PAPER

Restoring Delivery Tree from Node Failures in Overlay Multicast*

Zongming FEI† *and* **Mengkun YANG**†, ***Nonmembers***

SUMMARY One of the important problems in overlay multicast is how to deal with node failures and ungraceful leavings. When a non-leaf end host fails or leaves the multicast session, all downstream nodes will be affected. In this paper, we adopt the proactive approach, which pre-calculates a candidate node (called *parent-to-be*) for each node to connect to in case its current parent dies. The goal is to recover the overlay multicast tree quickly so that the disruption of service to those affected nodes is minimized. We combine the local parent-to-be locating and global parent-to-be locating schemes together, in order to take advantage of less interference in the local scheme and the flexibility of the global scheme. The quality of the recovered tree is improved while the responsiveness of the proactive approach is maintained.

key words: *overlay multicast, failure recovery, multimedia streaming, degree constraint, spanning tree*

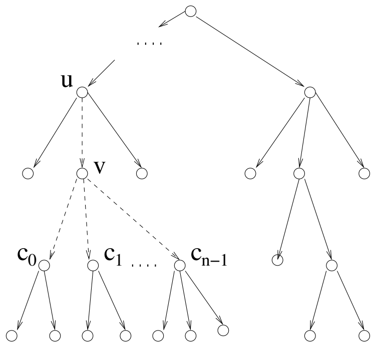

Fig. 1 Recovery from failure in overlay multicast

1. Introduction

Overlay multicast (also known as application-layer multicast) [1, 2] implements the multicast functionality at end hosts rather than routers. End hosts join the multicast tree not only as leaves as in IP multicast, but as non-leaf branching nodes as well. Because end hosts are potentially more susceptible to failures than routers and may leave the multicast group voluntarily, one of the key issues in maintaining the overlay multicast topology is how to reconstruct the overlay tree after nodes fail or leave the multicast session. The time to resume the data flow to those affected downstream nodes is an important measure of the quality of the recovery mechanism.

In this paper, we specifically investigate the problem of restoring the overlay multicast tree for multimedia streaming applications. Usually, significant bandwidth is needed for streaming multimedia data over the network. An end host only has limited bandwidth to participate in media streaming. The number of streams it can send and receive at the playout rate of the media is usually limited, and it can be abstracted as the *degree constraint* on the node. A node can accept more nodes as its children in the overlay tree only if it has not reached its degree limit. These nodes are called *unsaturated nodes* in the later discussion of the paper. Due to the degree constraint on end hosts, a disconnected node cannot always successfully reconnect to the multicast tree by connecting to an arbitrary tree node (e.g., the grandparent) after its parent leaves.

For example, in Figure 1, when node v fails, all of its children $(c_0, c_1, \cdots, c_{n-1})$ and their descendents are partitioned from the tree. The streaming of multimedia data to them will be interrupted. Because of the degree constraint, it may be not feasible to simply let all the children $(c_0, c_1, \cdots, c_{n-1})$ connect to their grandparent (u).

There are two approaches to the recovery of overlay multicast trees. One is the *reactive* approach [3, 4], in which the tree restoration process starts *after* node departures. The affected nodes may have to contact several nodes in the tree before they find an appropriate location. It usually takes quite some time to repair the overlay multicast tree before data can flow to the affected nodes.

The other is the *proactive* approach proposed in our previous work [5], which plans for departures before they really happen. The basic idea is that each non-leaf node in the overlay multicast tree pre-computes a rescue plan before it fails or leaves. The *rescue plan for a node* is to pre-calculate a *parent-to-be* for each of its children. Once it leaves the tree, all its children can contact their respective "parents-to-be" immediately. In this paper, we extend our previous work by taking into account the quality of the restored tree and its effect on the performance of the affected nodes when

Manuscript received January 1, 2004.
Manuscript revised January 1, 2004.
Final manuscript received January 1, 2005.

†The authors are with the Department of Computer Science, University of Kentucky, 301 Rose Street, 2nd Floor, Lexington, KY 40506-0495, U.S.A.

*This work was supported in part by the National Science Foundation under Grants CCR-0204304 and EIA-0101242.

pre-computing the rescue plans. The specific performance metric we consider is the latency from the root of the tree to those affected nodes, because it is an important performance measure for multimedia streaming applications.

Another important difference from our previous work is that we not only consider the local parent-to-be assignment, but propose a global parent-to-be locating technique for improving the quality of the recovered tree and dealing with the cases that cannot be solved by the local assignment technique. In the *local parent-to-be assignment*, a node can take either its grandparent or one of its siblings as the parent-to-be. Although this method can avoid the problems of conflicting parent-to-be and recovery circle caused by concurrent node departures, it is not always feasible when there are not enough residual degrees among siblings. Furthermore, even though the number of residual degrees among siblings is large enough to generate the parent-to-be assignments, the solution may result in a tall tree, which causes a very large latency for some nodes. The *global parent-to-be locating* allows a node to have any unsaturated node (except its descendents) as its parent-to-be. It has more flexibility and provides chances for a node to find a parent-to-be such that its latency after recovery is minimized.

