

Improving the Performance of Overlay Multicast with Dynamic Adaptation

Shuju Wu
School of Information Science
University of Pittsburgh
Pittsburgh, PA 15260
Telephone: (412) 624-6822
Email: swu@mail.sis.pitt.edu

Sujata Banerjee
University of Pittsburgh
and
Hewlett-Packard Laboratories
Email: sujata@hpl.hp.com

Abstract—Overlay multicast protocols should adapt to the dynamic network environment to maintain application performance. In this paper, we show how end-to-end network metrics can be used in the dynamic adaptation and thus improves the performance of the direct-tree overlay multicast protocols while providing good tree quality. We propose the Adaptive Overlay Multicast (AOM) approach that provides good tree quality and efficient adaptation. We compare AOM with a typical direct-tree protocol like HMTP. The simulation results show that AOM builds high-quality trees, adapts better to network conditions and is efficient.

Keywords— Protocols, performance, overlay multicast

I. INTRODUCTION

IP multicast [1], the technique for supporting group communications (e.g., conferences, classes, distributed computing and information dissemination systems) in the Internet, has been studied for more than ten years. However, its deployment is still in a small scale due to various issues. Recently, overlay multicast (also called application layer multicast) has been proposed to provide multicast service at the application layer and remove the dependence on multicast support from underlying networks. Operations such as group membership management and routing are implemented in the application layer, and data distribution depends on a multicast tree that consists exclusively of end hosts and unicast connections.

However, overlay multicast has to deal with two important performance issues. The first is the building and maintenance of an efficient distribution tree without excessive overhead. An overlay multicast tree uses the end hosts to forward the data, thus it has longer end-to-end latency and higher link stress. Link stress is defined as the number of duplicate packets on a single physical link. Also, an end host may fail more frequently than a router and it can leave the group at any time. In both cases, the distribution tree may get partitioned. Therefore, the overlay multicast protocol must monitor the tree status constantly. In order to scale to a large group size, the overhead to build and maintain the tree should be kept low.

The second issue is related to the fact that the Internet is dynamic and unpredictable in nature. Thus over time, the multicast tree structure may become inefficient or even partitioned and adaptation of the tree to the dynamic network environment is necessary to maintain application level performance. The dynamic events that cause the tree quality to degrade, include the changing membership, node or link failures, and network congestion. In this paper, we term any of the above events that affects application performance as a *fault*.

We propose a direct-tree protocol called the Adaptive Overlay Multicast (AOM) protocol, which has the following characteristics: (1) It constructs efficient overlay multicast trees by using both end-to-end and local network metrics and reasonable scoping technique to reduce overhead. Most of the previous works use Round Trip Time (RTT) as the only metric to build the tree, but our study shows that end-to-end network metric such as latency can help build overlays that match the underlying network topology better and thus decrease the penalty of using overlays. (2) It adapts the tree dynamically and efficiently as faults occur and maintains the end-to-end network performance metrics such as delay, loss and throughput. In this paper, we exclusively consider the end-to-end delay (EED) metric though additional metrics can also be used. The adaptation in most previous works (if it exists at all) is passive in that a member periodically looks for a new parent in the tree without the knowledge of the end-to-end performance metrics. (3) AOM is especially beneficial to non-interactive applications requiring large data distribution to large receiver sets. Examples of such applications include Internet audio/video broadcasting (delay and loss sensitive), video replication (loss intolerant) and large software distribution (loss intolerant).

To our knowledge, the only paper that studies the dynamic adaptation is [2], where bandwidth adaptation is shown to be important in the context of a mesh-first protocol, called NARADA. Since mesh-first protocols are only suitable for the small size groups and the direct-tree protocols are believed more scalable, in this paper, we study how to build high quality multicast trees and efficiently adapt the tree under dynamic network conditions to maintain the end-to-end delay metric in direct-tree protocols. Since none of the current direct-tree protocols actively adapts to the Internet faults and considers the end-to-end performance metric, we select HMTP [3], a typical direct-tree protocol, and compare it to our approach. Same as most other direct-tree protocols, HMTP uses RTT as the only metric to construct the tree.

