Available online at www.sciencedirect.com
ScienceDirect
Information Sciences 177 (2007) 3613–3626
INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL
www.elsevier.com/locate/ins

A degree-constrained QoS-aware routing algorithm for application layer multicast ☆

Baoliu Ye [a,b,*], Minyi Guo [b], Daoxu Chen [a], Sanglu Lu [a]

[a] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China
[b] School of Computer Science and Engineering, The University of Aizu, Aizu-Wakamatsu City, Fukushima 965-8580, Japan

Received 29 August 2005; received in revised form 29 January 2007; accepted 11 February 2007

Abstract

Application layer multicast (ALM) provides a low-cost solution for multicast over the Internet. It overcomes the deployment hurdle of IP multicast by moving all multicast related functions from network routers to end-hosts. However, since packet replication is performed on end-hosts, the system performance of an ALM is limited by the bandwidth of end-hosts. Therefore, degree-constrained QoS-aware multicast routing becomes one of the key concerns for implementing realtime multicast services, such as continuous streaming applications. In this paper, we claim that the QoS gained by most users will be better evaluated using the overall latency, and we explore the optimization of Degree-Constrained Minimum Overall Latency Spanning Tree (DCMOLST). The process for optimizing the overall latency is divided into two phases, i.e., the initialization phase and the dynamic adjustment phase. In the former phase, we present a heuristic DCMOLST algorithm which negotiates both transmission delay and node bandwidth simultaneously, so as to avoid QoS degradation caused by any single metrics. In the later phase, we define a set of distributed iterative optimizing operations to swap the position between nearby end-hosts for further optimization. Experimental results show that the proposed degree-constrained QoS-aware routing algorithm could improve the overall performance of application layer multicast services.

Keywords: Application layer multicast; Heuristic algorithm; Degree-constrained spanning tree; Overall latency

1. Introduction

In 1990s, IP multicast was proposed as an extension of IP unicast service model to support group communication efficiently over the Internet. Since then great efforts on IP multicast techniques have been made and many innovative approaches have been proposed by both the academia and the industry. However, IP multicast has not been widely employed over the Internet yet. Among the most quoted problems include:

☆ This work is partially supported by the National Basic Research Program of China (973) under Grant No. 2002CB312002; the National Natural Science Foundation of China under Grant No. 60573106 and No. 60402027; the National High-Tech Research and Development Program of China (863) under Grant No. 2006AA01Z199.
* Corresponding author. Tel./fax: +86 25 83596448.
E-mail addresses: yebl@nju.edu.cn (B. Ye), minyi@u-aizu.ac.jp (M. Guo), cdx@nju.edu.cn (D. Chen), sanglu@nju.edu.cn (S. Lu).

0020-0255/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2007.02.025

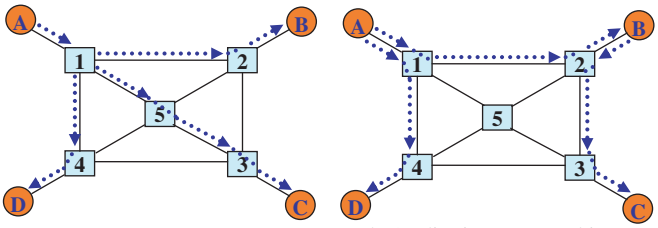

Fig. 1. A comparison between (a) IP multicast and (b) application layer multicast.

the complexity of multicast routing algorithms, the lack of standard multicast routing protocols, and the cost of updating the Internet infrastructure with multicast functionality [13].

Recently, with the flurry of Peer-to-Peer (P2P) computing, the research community began to reconsider whether IP layer is the right layer in which to implement multicast functionality at all, and advocate using overlay network architecture as an alternative solution. This technique is referred as *Application Layer Multicast* (ALM). It shifts the multicast functionality such as membership management and data replication from network routers to end-hosts through software approach. Fig. 1 illustrates the ideas of ALM and IP multicast respectively. Unlike IP multicast (Fig. 1a) for which data packets are replicated by network routers, packet replication in ALM is performed by end-hosts (Fig. 1b). To enable an ALM session, all the involved end-hosts must be organized into an overlay multicast tree where all non-leaf nodes have to forward incoming data to their children. Since data flows are transmitted as unicast packets, ALM deployment can be accelerated easily. In addition, it is possible to support some advanced features such as error recovery and congestion control effectively via leveraging some existing unicast solutions and utilizing application-specific characteristics.