The local parent-to-be assignment and the global parent-to-be locating are integrated as follows. Every non-leaf node tries to find parents-to-be for its children among themselves and their grandparent. If there is not enough residual bandwidth among its children to assign parents-to-be to them locally, or the local parent-to-be assignments can result in long latencies for some children after recovery, it has some of its children locate parents-to-be globally. Extensive simulations are performed to evaluate the effectiveness of the integrated scheme. They demonstrate that the quality of the recovered tree is improved and the responsiveness of the proactive approach is maintained.

The rest of the paper is organized as follows. The local and the global parent-to-be assignments are described in Section 2 and Section 3, respectively. Section 4 integrates the two assignment schemes together and describes the protocol for tree recovery. Performance evaluations are presented in Section 5 and related work is discussed in Section 6. We conclude the paper in Section 7.

2. Local Parent-to-be Assignment

2.1 The Idea of Local Proactive Recovery

The idea of proactive recovery is to make arrangement for affected nodes so that they know who should be their new parents in case of failure. Local parent-to-be assignment focuses on finding parent-to-be among the grandparent and siblings only. For example, in Figure 1, we want to calculate the rescue plan to deal with the failure of non-leaf node v. It should tell v's children $c_0, c_1, \ldots, c_{n-1}$ who will be their new parents. The local parent-to-be assignment will find parent-to-be from grandparent u and siblings $c_0, c_1, \ldots, c_{n-1}$.

The problem can be formulated as calculating a spanning tree among u and $c_0, c_1, \ldots, c_{n-1}$ with u as the root. The parent of c_i in this spanning tree will be its parent-to-be during recovery. Note the difference between the *multicast tree in use* and the *spanning tree to be calculated*. We do not actually add the edges in the calculated tree to the multicast tree in use. The only purpose of the result is to find the parent-to-be for each child.

The remaining problems we need to figure out are the degree-constraint on each node. We use *residual degree* of node x, denoted as $d_r(x)$, to represent how many more edges that a node can be connected to in the *multicast tree in use*, and *available degree* of node x, denoted as $d_a(x)$, to represent how many edges that a node can be connected to in the *spanning tree to be calculated*. Note the available degree of node x is the degree constraint on node x in the tree to be calculated. In order not to increase the burden on the grandparent, we let $d_a(u) = 1$. For each child, it will get one degree released when v fails. Therefore, we have the degree constraint on child c_i to be $d_a(c_i) = d_r(c_i) + 1$.

2.2 Problem Formulations

In this paper, we are more interested in the effect of the calculated spanning tree on the performance of those affected nodes. The performance measure we want to optimize is the end-to-end latency from the root to the end hosts in the multicast tree. Therefore, we can formulate the spanning tree problem as a minimum average latency degree-constrained (MALDC) tree problem, which minimizes the average latency to all the nodes c_i's in the tree. Considering that each c_i is the root of a subtree with different number of downstream nodes, we also formulate a minimum weighted latency degree-constrained (MWLDC) tree problem, which minimizes the weighted latency to all the nodes c_i's in the tree with the number of nodes in the subtree rooted at c_i as the weight. Together with the case of just finding a feasible solution to the degree-constrained spanning tree problem (DC), we formulate the problem as follows.

GIVEN: 1) an undirected complete graph $G = (V, E, r)$ with a weight function $w : E \to \mathcal{R}$, where $r \in V$, $\mathcal{R}$ is the set of real numbers and for an edge $e \in E$, $w(e)$ represents the latency (or cost) of edge e.
2) node degree constraints $d_a(i)$ for node $i \in V$.

DC: Find a spanning tree $T_{dc} = (V, E_{dc}, r)$ rooted at r such that $E_{dc} \subseteq E$, and T_{dc} satisfies the degree constraints that for each node $i \in V$, its degree is less than or equal to $d_a(i)$. This spanning tree is a *feasible solution* to the problem; or

MALDC: Find a shortest path tree $T_{maldc} = (V, E_{maldc}, r)$, such that the average path latency $\frac{\sum_{i \in V} l(i)}{|V|}$ is minimum among all feasible solutions, where the latency from the root to node i is defined as $l(i) = \begin{cases} l(p(i)) + w(p(i), i) & \text{if } i \neq r \\ 0 & \text{if } i = r \end{cases}$, where $p(i)$ denotes the parent of node i in the tree; or

MWLDC: Find a weighted shortest path tree $T_{mwldc} = (V, E_{mwldc}, r)$ such that the weighted latency $\frac{\sum_{i \in V} s(i) \times l(i)}{\sum_{i \in V} s(i)}$ is minimum among all feasible solutions, where $s(i)$ is the size of the subtree rooted at node i.