The paper is organized as follows. Section II introduces the related work. Section III describes and compares our adaptation scheme with HMTP in detail. In Section IV, the simulation results are presented, followed by conclusions and future work in Section V.

II. RELATED WORK

Previous overlay multicast studies focus on self-organizing the group members into a delivery tree and the approaches can

be classified as: centralized [4], [5], distributed direct-tree [3], [6], [7], and distributed mesh-first [8], [9]. ALMI [4] takes a centralized approach where a central controller builds the overlay and disseminates the tree information to the group members. NARADA [8] and Gossamer [9] build a mesh first and run a DVMRP-like routing protocol to build the tree. These protocols are generally designed for small groups due to the scalability issue. HMTP [3], NICE [10], Overcast [6], YOID [7] and AOM build the tree directly when members join the group, i.e., the tree is extended when a new joining member finds the most suitable position and connects to an existing member. All these protocols except for Overcast use Round Trip Time (RTT) as the metric to build overlays. Although RTT is a coarse estimation of the delay between two nodes, it eliminates the need for synchronized clocks among group members. Overcast targets to maximize the bandwidth of the members rather than the other metrics. However, totally disregarding the RTT could affect the tree quality and waste network resources. HostCast [11] uses the end-to-end delay metric to build the overlay, however, it requires the synchronized timing across all the members to calculate the end-to-end delay. In addition, the path computation algorithm in HostCast focuses on the path with the shortest end-to-end delay, therefore, no effort is given to group the close members together to make the overlay match the underlying network better.

[2] studies the overlay multicast protocol in the dynamic network environment. Their experiments were carried on a mesh-first protocol, NARADA. The experimental results show that it is important to adapt those metrics that matter to the application during the overlay construction. We believe it is necessary to study dynamic adaptation in the direct-tree protocols because: first, the objective of direct-tree protocols is to be scalable. Therefore, they do not have an explicit multicast routing protocol as in NARADA and Gossamer that can distribute useful information for the adaptation. Second, the routing protocol used in NARADA should not be applied in direct-tree protocols due to the scalability issue. The adaptability in mesh-first protocols depends on the routing update frequency and there is a tradeoff with the overhead.

Resilient Overlay Networks (RON) [12] allows a small group of nodes to form a full-mesh application overlay that can forward data among the RON members in face of the underlying Internet faults. A link-state routing protocol is run on the mesh to exchange the path metric information (latency, loss rate, etc.) of the overlay links and build forwarding tables based on the path metrics. If the underlying Internet path between a source RON node and a destination RON node is not the best one, the source RON node forwards the packet by way of other RON nodes. RON is not designed for multicast applications and the full-mesh architecture limits its scalability to only tens of nodes.

III. AOM - ADAPTIVE OVERLAY MULTICAST

In this section, we present AOM, the adaptive overlay multicast protocol, and compare it to HMTP [3], a typical direct-tree protocol. The objectives of AOM are to (1)

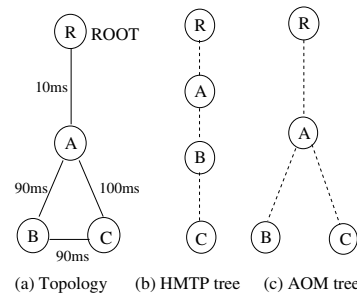


Fig. 1. Example of parent selection metric

maintain a good distribution tree, (2) adapt to the network conditions by making appropriate tree changes to keep good and stable end-to-end delay (EED) and (3) provide efficient adaptation with a low overhead. AOM consists of three modules: a tree building protocol, performance monitoring and fault detection, and adaptation.

A. AOM Tree Protocol

An AOM tree is built with a ROOT at the top level. Basically, it takes a member three steps to join and connect to the tree: join, initial parent searching and periodical improvement.

- 1) Join: new members always join the ROOT first.
- 2) Parent searching: a new member tries to find a parent that is topologically close.
- 3) Tree improvement: each member periodically re-evaluates its current position and switches to a new parent if necessary.

We only emphasize the differences between HMTP and AOM tree protocol that affect the tree quality. The details of the tree protocols, such as membership management, loop avoidance, detection and resolution, can be found in [3] and [13].

