DEGREE PRE-RESERVED HIERARCHICAL TREE FOR MULTIMEDIA MULTICAST*

Nan Zhang, Yuan-Chun Shi and Bin Chang

Dept. of Computer Science and Technology, Tsinghua University, Beijing, China
Email: {z-n04@mails., shiyc@, cb99@mails.}tsinghua.edu.cn

ABSTRACT

Overlay multicast tree is widely used to support large-scale real-time multimedia applications. The scalability and the robustness are two key issues in the overlay structure design. In this paper, we propose a Degree pre-reserved hierarchical tree for multimedia multicast, called DTree. It organizes the overlay tree in two hierarchies, and combines application-level multicast with IP multicast to achieve a low delay data delivery. A *Degree pre-reserved* mechanism is designed in DTree, which greatly shortens the time to resume the data flow, and achieve a fast recovery. Our simulation results show that DTree has a better delivery performance, compared with other overlay tree. It is quite responsive to the changes in the tree, and almost 3 times faster than other recovery strategies in some cases.

1. INTRODUCTION

The innovation and progress of network and multimedia technologies have greatly promoted the large-scale real-time applications, such as distance learning, audio/video conference, on-line games and so on. For these applications, a cost-effective delivery mechanism such as multicast is quite necessary as the number of users becomes larger.

Overlay multicast constructs a multicast delivery tree among end hosts. Unlike traditional IP multicast, the non-leaf nodes in the tree are normal end host, which are potentially more susceptible to failures than routers and may leave the multicast group voluntarily. A key issue in maintaining the overlay multicast topology is how to restore the overlay tree after nodes depart from the multicast session voluntarily. The time to resume the data is important for multicast applications such as audio/video conference.

In this paper, we propose a hierarchical overlay tree, called DTree. It organizes the overlay in two hierarchies: 1) a backbone tree and 2) the domain tree, which combines application-level multicast and IP multicast together to achieve the data delivery. This structure considers not only delay but also the closeness of overlay nodes. In DTree, we designed a *Degree pre-reserved* mechanism. The basic idea is that the node in the overlay tree accepts the unicast service request with a certain probability during the initial construction. Once the node departure happens, all of its children can find a new parent quickly. The mechanism shortens the time to resume the data flow, and greatly improves on the reconstruction performance of overlay tree.

The paper is structured as follows: The following section discusses related works. Section 3 provides an overview of DTree. Then, Section 4 describes *Degree pre-reserved* algorithm in detail. In Section 5, performance evaluation is presented. We conclude the paper in Section 6.

*This research is supported by NCET-04-0079 (Program for New Century Excellent Talents in University, China).

2. RELATED WORKS

Many research groups have proposed to use the overlay tree for multimedia multicast, because of its simple structure, convenient management, and the low cost, e.g. Yoid [1], HMTP [2], ALMI [3], RMX [4], TOMA [9] and so on.

ALMI [3] is a centralized scheme. A central controller is used to instruct the construction of the overlay tree based on distance and degree constraint information. For an *n*-node tree, $O\left(n^2\right)$ measurements are needed to calculate all the distance, which is not scalable.

HMTP [2] constructs a group-shared tree. It proposes a recovery approach for an overlay tree that each disconnected node contacts its grandparent directly. This mechanism is efficient to shorten the recovery time in some cases, but may over-burden the grandparent and violate its degree constraint.

Yoid [1] is a self-organization overlay tree. It provides a detailed discussion on the looping problem in overlay tree. In Yoid, a new node *A* will join the overlay tree as a child of a node *B* that is the closest node in the random subset *V* in terms of number of hops. However, this measurement is not sufficient to aware underlying topology information, compared with the round-trip-time probing technique [8].

3. OVERVIEW OF DTREE

DTree is a hierarchical overlay tree, it consists of two components: 1) a backbone tree; and 2) the domain tree.

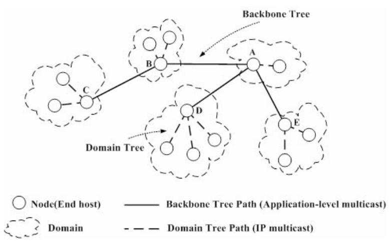

Backbone Tree
A
B
C
D
E
Domain Tree
Node(End host)
Backbone Tree Path (Application-level multicast)
Domain
Domain Tree Path (IP multicast)

Fig. 1 An example of DTree overlay.

Nodes in the backbone tree are in the different domains (In this paper, an IP multicast available area is called a domain), they use application-level multicast to deliver data. In the domain tree, IP multicast is used to achieve data delivery, which improves the delay properties of the backbone tree greatly, and saves the bandwidth resource. Considering that the total number of connections that a node can handle is limited by its service capacity and network conditions, each DTree node's degree is constrained. The degree represents the number of application-level connections a node can establish with other end hosts. A sample DTree overlay is shown in Fig. 1. There are five domain trees in this DTree, rooted at node A, B, C, D and E respectively. The roots of domain tree form the backbone tree. The current degree of node B is 3 (Because an IP multicast connection only occupies one degree).