Although ALM has the potential to address most problems associated with IP multicast, it is less efficient than IP multicast. Since each non-leaf end-host must send identical packets to different children via the same link, ALM introduces duplicate packets on physical links and incurs larger end-to-end delay (see Fig. 1b). Besides, as an overlay network based schema, the overall performance of ALM applications is mainly constrained by the available bandwidth of involved end-hosts. However, typical multicast applications such as streaming services are QoS-sensitive applications which are highly sensitive to on network bandwidth, transmission delay and other performance parameters. Therefore, new ALM routing algorithms which are capable of addressing the above challenges will be critical for the success of ALM applications.

In this paper, we evaluate the performance of QoS-aware ALM routing algorithm for latency sensitive real-time applications from a new point of view. We deem that the overall latency is more effective for evaluating the QoS perceived by most users and present the optimizing problem of *Degree-Constrained Minimum Overall Latency Spanning Tree* (*DCMOLST*). To the best of our knowledge, we are the first to tackle this problem. In order to minimize the overall latency, our scheme divides the optimizing procedure into two phases, i.e., the initialization phase and the dynamic adjustment phase. In the former phase, we present a heuristic DCMOLST algorithm to construct the multicast tree through negotiating transmission delay and node bandwidth simultaneously, so as to avoid QoS degradation caused by any single metrics. In the later stage, we define a set of distributed iterative optimization operations for further improvement based on the philosophy of "improving the overall performance at the cost of sacrificing some nodes' local benefit".

The rest of this paper is organized as follows: Section 2 reviews the related work. Section 3 gives the formal definition of the problem to be addressed. Section 4 describes the details of the proposed algorithm. Section 5 presents the group management mechanism of our proposal. Section 6 shows the experimental results. Finally we summarize our work in Section 7.

2. Related work

ALM techniques have been extensively studied in recent years. Several ALM protocols, such as Narada [3], Scribe [2] and QRON [8], have been proposed by the research community. Most of previous work assume that

all end-hosts participating in a multicast session are homogeneous in their capability and focus on addressing network dynamics as well as reducing topology maintenance overhead. However, most multicast applications are streaming oriented, it is essential to satisfy QoS constraints, such as transmission delay, network bandwidth. From the perspective of routing design, QoS-aware multicast routing aims to find the best distribution tree with respect to some constraints, so as to better utilize network resources.

A lot of work on QoS-aware IP multicast routing has been proposed in the last decade. Wang and Hou [11] classified these work into 12 different types according to their objective functions and tree/link constraints. The Shortest Path Tree (SPT) is a source routing algorithm which aims to minimize the sum of the weight along each individual path from the source to a receiver in the multicast tree. Bellman–Ford algorithm and Dijkstra algorithm are two widely accepted SPT algorithms used to solving tree constrained problems. The Minimum Spanning Tree (MST) tries to span all the group members with minimized total weight. Prim's algorithm is a classical algorithm dedicated to this kind of optimization problem. The Steiner tree is a well-known NP-complete problem which focuses on minimizing the total cost of a multicast tree. KMB heuristic is a good candidate for generating such a tree. The Steiner tree problem reduces to the MST problem when the multicast group extends to all nodes within the network. If the objective is to find a minimum cost tree with other QoS constraints, then the Steiner tree problem is termed as Constrained Steiner Minimum Tree (CSMT) problem. KPP [6] and BSMA [14] are two source-based routing algorithms for addressing the delay constrained Steiner tree optimization problem. Wu et al. [5] solved the multiple-constrained multicast problem by extending existing single-constraint Steiner tree algorithms and proposed a novel genetic algorithm named MCMGA. Jia [12] presented a distributed greedy heuristic algorithm to address this problem. QMRPD [7] is a QoS routing protocol with dynamic group topology. It attempts to reduce the overhead of constructing a QoS-constrained Steiner tree. Although Steiner tree algorithms could be used to solve tree optimization problems, they cannot fulfill the tree constraints on an end-to-end basis.