1) *Tree building metrics*: Tree building metrics are important factors that affect the tree quality. HMTP uses the RTT between two members as the only metric and a new member moves as far as possible from the ROOT as long as the RTT satisfies some conditions. In the simple network topology in Fig. 1 (a), assume A and B are already in the group, and B is the child of A. Now C joins the group and moves down as A's child, too. In HMTP, C will move down as the child of B if only $RTT_{C,A} > RTT_{C,B}$ and $RTT_{C,A} > RTT_{B,A}$. The final HMTP tree is in Fig. 1 (b) and C has an EED of 190ms, which is much larger than being A's child.

We define a member's end-to-end delay (EED) as the sum of its parent's EED and half of the RTT between them. In reality, delays are not symmetric in both directions. We make this simplifying assumption to remove the requirement of having tightly synchronized clocks between any two members. AOM can always use more accurate estimates of EED if they are available. AOM uses both the EED and the RTT between nodes to construct the tree. The reason is that EED reflects the relative position of a member to the ROOT, but RTT reflects the relative position among neighbors. Before a member decides to switch to a new parent in the tree, it should compare its EED through the new parent to the EED through the current parent. If the new EED is too large, the member

TABLE I
FAULT ADAPTATION ALGORITHM

1.	on detection of faults at member m : probe $S_a = \{ROOT, ancestors\}$ for RTTs and EEDs.
2.	wait for reply, then update performance metrics through ancestor a as: $RTT_{new} = \beta \cdot RTT_{new} + (1 - \beta) \cdot RTT_{old}$; $EED_{m,a,ROOT} = EED_{a,ROOT} + RTT_{new,a}/2$;
3.	add a to potential parent list pl if: $EED_{m,a,ROOT} < scale \cdot EED_LIMIT$
4.	find closest potential parent: $cur_potential_parent = min_rtt(pl)$; if $cur_potential_parent == NULL$, adaptation fails, end. $pl = delete(pl, cur_potential_parent)$; $join(cur_potential_parent)$.
5.	if not accepted by $cur_potential_parent$, go to step 4.

should not switch even if the RTT to the new parent is smaller. This avoids the long and tortuous ROOT paths as in the case of only using RTT.

In AOM, a more suitable parent is the one that is closer to the member than the current parent (i.e., smaller RTT), closer to the ROOT than the member (i.e., smaller EED), and through which the member's new EED is not penalized too much¹. Thus in the above example, C moves down a level as the child of B only if:

$$RTT_{C,B} \leq \alpha * RTT_{C,A} \quad (0 < \alpha < 1) \quad (1)$$

$$EED_{B,ROOT} \leq EED_{C,A,ROOT} \quad (2)$$

and

$$EED_{C,B,ROOT} \leq (1 + p) * EED_{C,A,ROOT} \quad (3)$$

Our simulation results [13] show that $p = 0.2$ produces good quality trees. The final AOM tree is shown in Fig. 1 (c).

To limit the joining overhead, a member looks for a new parent from a *potential set*. Assume the ROOT is at level 0. A member is at level i if its parent is at level $i - 1$, and we say they are 1 overlay hops away from each other. The *potential set* of a level i member is $\{member_{l,h}\}$ where $l = i$ and $h \in \{2, 4\}$. l is the level in the tree and h is the number of overlay hops from this member. A member probes some of the members in its *ancestor set* (will be described in Section III-A.2) to obtain the potential set. If a level i member finds a new parent in the potential set, its level becomes $i + 1$. The member continues the search until no new parent can be found.

2) *Tree improvement*: Both HMTP and AOM have tree improvement algorithms using which a member periodically evaluates its position and switches if necessary. In HMTP, a member randomly selects another member in its ROOT path and explores the branch rooted at that member to find new parent. A member switches to a new parent only if the new parent can offer some RTT improvement. This periodical level-by-level exploration accounts for most of the HMTP overhead.

Since topologically close members are more likely to stay close in the AOM overlay, the tree improvement is carried in a local range. A member periodically probes the members in its *ancestor set* to get the EED and RTT information for the improvement. The ancestor set of a level i member is $\{member_{l,h}\}$ where $l \in \{i, i - 1, i - 2\}$, and $h \in \{1, 2, 3\}$.