For example, in Figure 1, to find the parents-to-be for v's children $c_0, c_1, \cdots, c_{n-1}$, we calculate a spanning tree with $V = \{u, c_0, c_1, \cdots, c_{n-1}\}$ and $r = u$. Since the latency of every c_i affects the latencies of all its downstream nodes, we want that c_i with a large subtee can get small latency after recovering from parent failure. The MWLDC formulation achieves this goal by weighing the node latencies and considering the weighted latency of node i defined as $\frac{s(i) \times l(i)}{\sum_{i \in V} s(i)}$. We observe that the MALDC problem is a special case of the MWLDC problem with $s(i) = 1$. In the remaining parts of this paper, we mainly discuss the minimum weighted latency degree-constrained (MWLDC) tree. Both the MALDC and the MWLDC problems are NP-hard [6]. We give a heuristic algorithm in the next subsection.

2.3 A Heuristic Algorithm for Local Parent-to-be Assignment

Given node v's parent u and its children set $C = \{c_0, \ldots, c_{n-1}\}$, Algorithm 1 finds a tree T among u and C with u as the root. Unlike the algorithm proposed in [7], this algorithm consists of several iterations, each of which forms a different tree. In each iteration, we select a different node $c \in C$ with $d_a(c) > 1$ as the child of the root u, and then insert the remaining nodes in set C into the tree one by one, without violating the degree constraint on each node. In the end, we compare all the trees produced by these iterations, and choose the one with the least weighted latency. Through these iterations, the algorithm does a more extensive search in the solution space and therefore is more likely to find a better solution. Because the size of the children set C is usually small and u only accepts one child, the computational cost is acceptable.

2.4 Discussion

The local parent-to-be locating only uses local nodes as the candidates for parent-to-be. The number of links on the grandparent is the same before and after the actual recovery. Therefore, it can accommodate concurrent failures, such as two siblings fail at the same time [5]. However, the local parent-to-be locating cannot guarantee finding a parent-to-be for each affected node. This will happen when the total residual degree of c_i's is not large enough. In general, if the total residual degree of c_i's is k ($k < n - 1$), we can only find parent-to-be for $k + 1$ children. This leads us to consider global parent-to-be locating.

Algorithm 1 *localPtbAssignment(v,C)*

```
Set L_min = ∞, and d_a(c_j) = d_r(c_j) + 1 (j = 0 ... n − 1).
if every c ∈ C has d_a(c) == 1 then
    Select c ∈ C such that it has the least w(u,c) among all
    nodes in C. Return T_l = ({u,c}, {(u,c)}, u).
for every element c in C with d_a(c) > 1 do
    Set A = {u,c}, and B = {c_0, c_1, ..., c_{n−1}} \ {c}.
    Set l(c) = w(u,c), and d_a(c) = d_a(c) − 1.
    Set E_l = {(u,c)}, and L = s(c) × l(c).
    For every y ∈ B, set l(y) = l(c) + w(c,y), wl(y) = s(y) ×
    l(y), and p(y) = c.
    while B ≠ ∅ and Σ_{a∈A\{u}} d_a(a) > 0 do
        if |B| == 1 then
            Let the only element in B be b.
        else
            If B has at least one node with available degree greater
            than 1, select b ∈ B, such that its wl(b) is the smallest
            among all those nodes currently in B with d_a(b) >
            1; otherwise, select b ∈ B such that its wl(b) is the
            smallest.
        Set A = A ∪ {b} and B = B \ {b}.
        Set L = L + wl(b), and E_l = E_l ∪ {(p(b), b)}.
        Set d_a(p(b)) = d_a(p(b)) − 1, and d_a(b) = d_a(b) − 1.
        If d_a(b) ≥ 1: for every y ∈ B, if wl(y) > s(y) × (l(b) +
        w(b,y)), set l(y) = l(b) + w(b,y), wl(y) = s(y) × l(y),
        and p(y) = b.
        If d_a(p(b)) == 0: for every y ∈ B with p(y) == p(b),
        find a node a ∈ A with d_a(a) ≥ 1 and the smallest value
        s(y) × (l(a) + w(a,y)) among all nodes in A, and set
        l(y) = l(a) + w(a,y), wl(y) = s(y) × l(y), and p(y) = a.
    if L / Σ_{a∈A} s(a) < L_min then
        Set L_min = L / Σ_{a∈A} s(a), and T_l = (A, E_l, u).
    Set d_a(c_j) = d_r(c_j) + 1, for j = 0 ... n − 1.
Return T_l.
```

3. Global Parent-to-be Locating

Global parent-to-be locating will not limit the candidates for parent-to-be to the grandparent and siblings. Rather, a node can have any non-descendent unsaturated node as its parent-to-be. This will solve the problem of finding a parent-to-be when the local parent-to-be locating method is unsuccessful. Even when we can find a local parent-to-be for a node, the global parent-to-be locating method can be used to find a parent-to-be that has a smaller latency.