Previous QoS-aware algorithms on IP multicast give insights to construct ALM trees efficiently. However, they are unsuitable for the new multicast environment. ALM networks are different from traditional network in network model, cost metrics and routing constraints. In ALM, data replicating and forwarding are performed by end-hosts. Hence, the bandwidth of end-hosts' network interface is the main constraint in routing design. One of the key issues on ALM protocol research is how to construct a bandwidth constrained multicast tree. It is usually referred as the *degree-constrained multicast tree problem*. EMS [3] is a QoS-driven ALM protocol which uses a variant of the shortest widest path algorithm. It considers both bandwidth and latency simultaneously when constructing a multicast tree, but prioritizes bandwidth over latency. Shi et al. [9] proposed two centralized greedy heuristics to construct a Minimum Diameter Degree-Bounded Spanning Tree (MDDBST) and a Bounded Diameter, Residual-Balanced Spanning Tree (BDRBST), respectively. Riabov et al. described a constant-factor approximation algorithm to construct a degree-constrained multicast tree with the objective of minimizing the maximum communication delay. Brosh [1] presented a logarithmic approximation algorithm and an alternative heuristic solution to capture the trade-off between the desire to select shortest path trees and the need to constrain the load on the hosts. OMNI [10] is an iterative distributed solution for the min average-latency problem. However, this schema is for infrastructure based ALM, rather than P2P systems. Recently, Jia [4] gave a set of novel distributed algorithms on m-D mesh overlay configurations for short delay and low resource consumption application layer multicast.

In practice, end-hosts are generally heterogeneous in their available bandwidth and the edge delay between nodes. Therefore, there may exist a class of high degree nodes with large delay and another class of low degree nodes with small delay. In such a scenario, if we attach low delay nodes to the position near the root firstly, the overall latency of the resulted multicast tree may be increased due to the bandwidth bottleneck of these nodes. On the other hand, if we place high degree nodes near the root, the overall delay of the multicast tree can also be enlarged by these long latency edges close to the root. Thus, previous greedy algorithms based on single metrics are not helpful for creating an ALM tree with minimized overall latency. It is essential to capture the join priority of nodes according to both the edge delay and the available degree. In order to achieve the objective, we provide a heuristic algorithm to initialize the multicast tree by extending Prim algorithm and present a set of distributed optimizing operations to adjust the multicast tree adaptively during the multicast session.

3. Problem formulation

An overlay network is a fully interconnected logical network formed by end-hosts. It utilizes the regular unicast infrastructure to provide communication services among end-hosts, while not requiring any special support from the network level. Therefore, the delay experienced by a message traveling between end-hosts equals to the unicast transfer delay over the underlying physical network.

According to the graph theory, an overlay network could be modeled as an undirected weighted graph $G = (V, E)$, where V is a set of vertices representing end-hosts and E is a set of overlay edges denoting the unicast paths. $|V|$ and $|E|$ are the number of end-hosts and logical links of G, respectively. These two parameters satisfy the following relationship: $|E| = |V|^*(|V| - 1)/2$. Let $d(e)$ represent the transfer delay of edge $e \in E$ and $b(v)$ denote the available bandwidth of an end-host, we first give several definitions regarding overlay multicast networks.

Definition 1. An *ALM Tree* is a subgraph of $G = (V, E)$ that spans all the nodes in V and can be represented by a tri-tuple $T(s, M, E')$, where $s \in V$ is the source node, $M = V - \{s\}$ a set of receiving nodes, and $E' \subset E$ a set of edges forming a multicast tree. We define several functions related to tree T as follows:

- $P_n(T)$: The parent of node n.
- $G_n(T)$: The grandfather of node n.
- $C_n(T)$: The number of children of node n, i. e., $C_n(T) = |n' : P_{n'}, (T) = n|$. For a degree-constrained ALM tree, $C_n(T)$ must satisfy $C_n(T) <= d_n(T) - 1$.
- $L_n(T)$: The overlay latency from source node s to node n $(n \in M)$. Here we have $L_n(T) = L_{P_n(T)}(T) + d(P_n(T), n)$
- $N_n(T)$: The total number of nodes within a subtree rooted at node n. Here $N_n(T) = \sum_{\forall P_i(T)=n} N_i(T)$ We denote the subtree of tree T rooted at node n as $S_n(T)$.

Definition 2. *Node Degree* is a non-negative integer representing the maximum number of neighbors that a node can support in a multicast session/tree. Let $d_n(T)$ denote the node degree of node n in a multicast session T, $\overline{d(T)}$ the average node degree of T. $d_n(T)$ implies that node n can simultaneously forward incoming packets to at most $d_n(T) - 1$ other nodes.