Each DTree node keeps the following state information, which enable to organize the overlay easily:

- *self_information*: contains the addresses of its parent, the delay from the root via the overlay tree to itself (hereafter refers to as *RTT2TOP*), the degree bound, and some other private parameters of this node.
- *unicast_children_list* and *multicast_children_list*: maintain the real-time status of its children in the backbone tree and domain tree respectively (in this paper, children are the nodes that connect to the parent directly), which are also used to calculate the node's current degree. For this distributed maintenance manner, DTree is quite scalable.
- *candidate_parent_list*: records the information of inter-domain joining targets. The new node chooses a parent from this list during the inter-domain discovery.
- *future_parent_list*: maintains the information of the nodes from the root via the overlay tree to its grandparent. This list plays an important role in DTree recovery process, and is also used as a simple loop prevention and detection technique during the overlay construction process.

Next section provides the formal description for *Degree pre-reserved* mechanism in DTree.

4. DEGREE PRE-RESERVED MECHANISM

This section provides the detailed description of *Degree pre-reserved* mechanism designed in DTree. This mechanism works in the inter-domain discovery process of the overlay tree initial construction, and takes effect in the recovery time.

4.1. DTree initial construction

Considering the hierarchy of DTree, the initial construction is divided into two steps: First, the new node tries to find a parent in the local domain tree; then in the backbone tree.

4.1.1. Intra-domain discovery

Initially, a new coming node obtains the basic information (the session address) of the overlay tree from a well-known rendezvous point. Then it sends a multicast service request to the session address. If a response is received in a certain time, the node will join the domain tree successfully; otherwise, it will process the inter-domain discovery.

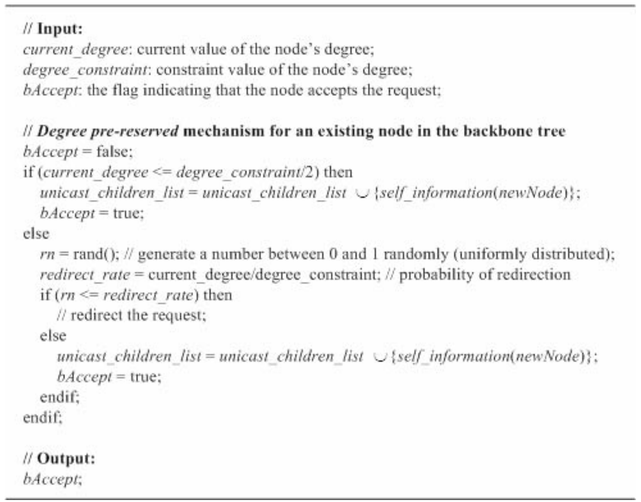
Note that the first node that joins the session in a domain acts as the root of the domain tree. And only the root of the domain tree has the privilege to respond to the multicast service request. Thus a new node need not do parent selection, which makes the node join the overlay tree quickly. Since data is delivered by IP multicast in the domain tree, which occupies only one degree of the node, the root can accept all multicast service requests.

4.1.2 Inter-domain discovery

If intra-domain discovery fails, the new node will process inter-domain discovery to try to join the backbone tree. First, the new coming node adds the root of backbone tree to *candidate_parent_list* to fix it as joining target and sends a unicast service request to it. If the request is accepted, the new node will join the backbone tree successfully; otherwise, the root will send back the information of its backbone tree's children, which will be added to the list. The new node then sends to each of them a unicast service request, and deletes the node that has been tested from the list. If more than one request is accepted, the new node will choose a "*good*" one with the lowest *RTT2TOP* from the top three nodes that send back the acceptance fastest and join the backbone tree successfully; otherwise, the new joining targets will return. Another set of joining process begins. This essentially guides the new node to search down the tree, rather than selecting joining targets randomly.

In the backbone tree application-level multicast is used, so a node that accepts more unicast service requests will consume more bandwidth. Considering the capacity constraint and heterogeneity of the end hosts, DTree is in a degree-constrained form. With the degree-constrained, the inter-domain initial construction becomes more complex. This constraint usually results in the contact time increasing when the disconnected node processes recovery [5].