3.1 Maintenance of Unsaturated Node Information

The key problem of the global parent-to-be locating technique is to find an unsaturated node. A simple way to do this is that every node independently searches in the overlay multicast tree based on some searching strategy until some unsaturated nodes are found. However, there are two problems. First, it will generate quite some traffic before a node can find an unsaturated node because it has no knowledge about which paths will lead to some nearby unsaturated nodes. Second, independent searches for parents-to-be by different nodes result in searching the same unsaturated node multiple times.

Our strategy is to let the *Rendezvous Point (RP)* maintain the list of unsaturated nodes. Each unsaturated node periodically sends update about its states to the RP. The states of an unsaturated node x sent to the RP include its residual degree, *ancestor list* $\mathcal{A}(x)$, and the number of its children-to-be $n_{ctb}(x)$. The *children-to-be* of a node x are those nodes that take x as their parent-to-be.

The RP may, or may not, have the information about latencies among nodes in the multicast tree. If it has, it will make a more informed selection of candidate parent-to-be nodes. However, we will not depend on the maintenance of accurate latency information. Also the residual degree information about the unsaturated nodes may even be outdated. It is individual node's responsibility to check out whether the unsaturated nodes returned from the RP really have the capacity to support more children and thus can serve as its parent-to-be, and select the best unsaturated node with the minimum latency from the candidate set by probing those nodes.

3.2 Global Parent-to-be Locating Scheme

When a node c cannot find a parent-to-be locally, it will send a request to the RP, which returns a parent-to-be candidate set P to c. After getting the parent-to-be candidate set P, c probes the nodes in P, and selects a node $f \in P$ as its parent-to-be such that f has at least one residual degree and the value $l(f) + w(f, c)$ is the smallest among all nodes in P, where $w(f, c)$ is the probed latency between f and c. Node c periodically probes this non-localized parent-to-be.

The global parent-to-be locating scheme is implemented in Algorithm 2. It uses the ancestor list $\mathcal{A}(x)$ to determine the valid candidate for parents-to-be. Node p is *globally eligible* to be the parent-to-be of c if and only if 1) c is not in the ancestor list $\mathcal{A}(p)$ of p, and 2) p has at least 1 residual degree. The algorithm also tries to balance the load on those qualified nodes. It computes the ratio of the number of children-to-be over the number of residual degrees of the unsaturated nodes, and chooses β unsaturated nodes with the least ratios, where β is a system design parameter specifying the size of the candidate set.

There are several benefits to do load balancing.

Algorithm 2 *globalPtbCandidates(c)*

Given: A node c, and a system design parameter β specifying the size of the parent-to-be candidate set.
Find: A parent-to-be candidate set P for c.

1: Let a be the number of unsaturated nodes that have a valid timer at the RP.
Set $b = 0$, and $P = \emptyset$.
2: **while** $b < \beta$ and $b < a$ **do**
3: Select x from the set of unsaturated nodes, such that c is not in its ancestor list $\mathcal{A}(x)$, and it has the least $\frac{n_{ctb}(x)}{d_r(x)}$ value.
4: Set $P = P \cup \{x\}$, and $b = b + 1$.
5: Set $n_{ctb}(x) = n_{ctb}(x) + 1$. Remove x from the set of unsaturated nodes.
6: Return P.

First, it helps reduce the occurrences of parent-to-be conflicts in the face of concurrent node failures. Although it cannot completely avoid the parent-to-be conflicts, the node that fails to use its parent-to-be to immediately recover from parent departure can still quickly find another new parent through the RP. Second, the departure of any unsaturated node does not lead too many nodes who are its children-to-be to update their parents-to-be. Third, since each node proactively probes its parent-to-be, load balancing prevents a single unsaturated node from being overloaded by these probes.

The global parent-to-be locating scheme can potentially find a better parent-to-be than the local scheme, but it is not necessarily a good idea to use the global scheme alone. This is because it may result in that more nodes depend on the globally located nodes as parents-to-be and thus increase the possibility of conflicts. The result is either that we have to find new parents-to-be frequently, or that we have to suffer longer recovery time.

4. Integrating Local and Global Parent-to-Be Locating Methods

In this section, we will present a combined parent-to-be locating algorithm. We will also discuss a circle avoidance method for recovery, and a protocol for repairing the tree.