¹Without any tolerance to the EED penalty may result in *multiple unicasts*, the worst tree with the link load proportional to the group size near the ROOT.

To let each member know its ancestors, starting from the ROOT, a level i member (parent) tells its children at level $i + 1$ in the PROBEREPLY message about the following members: the $member_{i-1,1}$, all the $\{member_{i,2}\}$ and all the $\{members_{i+1,1}\}$. Members keep the exponential averaged EED and RTT information of their ancestors. In the periodic improvement process, a member re-evaluates its current position and those through the ancestors using the same criteria as in section III-A.1 and joins an ancestor if necessary.

B. Performance monitoring and fault detection

The previous direct-tree protocols, including HMTP, do not actively monitor the EED and therefore adapt to only local network conditions as dictated by RTT increases. In AOM, a member monitors the performance of not only its current ROOT path, but also the paths through its ancestors (backup paths). Therefore, when a fault happens on ROOT path, the member can select a backup path to improve the performance.

An AOM member monitors the ROOT path EED by periodically probing its parent. Both HMTP and AOM require a child to periodically probe its parent. In HMTP, this is to check whether the parent is alive, measure the RTT to the parent, as well as get the ROOT path. In AOM, the child also gets its ancestor information. The EEDs of backup paths are measured in the same way but less frequently to have a lower overhead and because no other important information is exchanged on these paths. To prevent the instability problem, the periodical measurements are smoothed with exponential averaging.

Every time a member updates its EED, it checks whether the ROOT path EED is over the desired limit, and once this happens, a fault is assumed to have happened.

C. Adaptation

1) *The approach*: First, we discuss why HMTP could fail in adapting to the EED fault. Without an adaptation algorithm, the only chance for HMTP members to get away from the EED fault is the periodical improvement, where RTT is used to look for a closer member. Therefore, a member can bypass the EED fault if it finds a closer member not suffering the fault. However, such improvement does not consider the EED metric, therefore is random in nature and may not work in face of EED fault (a closer member is not necessary an EED fault-free member). Another problem is that normally a faulty link affects many members' EEDs, but only one RTT of the member that is closest to the faulty link is increased. The result is that if this member fails to find a better path, the other members do not switch in the improvement because their RTTs to the current parents are still "good" even their EEDs are already over the limit.

The adaptation in AOM is simple, given that each member keeps the information of its ancestors. Once the fault is detected, a member actively probes the ROOT and the ancestors to get the most updated RTT and EED information. Those members that can satisfy its EED requirement become the potential parents. The member always selects the closest potential parent to send a join request. If none of the potential

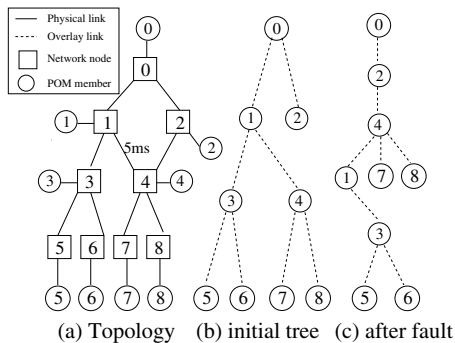


Fig. 2. Example of adaptation

parents accepts it, the adaptation fails. It is worth pointing out the difference between the tree improvement and the fault adaptation. Both of them involve looking for new parents. The tree improvement process creates a more efficient tree, while the fault adaptation process satisfies the EED requirement.

Note our adaptation algorithm neither probes all the group members, nor randomly selects a tree branch to probe. The reason is that a member is unlikely to have a separate path to every other member in the group, but reaches them through a few outgoing links. We assume that in most cases, if the member does not have a good path through any of the close members, it cannot find other better paths. The result is that some fault-free paths may not be found, but the overhead is greatly decreased and scalability is improved. The adaptation algorithm is summarized in Table I.