The goal of *Degree pre-reserved* mechanism designed in DTree is to shorten the recovery time for the disconnected node. The mechanism is described as follows:



```
// Input:
current_degree: current value of the node's degree;
degree_constraint: constraint value of the node's degree;
bAccept: the flag indicating that the node accepts the request;

// Degree pre-reserved mechanism for an existing node in the backbone tree
bAccept = false;
if (current_degree <= degree_constraint/2) then
   unicast_children_list = unicast_children_list ∪ {self_information(newNode)};
   bAccept = true;
else
   rn = rand(); // generate a number between 0 and 1 randomly (uniformly distributed);
   redirect_rate = current_degree/degree_constraint; // probability of redirection
   if (rn <= redirect_rate) then
      // redirect the request;
   else
      unicast_children_list = unicast_children_list ∪ {self_information(newNode)};
      bAccept = true;
   endif;
endif;

// Output:
bAccept;
```

Fig.2 *Degree pre-reserved* mechanism at an existing node in the overlay multicast tree.

The node in the backbone tree accepts the unicast service request with a certain probability. The less spare degree it has, the more probably it redirects the request. The study shows that *Degree pre-reserved* mechanism does not decrease the performance of the initial constructed tree. A pseudo code of *Degree pre-reserved* mechanism at each node is shown in Fig. 2.

4.2. DTree maintenance

In DTree, every node sends keep-alive packet to its parent periodically to maintain the overlay topology, which supports DTree to real-time detect whether there is the node/link failure or link congestion happens. When some nodes lose connectivity to the parent, a failure handling mechanism will be triggered. For the node in different hierarchy, there are two recovery mechanisms designed:

4.2.1 Recovery in the domain tree

When the root of the tree departs, in order to prevent all the children in the domain tree from contacting the nodes in the backbone tree at the same time, the first intra-domain child is pre-appointed to take charge of the inter-domain contact task, and other children in the domain tree join this node directly. This makes the nodes in the domain tree recover from the parent departure quickly.

4.2.2 Recovery in the backbone tree

For the node in the backbone tree losing connectivity to its parent, and the node in the domain tree pre-selected, it tries to find a new parent by sending the unicast service request to a node chosen randomly from its *future_parent_list*. The node receiving the request will accept it as long as has a spare degree. With *Degree pre-reserved* mechanism, nodes in the backbone tree have reserved degrees for the recovery purpose, so it won't take a long time to find a new parent. The study shows that *Degree pre-reserved* mechanism makes a great contribution to the overlay tree recovery process. It shortens the time to resume the data flow greatly.

5. PERFORMANCE EVALUATION

We have conducted several simulation experiments with our DTree, in order to compare with several other common overlay multicast tree. We are mainly interested in the two aspects: the *quality* of the initial constructed tree, and the *responsiveness* of the recovery process.

5.1. Simulation environment

We follow four steps to carry out the simulation:
Step 1: Use GT-ITM [6] to generate 2000 node transit-stub topology as the underlying network.
Step 2: One stub node is randomly chosen as the root node.
Step 3: Choose nodes to join the overlay tree, the number of which varies from 50 to 950. The nodes join or leave the tree dynamically.
Step 4: Construct and maintain a DTree.

In our experiment, the weights of the edges in the graph represent the network-layer link latencies, which range from 0.12ms to 400ms. The degree constraint of each node is uniformly distributed between 2 and 8. It represents the maximal number of IP unicast connections it can establish with other end hosts. The nodes join or depart the tree dynamically. The join and depart are both modeled as a Poisson process with rate 3/minute, which means that there is a node joining and a node departing from the overlay multicast tree every 20 seconds on average.

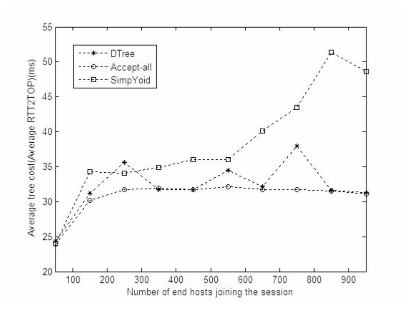
5.2. Comparison: quality of the initial constructed tree

The quality of the initial constructed tree can be measured by two metrics:

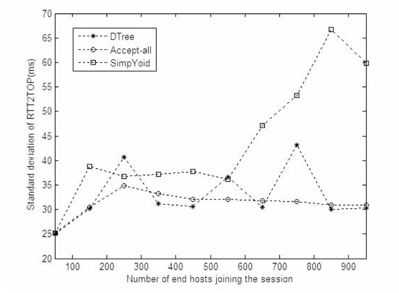
- The average tree cost, which is the average of *RTT2TOP* value.
- The standard deviation of *RTT2TOP* value, which indicates how spread out a distribution is. It is computed as the average squared deviation of each joining node's *RTT2TOP* value from its average.

We compare DTree with other two common overlay tree construction strategies. One is a simplified version of Yoid [1], we call it *SimpYoid*, which chooses the closest node of all nodes currently in the tree as a parent for a new coming node, and is also degree-constrained. The other strategy called *Accept-all*, which is similar with DTree without *Degree pre-reserved* mechanism. Fig. 3 plots the average tree cost and the standard deviation of *RTT2TOP* when the number of nodes in the tree varies from 50 to 950.