Node degree is determined by the network bandwidth of a node. In this paper, we suppose the transmission rate of a multicast session (such as streaming application) to be a constant bit rate r. $d_n(T)$ is calculated by formula (1):

$$d_n(T) = \max\{\lfloor (b(n))/r \rfloor, 0\} \tag{1}$$

Generally speaking, high node degree is helpful for constructing a low-latency multicast tree.

Definition 3. *Overlay Latency* is the data transmission delay between two nodes over an ALM tree. Let $l_T(i,j)$ represent the overlay latency between node i and j. Since i and j may communicate with each other indirectly over a path $\langle i, v_1, \ldots, v_n, j \rangle$ in the ALM tree, $l_T(i,j)$ is the sum of transmission delay along the path. Therefore, $L_T(i,j) = d(i, v_1) + \sum_{m=1}^{m=n-1} d(v_m, vm + 1) + d(v_n, j)$. Obviously, this value may be larger than corresponding unicast delay. We further define the *Overall Latency* as the sum of overlay latency from the source node to all other nodes within the ALM tree. Let $D_s(T)$ be the overall latency of T rooted at source node s, we have $D_s(T) = \sum_{n \in M} L_n(T)$.

According to Definition 1, $D_s(T)$ can also be computed as follows:

$$D_s(T) = \sum_{\forall P_i(T)=s} ((N_n(T) + 1) \times l(s,i) + D_i(S_i(T))) \tag{2}$$

Overall latency is more reasonable for evaluating the overall QoS performance perceived by users. The lower the overlay latency, the better the QoS experienced by most users. We think it is one of the important optimizing goals of ALM QoS routing design and formally define the DCMOLST as follows.

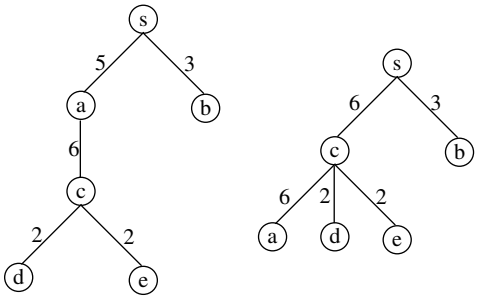

Fig. 2. An example of DCMOLST tree: (a) MDDBST and (b) DCMOLST.

Definition 4 (*Degree-Constrained Minimum Overall Latency Spanning Tree* (***DCMOLST***)). The optimization problem of DCMOLST is to find an ALM tree $T_{\min}(s, M, E')$ over an undirected complete graph $G = (V, E)$, where

(1) $E' \subset E$, $V = M \cup \{s\}$;
(2) $C_n(T_{\min}) < d_n(T_{\min})$ for any node $v \in M$;
(3) $D_s(T_{\min}) \leqslant D_s(T')$ for any other trees T' that satisfies with both (1) and (2).

DCMOLST tries to capture the overall performance of the resulted ALM tree. However, previous solutions such as the min max-latency algorithm and the minimum diameter algorithm cannot guarantee this criteria. Fig. 2 illustrates an example. Fig. 2a depicts an MDDBST tree constructed by the centralized greedy heuristic proposed in [9]. We can see that the overall latency of the resulted tree in Fig. 2a is $4 \times 5 + 3 \times 6 + 2 \times 2 + 3 = 45$. If we make a trade-off between the edge delay and the node degree when determining the position of end-hosts, then it is possible to yield an ALM tree with better QoS performance (see Fig. 2b). Different from Fig. 2a, b lets node c join the ALM tree firstly, although its distance to s is longer than that of node a. Since node c has larger node degree, all the remained nodes (node a, d and e) could be directly attached to it, instead of introducing a new branch node. Consequently, the overall delay is further reduced to $4 \times 6 + 6 + 2 + 2 + 3 = 37$.

4. The DCMOLST algorithm

4.1. Proof of feasible solution

Most of the existing work on degree-constrained multicast tree assume $d_n(T) \geqslant 2$. However, in practice, it may encounter cases where some end-hosts have no enough bandwidth to receive and duplicate packets simultaneously. Meanwhile, leaf nodes only need to receive incoming packets. So it is possible to find a feasible solution to address the problem, even if there exists nodes with $d_n(T) = 1$. We first state and prove the sufficient and necessary condition of a feasible solution for the degree-constrained spanning tree problem.