2) *Efficient adaptation*: Multicast has the “shared fate” problem, i.e., the fault on one link in the tree affects multiple group members. [14] demonstrated that the use of a representative in the “shared fate” area provides effective feedback suppression in reliable IP multicast. The fault adaptation in overlay multicast has a similar problem: if multiple members respond to the same fault by looking for new parents independently, much overhead will be incurred and loop could be formed during the concurrent parent switchings. In fact, for a single fault, if one of the “shared fate” member finds a better ROOT path, all of its descendants become fault-free.

AOM allows a member to decide when to look for a new ROOT path. From the periodic probing, a member knows the current EED of the parent. On similar lines as in [15], if a member under fault finds that its parent is fault-free, it assumes the fault is on the overlay link between them, and it should represent its descendants to find a better ROOT path. Otherwise, it assumes the fault is on the ROOT path of the parent, and leaves the responsibility of adaptation to the parent. If a member fails the adaptation, it notifies the children to look for new ROOT paths. A successfully adapted child can always invite its failed old parent to join it.

3) *Adaptation example*: To illustrate how AOM adapts to the EED fault, we show the adaptation process in a simple scenario. Fig. 2 shows a 9-node network topology in (a), the AOM tree before the fault in (b) and after the fault in (c). In this example, an EED fault occurs on $link_{01}$ at time $t=50$. Fig. 3 shows that member 4 in the initial tree responds and

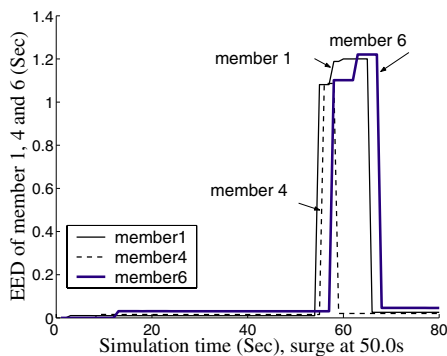


Fig. 3. EED vs. time

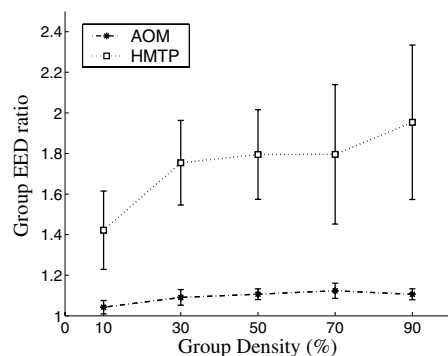


Fig. 4. Tree quality: group EED ratio

recovers earlier than member 1 by switching its sub-cluster (4, 7 and 8) to member 2 (Fig. 2 (c)). The reason is that in this simulation, member 1 happens to use the faulty link in probing member 2, so it cannot find a better path. Later, member 4 invites member 1 to join it. From Fig. 3, we also see that the EED of member 6 also changes with member 1, but this is not because it finds a new parent but because its ancestor, member 1, changes the ROOT path.

IV. SIMULATION RESULTS

We evaluate the properties of AOM to show how the consideration of end-to-end delay (EED) network metric improves the tree quality and the adaptation performance under the dynamic network conditions. We implement the HMTF described in [3] and compare it with AOM.

A. Performance metrics

The network environment is static in evaluating the tree quality, i.e., there is no fault during the simulation. We consider the following metrics:

- *Group EED ratio*: A member’s EED ratio is the ratio between its overlay EED and its EED in the Shortest Path Source Tree (SPST) in IP multicast. It shows the increase of the EED when using overlay multicast. Group EED ratio is the average of the group members’ EED ratios.
- *PDF of the EED ratio*: an EED ratio could fall into the following categories: [0,1) (impossible), [1,2), ..., [6,7), [7, ∞). The PDF shows the distribution of the EED ratios.
- *Average link stress*: assume $LS(i)$ is the number of duplicate packets on a link i . Average link stress is defined as: $\sum_{i, LS(i) >= 1} LS(i) / \sum_{i, LS(i) >= 1} 1$. It reflects the load added to a link by an overlay multicast protocol.
- *CDF of the path length*: Path length is defined as the number of physical links (hops) in a ROOT path. A longer path is bad because of the excessive resources it needs.
- *Recovery speed*: the time that the members suffer large EED before the recovery from a fault.
- *Control overhead*: the total number of the control packets. It’s normalized to per member per second.