4.1 Integrated Parent-to-be Assignment Algorithm

Algorithm 3 illustrates the integrated parent-to-be assignment scheme. It is based on the local scheme, and gets help from the global scheme when necessary. Each non-leaf node v first tries to find the parents-to-be for its children $c_0, \ldots, c_{n-1}$ only among its parent u and its children. Then v lets some of its children find their parents-to-be through the global parent-to-be locating, if 1) v is not able to find the parents-to-be locally for these children due to lack of enough total residual degree among $c_0, \ldots, c_{n-1}$; or 2) the locally

Algorithm 3 *integratedPtbAssignment(v)*

Given: The child set of node v is $C = \{c_0, c_1, \ldots, c_{n-1}\}$.
Find: The parent-to-be for each node in C.

```
Get T by calling the local parent-to-be assignment algorithm.
For each node in C that is not included in T, use global
  parent-to-be locating algorithm to find a parent-to-be.
Let C' be the set of children of v that are included in T with
  the latency degradation greater than α.
while not all nodes C' have been checked do
    Select c ∈ C' such that it has the greatest weighted latency
      degradation among all nodes that have not been checked
      yet in C', where the weighted latency degradation for a
      node c is the product of its latency degradation and the
      size of its subtree s(c).
    Call the global parent-to-be locating algorithm to get a set
      P for c.
    Choose a parent-to-be p' ∈ P such that it has at least
      one residual degree and the least l(p') + w(p', c) among all
      nodes in P, where l(p') is the latency of p' in the current
      multicast tree.
    Let l(c) be the latency of c in the current multicast tree.
    if l(p') + w(p', c) < (1 + α) × l(c) then
        Set the parent-to-be of c as p(c) = p'.
    Set C' = C' \ {c}.
```

assigned parents-to-be of these children will cause a latency degradation greater than α, i.e., the latency for the node in the recovered tree will be more than $(1+\alpha)$ times its latency in the current multicast tree.

When a non-leaf node v finds a child leave or a child join, it triggers a new recovery plan computation for its children. It will not initiate the pre-computation algorithm if the joined node is its first child or the leaving node is the last child. It will delay the pre-computation if it expects that joins and leaves of other children will happen soon. For example, after a child leaves, it may expect that its grandchildren, if any, will join soon. Each node c that has a non-localized parent-to-be probes its parent-to-be periodically with a certain probing frequency. Whenever this parent-to-be is found to have left the tree, or is not globally eligible to be the parent-to-be of c any more, c contacts the RP and resorts to the global mechanism to find another parent-to-be.

4.2 Interference among Recoveries from Concurrent Failures

When a set of nodes fail at the same time, their children independently resort to the parents-to-be to reconnect to the overlay multicast tree. The following three cases of *recovery interference* may occur. 1) *Failed parent-to-be:* A node contacts its parent-to-be, but this parent-to-be happens to have failed, too. 2) *Conflicting parent-to-be:* The children of different departed nodes have the same node as their parent-to-be, but this parent-to-be does not have enough residual degree to accept all of them as children. 3) *Recovery circle:* A recovery circle is formed among a set of concurrent disconnected nodes $x_0, \ldots, x_{m-1}$ when the parent-to-be of x_j is a downstream node of $x_{(j+1) \bmod m}$, for $j = 0, \ldots, m-1$.

For the first two cases, either the parent-to-be does not respond to the join request, or the parent-to-be finds that it has become saturated by accepting the join request(s) from other nodes. Hence, a timeout or a rejection to the join request can imply a failed recovery due to the interference caused by concurrent node failures. For the third case, we observe that a recovery circle appears only if every parent-to-be in the circle also experiences data disruption. Therefore, if node c sends a join request to its $ptb(c)$ and gets a reply implying that $ptb(c)$ is also experiencing data disruption, it will send a request to the RP to find other nodes that are globally eligible to be its parents-to-be. If its original parent-to-be receives streamed data before a new parent-to-be is identified, c attaches itself to the original parent-to-be; otherwise, it tries to connect to the newly found parent-to-be. This process is described in more details in the next subsection.

4.3 Protocol for Repairing the Tree

We now describe the protocol for nodes to repair the overlay multicast tree once node failures really happen. We assume that there is a heartbeat mechanism for nodes to detect the departure of their parent and children. When the parent leaves, each of its children needs to contact a new parent by sending a *JOIN* message.

When a node receives a *JOIN* request, it will respond based on its own situation. 1) If its residual degree is at least 1, and it is not experiencing data disruption, it will accept the request and reply with *ACCEPT* message; 2) If it has no residual degree, it rejects the request and reply with *REJECT* message; 3) If its residual degree is at least 1, and it is also experiencing data disruption, it will partially accept the request and reply with *PARTIAL-ACCEPT*;

When a node detects its parent leaving, it will start the join process. 1) If the parent-to-be is not empty †, it will first send a *JOIN* request to the parent-to-be and set a timer. If an *ACCEPT* message arrives before the timer expires, it successfully connects to the new parent. 2) If it has no parent-to-be information, it asks the RP to return a parent-to-be candidate set, and then sends *JOIN* requests to all the nodes in this set.