Depicted in Fig. 3(a), as the number of nodes joining the tree increases, *Accept-all* strategy has the steadiest the average tree cost, and DTree has a comparable average cost. However, the trees constructed by *SimpYoid* strategy double



(a) Average tree cost comparison.



(b) Standard deviation of *RTT2TOP* comparison.

Fig. 3 Quality of the initial constructed tree.

the cost of those constructed by *DTree*. Fig. 3(b) plots the standard deviation of *RTT2TOP*. *Accept-all* strategy has the smallest and steadiest standard deviation, and DTree has a close performance, but the stability is a little worse. The reason is that *Degree pre-reserved* mechanism makes the node accept the unicast service request with a certain probability during the tree initial construction. *SimpYoid* strategy also performs the worst of all, and it increases as the number of nodes rises.

5.2. Comparison: responsiveness of the recovery process

We use the average recovery time (*ART*) as the performance measures, which is the average time for a disconnected node to find a new parent. We compare DTree with two existing recovery strategies [7]: One is called Grandfather (*GF*), the children of the depart node contact the grandfather directly; another is Root (*RT*), the children contact the root of the overlay multicast tree. Fig. 4 plots the *ART* for the three strategies, as the number of end hosts increases from 50 to 950. From the curves, we find that DTree has the lowest *ART* that can be as low as 500ms. It is about one third of the time of *RT*. The reason is that DTree designed two failure handling mechanisms for the two hierarchies respectively. And *Degree pre-reserved* mechanism worked in the initial construction makes a great contribution to improve the recovery performance.

6. CONCLUSION

In this paper, we present a Degree pre-reserved hierarchical tree for multimedia multicast, called DTree. It uses a

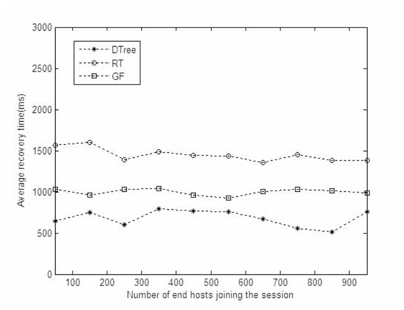


Fig. 4 Average recovery time.

hierarchical structure to support the large-scale multimedia application systems, and combines application-level multicast with IP multicast to implement the multicast functionality, so the data can be delivered with a low delay. A *Degree pre-reserved* mechanism designed in DTree makes the overlay tree achieve a fast recovery. The simulation experiment results shows that *Degree pre-reserved* mechanism does not decrease the performance of the initial constructed tree, and shortens the time to resume the data flow greatly.

7. REFERENCES

[1] P. Francis, "Yoid: Extending the Internet Multicast Architecture," *White paper http://www.aciri.org/yoid/*, April 1999.

[2] B. Zhang, S. Jamin, and L. Zhang, "Host Multicast: A Framework for Delivering Multicast to End Users," *in Proceedings of IEEE INFOCOM 2002*, New York, pp. 1366-1375, June 2002.

[3] D. Pendarakis, S. Y. Shi, D. Verma, and M. waldvogel, "ALMI: An Application Level Multicast Infrastructure," *in proceedings of 3rd USENIX Symposium on Internet Technologies and systems (USITS '01)*, San Francisco, CA, USA, pp. 49-60, March 2001.

[4] Y. Chawathe, S. McCanne, and E. Brewer, "RMX: Reliable Multicast in Heterogeneous Environments," *in Proceedings of IEEE INFOCOM 2000*, Tel Aviv, Israel, pp. 795-804, March 2000.

[5] M. Yang, and Z. Fei, "A Proactive Approach to Reconstructing Overlay Multicast Trees," *in proceedings of INFOCOM 2004*, HongKong, pp. 2743-2753, March 2004.

[6] E. W. Zegura, K. Calvert, and S. Bhattacharjee, "How to Model an Internetwork," *in proceedings of INFOCOM 1996*, San Francisco, CA, USA, pp. 594-602, March 1996.

[7] H. Deshpande, M. Bawa, and H. Garcia-Molina, "Streaming Live Media over a Peer-to-Peer Network," technical Report, Stanford University, 2001.

[8] T. S. E. Ng, Y. Chu, S. G. Rao, K. Sripanidkulchai, H. Zhang, "Measurement-Based Optimization Techniques for Bandwidth-Demanding Peer-to-Peer Systems," *in Proceedings of INFOCOM 2003*, San Francisco, CA, USA, pp. 2199-2209, April 2003.

[9] L. Lao, J. Cui, and M. Gerla, "TOMA: A Viable Solution for Large-Scale Multicast Service Support," *in proceedings of NETWORKING 2005*, Waterloo, Ont., Canada, pp. 906-917, May 2005.