B. Simulation background

The simulator is NS-2 [16] and the 100-node network topology is randomly generated by GT-ITM [17]. The links

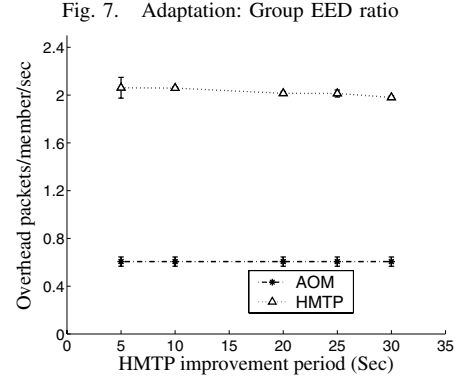
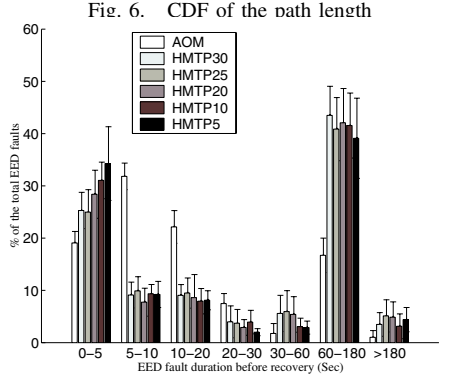
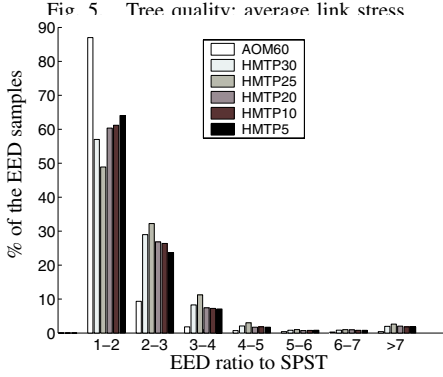
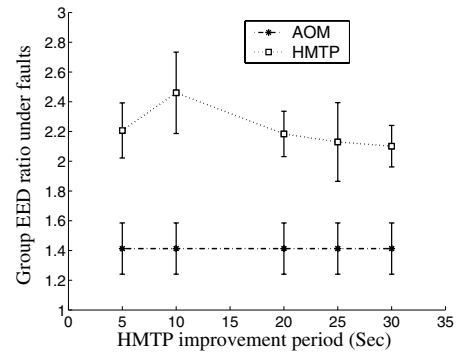
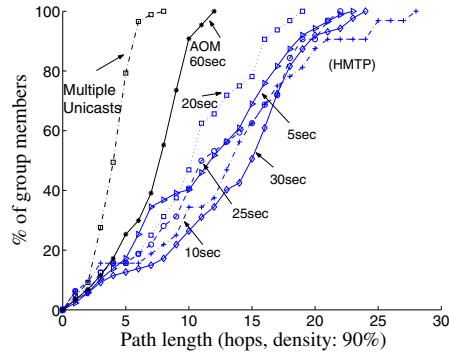
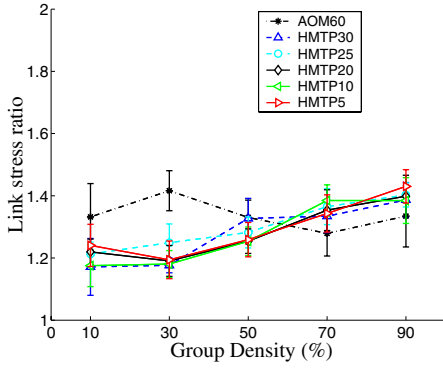


Fig. 8. PDF of the group EED ratio

Fig. 9. EED fault length before recovery

Fig. 10. Overhead packet number, density=85%

have symmetric random delays. We assume that each network node is attached with one end-host (potential HMTP and AOM member) and there is no delay between the network node and the attached end-user. The result is the average of 10 groups with C.I. of 95% wherever applicable. Members join the group in the first 60 seconds in random sequence. In the tree evaluation, simulation time is 300 seconds.