The node also requests the RP to return a parent-to-be candidate set and sends *JOIN* requests to this set in the following cases: 1) no *ACCEPT* arrives before the join timer expires; or 2) a *REJECT* is returned; or 3) a *PARTIAL-ACCEPT* arrives. For the first two cases or the situation with empty parent-to-be, the node from

† The case that a parent-to-be is empty happens when its parent fails before calculating a rescue plan. This cannot be avoided even with our proactive approach. For example, a node just joins the tree, and its parent fails before sending a parent-to-be to it.

the probed set that first responds with $ACCEPT$ becomes the new parent and the $JOIN$ requests to all other nodes are invalidated by sending $CANCEL$-$JOIN$ messages. For the third case, if the node does not get streamed data from its original parent-to-be before the first response from the candidate set arrives, the first responding node in the candidate set becomes the new parent, and $CANCEL$-$JOIN$ messages are sent to the original parent-to-be and all other nodes in the candidate set; otherwise, $CANCEL$-$JOIN$ messages are sent to all nodes in the candidate set.

5. Performance Evaluations

The benefits of integrating the local and global parent-to-be locating together are examined by answering two questions. 1) How many nodes in a multicast tree cannot find feasible local parents-to-be due to lack of enough residual degree? and 2) How much can the quality of the restored tree be improved through the integrated parent-to-be assignment? In addition, we study the *recovery responsiveness*, which refers to how fast the multicast tree can be restored.

In the simulation, we use GT-ITM [8] to generate 1600 node transit-stub topologies as the underlying network. One stub node is randomly chosen as the source and other end-hosts are randomly distributed in stub domains. The total number of end-hosts varies from 200 to 480 in our experiments. The weights of the edges in the graph represent the network-layer link latencies, which vary from 1ms to 245ms. The application-level distance between two end-hosts is the sum of link latencies on the shortest path between them. The total degree of each node is uniformly distributed between 2 and 6, if not mentioned otherwise. All experiments begin with a delivery tree established by a algorithm similar to Algorithm 1. Then end-hosts join or leave the tree dynamically, which are both modeled as a Poisson process with rate $\lambda = 0.2/second$. Each experiment lasts for two hours. In the experiments, the degradation factor α is set to zero because we hope that the latency of a node does not degrade after it recovers from its parent failure. The size of the parent-to-be candidate set, β, is set to 15 if not mentioned otherwise. In addition to the average values, we plot the 90% confidence intervals for Figures 2, 3 and 5.

5.1 Local parent-to-be assignment is not always feasible

As discussed in Section 2.4, for a node v, when the total residual degree k among its n children is less than $n-1$, some of its children cannot find parents-to-be locally. Figure 2 depicts the percentage of tree nodes that can find parents-to-be locally. All other nodes *must* resort to the global parent-to-be locating. The x axis is the average number of node degrees D_{avg} varying from 2.5 to 4. The node degree is uniformly distributed within range $[2, 2 \times D_{avg} - 2]$. Different curves represent the results for multicast trees with different numbers of end hosts. When $D_{avg} = 2.5$, there are about 17% of nodes that cannot find parents-to-be locally. Even for the best case where $D_{avg} = 4$, there are still about 7% $\sim$ 8% such nodes. The observation is that the global parent-to-be locating is necessary since the local parent-to-be assignments cannot find feasible solutions for some nodes.

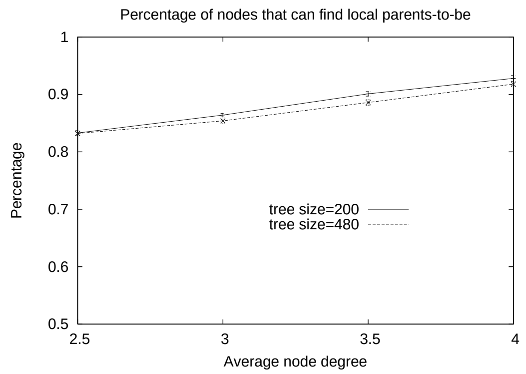

Fig. 2 Percentage of nodes that can find parents-to-be locally

Unlike the local parent-to-be assignment, the integrated approach can always find a parent-to-be for every node in our simulation settings. However, when a node needs to recover from the parent failure, it is possible that its parent-to-be has reached the degree limit, or the recovery protocol finds that there exists recovery interference. In these cases, the parent-to-be is no longer valid as the new parent at the time of recovery. We study the integrated assignment and find these cases only happen with a very small probability, i.e., 0.5% $\sim$ 3%. So the integrated approach can improve the effectiveness of the proactive recovery method over the local approach by 7% $\sim$ 14%.

