to extend the comparison of ALM algorithms, such as Narada and NICE [2], and off-line algorithms that originate from research in the field of ATM networks. Further study of the effects of random and organized local search on various algorithms is also planned.

References

- [1] A. Wierzbicki, R. Strzelecki, D. Świerczewski, M. Znojek, "Rhubarb: a Tool for Developing Scalable and Secure Peer-to-Peer Applications", Second IEEE International Conference on Peer-to-Peer Computing, 2002
- [2] S. Banerjee, B. Bhattacharjee, C. Kommareddy, "Scalable Application Layer Multicast", SIGCOMM'02, 2002
- [3] Z. Dziong, "ATM Networks Resource Management", McGraw-Hill, 1997
- [4] J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, J. O'Toole, "Overcast: Reliable Multicasting with an Overlay Network", Proceedings OSDI'01, 2000
- [5] P. Francis, "Yoid: Your Own Internet Distribution", Technical Report, ACIRI, April 2000, www.aciri.org/yoid
- [6] E. Zegura, K. Calvert, S. Bhattacharjee, "How to model and internetwork", Proc. IEEE Infocom, pp. 40-52, March 1996
- [7] P. Druschel, A. Rowstron, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems", Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware'01), 2001
- [8] I. Stoica, R. Morris, D. Krager, M. F. Kaashoek, H. Balakrishnan, "Chord: a scalable peer-to-peer lookup service for internet applications", Proceedings of ACM SIGCOMM'01 Conference, 2001
- [9] B. Zhao, J. Kubiatowicz, A. Joseph, "Tapestry: An infrastructure for fault-resilient wide-area location and routing", Technical Report CSD-01-1141, U.C.Berkeley, 2001