In evaluating the adaptability, EED faults are added on randomly selected links after all the members join the group. Each time, two links are selected and assigned a link delay of 5 seconds (much larger than the normal EED). The EED faults last 180 seconds, then the links are recovered to the original delay. After that, two other links are selected, and so on. The simulation time is 18000 seconds for this part.

C. Simulation results of tree quality

1) *Group EED ratio*: The group EED ratios in AOM and HMTP with different group densities are shown in Fig. 4. Group density specifies the percentage of the end hosts that join the group. Improvement is run every 30 seconds in HMTP, and 60 seconds in AOM.

The results show that a member in AOM has a much smaller EED than in HMTP even with a larger improvement period. One reason is that AOM considers both the end-to-end and local metrics in building the tree. This avoids the long paths that could occur in HMTP, as will be shown in Section IV-C.3. When the group size increases, the group EED ratio in HMTP also increases. This means that the path could be even longer in a large group. As for AOM, the group EED ratio does not

change much with the group density, which means the AOM tree matches better to the underlying network topology.

2) *Average link stress*: In Fig. 5, we show the average link stress for AOM and HMTP with different group densities. We also simulated HMTP with different improvement periods and the results show that decreasing the tree improvement period does not improve the tree quality. At small group densities, AOM has higher average link stress than HMTP, but the stress becomes smaller at higher densities. Overall, the two schemes have comparable link stress, thus the EED ratio advantage of AOM is not obtained by sacrificing the link stress.

3) *CDF of the path length*: An ideal overlay multicast tree is one that the members have short ROOT paths and the link stress is low. One extreme tree is Multiple Unicasts where the path is short but the link stress is high. Another extreme is that the path is extremely long, but the link stress is low. Fig. 6 shows the CDF of the path length in AOM, HMTP and multiple unicasts. As we can see, HMTP has a much longer ROOT path than AOM no matter what the improvement period is. At least 35% of the group members have a path length larger than the longest path in AOM.

D. Simulation results of fault adaptation

Under dynamic network delays, we sample the EED of each group member in the two schemes about every 5 seconds during the simulations. Group density is 85% in this section.

1) *Group EED ratio*: In Fig. 7, the group EED ratio is different from that in Fig. 4, where the network delay is static and the EED ratio becomes constant after the tree is stable. In the dynamic environment, we use the average of the sampled

EEDs as the final EED of a group member, then calculate its EED ratio. For HMTP, we observe whether decreasing the improvement period can make the members respond to the faults more quickly and thus improve the EED performance, and the results are compared to the single result of AOM.

First, the EED ratios in both schemes are larger than that in Fig. 4 due to the EED faults, and AOM still performs much better. Second, increasing the frequency of improvement does not benefit the overall EED performance in HMTP.

2) *PDF of the EED ratio:* For each group, we classify the sampled EED ratios of all the group members. Fig. 8 shows that for almost 90% of the samples, AOM has an EED ratio less than 2, but in HMTP, the best case is about 65%. A large portion of the EED samples in HMTP is between 2 to 4 times of the SPST.

3) *Recovery speed:* Fig. 9 shows the length of the EED fault before recovery occurs. For all the members in a group, the fault lengths in their samples are calculated and categorized. We can see that: 1. Under an EED fault, more than 50% AOM members can find a good path in less than 10 seconds and about 75% members in less than 30 seconds. 2. For HMTP, a higher percentage of members can recover from the fault in less than 5 seconds, especially when the improvement period is small. This is because every member improves its position independently, therefore, when the fault happens and there are better paths for those members whose RTT are affected, parent switching happens quickly. But in AOM, lower level members do not begin the adaptation until the higher level members fail. That is why the second and the third AOM bars have a much higher probability. Overall, a higher percentage of faults can be detected in a short period of time in AOM than in HMTP. 3. In HMTP, at least 40% of the delay faults last up to 180 seconds (the adaptation fails), but less than 20% in AOM. Many of the faults in this bar group happened at the links that virtually it is impossible to find a better path, but we can see that HMTP fails with much higher percentage.