5.2 Integrated parent-to-be assignment improves the quality of the restored trees

The quality of the restored tree is measured by the latency from the root to the tree nodes after recovering from node failures or leaves. Figure 3 compares the maximum delays of the nodes in the tree from the root, when different parent-to-be assignment schemes are used. We use $D_{avg} = 4$ in this experiment. For the integrated parent-to-be locating, the maximum latency ranges between 2766ms and 3147ms. In contrast, the local parent-to-be assignment produces much higher maximum latency ranging between 3335ms and 4328ms, and its curve climbs up fast as the size of the multicast tree increases. We can find that the integrated scheme is much more effective in bounding the latency of tree nodes, compared with the local parent-to-be assignment.

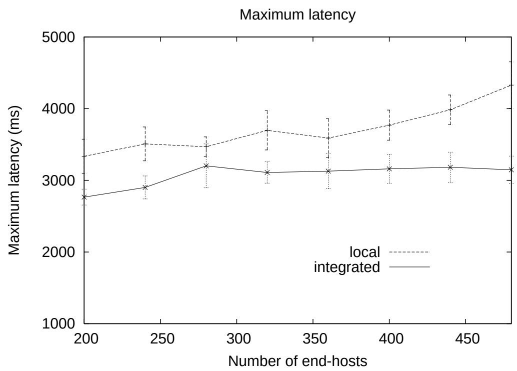

Fig. 3 Maximum latency for all nodes from the root

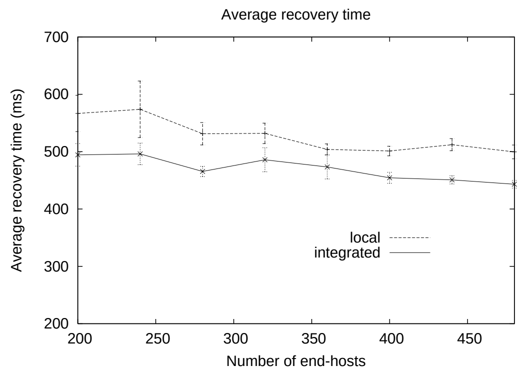

Fig. 5 Average recovery time

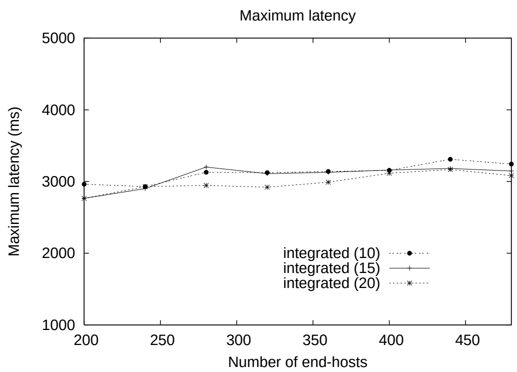

Fig. 4 Maximum latency for all nodes from the root

Also shown in Figure 3 is that the maximum latencies of the restored tree for the integrated parent-to-be assignment climbs up only slightly as the tree size increases, in contrast to the fast climb-up for the local approach. In the local assignment approach, without taking into account how much the node latencies will degrade after recovering from parent failures when locating the parents-to-be, the multicast tree may be recovered in such a way that the latencies for some nodes are larger than before. This effect becomes more obvious when we have more nodes in the tree. The integrated parent-to-be assignment ensures that the node latencies do not degrade as the time goes on. So the maximum tree latency is similar to the maximum latency when the multicast tree is initiated, which increases only slightly as the tree size increases.

Figure 4 depicts how the size of the parent-to-be candidate set, β, affectes the maximum latency of the restored multicast trees, when using the integrated parent-to-be assignment. Three curves labeled by "integrated(β)" correspond to the experiments with β equal to 10, 15 and 20, respectively. We observe that different values of β lead to quite similar maximum tree latencies, all of which are much smaller than that of the local approach. In another word, the efficacy of the integrated parent-to-be assignment on improving the tree quality does not depend heavily on the size of the parent-to-be candidate set, although larger size may lead to some degree of improvement.

5.3 Integrated parent-to-be assignment achieves good recovery responsiveness

The recovery responsiveness is measured by the recovery time, which is the average time for an affected node to find a new parent after a node fails or leaves the tree. Specifically, we calculate the sum of latencies of the links the control message travels before finding the new parent as the recovery time. Here we do not present the comparison between the proactive and reactive recovery approaches since it has been shown that the proactive approach can reconstruct the tree much faster [5]. Rather, we compare the average recovery time of the local parent-to-be assignment and the integrated parent-to-be locating in Figure 5. The average recovery time of the integrated scheme is better than that of the local parent-to-be assignment. This can be explained as follows. The integrated parent-to-be assignment with a parameter α ensures that the node latencies after recovering from parent failures are no greater than $1 + \alpha$ times of the latencies before the recovery. With α set to zero, the latencies of the affected nodes after failure recovery do not degrade. This implies the chances that a node locates a parent-to-be which has large latency to it are small, which contributes to the relatively small average recovery time. The local approach limits the choice of the parent-to-be for a node to a very small set consisting of only its grandparent and siblings. This may have a node assigned with a parent-to-be that has a large latency to it and thus lead to large recovery time. Both approaches show a trend of decreasing recovery time. With more nodes in a tree, the average latency between nodes is reduced. Therefore, the time for messages to travel the links is reduced. This results in shorter recovery time. We also use different β values (10, 15, 20) for the experiment and observe that the value of β does not impact much on the average recovery time. We use $\beta = 15$ in Figure 5.

6. Related Work

The problem of reconstructing overlay multicast trees has been recognized and dealt with in several recent papers. SpreadIt [3,4] proposed *reactive* strategies to deal with node leaving or failure in overlay multicast. They find appropriate places in the subtree of the grandparent or root for the affected nodes after failure happens. Because it uses the reactive approach, the time to find an appropriate place can be long and those affected nodes may compete with each other for available resources at the grandparent or the root.

CoopNet [9] depends on the complete tree topology maintained at the root of the tree for recovery. After each node departure, the root searches this locally maintained tree topology and finds new parents for the children of the leaving node. The efficacy of this reactive recovery approach relies heavily on the accurate and up-to-date states about all nodes maintained at the root. In addition, every reconstruction has to go through the root. All these can impose high control traffic concentration on the root and its neighborhood and the root can become a single point of failure.

Our previous work [5] proposes a proactive approach for tree reconstruction in overlay multicast. It formulates the problem as the minimum spanning tree problem and only uses local parent-to-be assignment. In this paper, we not only use local parent-to-be locating method, but use the global parent-to-be locating method to deal with the cases when the local solution is not able to find the parent-to-be for affected nodes. We also improve the quality of the reconstructed tree by redefining the local assignment problem.

7. Concluding Remarks

Maintaining the overlay multicast tree involves reconstructing it after a node fails or leaves the session. We need to do it in a timely manner so that the disruption to the affected nodes is minimized. The proactive approach is promising in that it precalculates the parent-to-be before the departure actually happens. However, the local proactive assignment method cannot generate a parent-to-be when the total number of residual degrees of local nodes is not big enough. This paper solves this important problem by proposing a global scheme and integrate the two together. We also define a new metric for the local scheme to optimize for multimedia applications. Extensive simulations demonstrate that the quality of the recovery tree is improved while the responsiveness of the proactive approach is maintained.

References

[1] Y. Chu, S. Rao, and H. Zhang, "A case for end system multicast," Proceedings of ACM Sigmetrics, June 2000. Santa Clara, CA.

[2] Y. Chawathe, S. McCanne, and E.A. Brewer, "An architecture for Internet content distribution as an infrastructure service," Feb. 2000. Available at http://www.cs.berkeley.edu/ỹatin/papers/scattercast.ps.

[3] M. Bawa, H. Deshpande, and H. Garcia-Molina, "Transience of peers and streaming media," Proceedings of HotNets 2002, October 2002. Princeton, NJ.

[4] H. Deshpande, M. Bawa, and H. Garcia-Molina, "Streaming live media over a peer-to-peer network," 2001. Technical Report CS-2001-31, CS Dept. Stanford University.

[5] M. Yang and Z. Fei, "A proactive approach to reconstructing overlay multicast trees," Proceedings of the IEEE INFOCOM 2004, March 2004. Hong Kong.

[6] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller, "Construction of an efficient overlay multicast infrastructure for real-time applications," Proceedings of IEEE Infocom'03, March 2003. San Francisco, CA.

[7] S. Shi, J. Turner, and M. Waldvogel, "Dimensioning server access bandwidth and multicast routing in overlay networks," Proceedings of 11th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'01), June 2001. Port Jefferson, New York.

[8] E.W. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," Proceedings of INFOCOM'96, pp.594 –602, 1996. San Francisco, CA.

[9] V. Padmannabhan, H. Wang, and P. Chou, "Resilient peer-to-peer streaming," Proceedings of the 11th IEEE International Conference on Network Protocols (ICNP'03), November 2003. Atlanta, GA.

Zongming Fei is an Assistant Professor in the Department of Computer Science at the University of Kentucky. He received his Ph.D. in computer science from Georgia Institute of Technology in 2000. His research interests include network protocols and architectures, overlay networks, mobile computing, and multimedia systems.

Mengkun Yang is a Ph.D. student in the Department of Computer Science at the University of Kentucky. She received her M.E. and B.E. in computer science from Sichuan University in 2000 and 1997, respectively. Her research interests include overlay networks, peer to peer systems, and multimedia systems.