4) *Control overhead:* The control packets in both schemes are collected and shown in Fig. 10. The results are normalized to per member per second. We can see that AOM has much less overhead than HMTP in different improvement periods. Note in HMTP, whenever a member starts the improvement, it selects a tree branch and queries down. Every member does this independently and periodically (30 seconds in the simulation). The improvement period can be increased to decrease the overhead, but this affects the fault recovery speed. In AOM, the member improves the position by periodically probing its ancestors (60 seconds in the simulation). To further decrease this overhead, we can increase the probing period after the tree converges because AOM has the fault detection mechanism. Since an improvement cannot begin until the last improvement finishes, a very small improvement period may not make a big difference from a larger improvement period.

V. CONCLUSION

This paper proposes AOM, the adaptive overlay multicast protocol. The simulation results show that AOM adapts to the

network conditions better and improves the performance of direct-tree protocols. The overhead incurred by AOM is low and the tree quality is better. Future work can be extended to the adaptation of application metrics such as loss rate and throughput. A refinement of AOM is to set up some thresholds such that the adaptation can happen before the fault occurs. In addition, the schemes should be tested further in various network topologies and larger group size. Currently, we are carrying out a more comprehensive evaluation to test the performance of AOM in topologies with up to 1000 nodes. The preliminary results are very promising and show that AOM can scale well to large networks. Although our ultimate goal is to embed AOM in real-world applications such as Internet audio/video broadcasting and large software distribution, in the near future, we will test AOM with real Internet characteristics such as using a snapshot of the Internet to select network nodes as well as the delay and loss measurement.

REFERENCES

- [1] S. Deering, "Host extensions for ip multicasting," *RFC1112*, 1989.
- [2] Y. Chu, S. G. Rao, S. Seshan, and H. Zhang, "Enabling conferencing applications on the internet using an overlay multicast architecture," in *ACM SIGCOMM 2001*, San Diego, CA, Aug. 2001.
- [3] B. Zhang, S. Jamin, and L. Zhang, "Host multicast: a framework for delivering multicast to end users," in *Proceeding of IEEE INFOCOM*, 2002.
- [4] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel, "Almi: an application level multicast infrastructure," in *Proceedings of the 3rd USENIX Symposium on internet technologies and systems (USITS 2001)*, Mar. 2001.
- [5] J. Touch, "Dynamic internet overlay deployment and management using the x-bone," *Computer Networks*, pp. 117–135, July 2001.
- [6] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. J. W. O'Toole, "Overcast: Reliable multicasting with an overlay network," in *Proceedings of the 4th USENIX Symposium on Operating Systems Design and Implementation*, Oct. 2000.
- [7] P. Francis, "Yoid: Extending the multicast internet architecture," 1999, white paper <http://www.aciri.org/yoid/>.
- [8] Y. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," *To appear in IEEE journal on selected areas in communication (JSAC), special issue on networking support for multicast*.
- [9] Y. Chawathe, S. McCanne, and E. Brewer, "An architecture for internet content distribution as and infrastructure service," unpublished, available at <http://www.cs.berkeley.edu/yatin/papers>.
- [10] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *Proceeding Of ACM SIGCOMM*, 2002.
- [11] Z. Li and P. Mohapatra, "Hostcast: A new overlay multicasting protocol," in *Proceeding of IEEE International Conference on Communications*, 2003.
- [12] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proceeding of the 18th ACM SOSP, Banff, Canada*, Oct. 2001.
- [13] S. Wu, "An area-based multicast approach in ip and overlay networks," *Dissertation Proposal*, 2002, <http://www2.sis.pitt.edu/~swu>.
- [14] S. Wu and S. Banerjee, "Multicast feedback control using loss-pattern matching," in *Proceeding of IEEE International Conference on Communications*, 2002, pp. 1253–1258.
- [15] Z. Xu, C. Tang, S. Banerjee, and S.-J. Lee, "Receiver initiated just-in-time tree adaptation for rich media distribution," in *Proceedings of the 13th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, Monterey, California, June 2003.
- [16] "The network simulator - ns-2," <http://www.isi.edu/nsnam/ns/>.
- [17] "Gt-itm: Georgia tech internet network topology models," <http://www.cc.gatech.edu/fac/Ellen.Zegura/graphs.html>.