Theorem 1. *For the degree-constrained spanning tree problem $T(s, M, E')$ of a complete graph $G = (V, E)$ with $|V| = v$ and node degree $d_n(T)$ of node n ($n \in V$), it has a feasible solution if and only if $d_n(T) \geqslant 1$ and $\sum_{n=1}^{v} d_n \geqslant 2(v-1)$.*

Proof. A v-node spanning tree has $v - 1$ edges. Each edge results in 2 degrees used, one for each node. So the total used degree is $2(v - 1)$. Moreover, each node must be connected to the spanning tree, i.e., $d_n(T) \geqslant 1$. Therefore, these are necessary conditions.

Conversely, assume $d_n(T) \geqslant 1$ and $\sum_{n=1}^{v} d_n \geqslant 2(v-1)$. For $v = 2$, we can obviously generate a feasible tree by connecting the two nodes together. For a graph with v ($v > 2$) nodes, we assume that node n has the smallest degree constraint $d_n(T) \geqslant 1$. If $d_n(T) = 1$, we know the total degree of the rest v-1 nodes will be

$\left(\sum_{i=1}^{v} d_i\right) - d_n \geqslant 2(v-3)$. If $d_n(T) \geqslant 2$, then $d_n(T) \geqslant 2$ for $0 \leqslant i \leqslant v$ and $i \neq n$. We also have $\left(\sum_{i=1}^{v} d_i\right) - d_n \geqslant 2(v-3)$. So we can construct a spanning tree using the rest of $v-1$ nodes and the total used degree of the tree is $2v-4$. This tree must have a node with free degree of at least 1. Connecting this node with node n will form a spanning tree with v nodes. □

4.2. Heuristics for ALM tree initialization

Suppose m nodes send join requests to the source node s prior to the starting time of a multicast session. We must organize these nodes into an initial ALM tree with respect to the optimizing objective. According to the description in previous section, we should determine the join priority of end-hosts based on both the edge delay and the node degree, so as to optimize the overall latency. Concretely, the priority ought to be in direct proportion to node degree and in inverse proportion to edge delay. Therefore, there needs a tradeoff to negotiate these two parameters. In this section, we present a heuristic algorithm to initiate the DCMOLST tree by extending Priming algorithm.

The heuristic algorithm maintains the following three node sets:

M_{avail}: This set consists of all the nodes which have already been attached to the tree and still has residual node degree.
M_{off}: It includes all the nodes to be attached.
M_{full}: This node set includes nodes that have been connected to the tree but cannot accept any additional child.

Initially, $M_{\text{avail}} = \{s\}$, $M_{\text{off}} = M$, $M_{\text{full}} = \phi$. During the initialization phase, M_{avail}, M_{off}, M_{full} are mutually exclusive sets which satisfy $M_{\text{avail}} \cup M_{\text{off}} \cup M_{\text{full}} = \{s\} \cup M$. Before describing the details of the heuristic DCMOLST algorithm, we first give the definition of the minimum ALM distance from an off-tree node to the source node s.

Definition 5. The minimum ALM distance from an off-tree node n_1 ($n_1 \in M_{\text{off}}$) to the source node s is the shortest overlay path from n_1 to s via any node already in T with available degree. Let $\delta(n_1, T)$ denote this distance, it could be expressed as follows:

$$\delta(n_1, T) = \min(L_{n'}(T) + l(n', n_1)) \quad \forall n' \epsilon M_{\text{avail}} \tag{3}$$

Node n' is termed as the **access node** of node n_1.

Similar to Prim algorithm, the initialization process starts from the source node s and performs the following procedure repeatedly till $M_{\text{off}} = \phi$.

(1) Select a node n ($n \in M_{\text{off}}$) with the highest priority and add it to the existing component T through its access node n'. Let $d_n(T) = d_n(T) - 1$, $d_{n'}(T) = d_{n'}(T) - 1$.
(2) Node n computes and saves its overlay latency to the source node s in the current component T, i.e., $L_n(T)$.
(3) Let $M_{\text{off}} = M_{\text{off}} - \{n\}$. If $d_n(T) \geqslant 1$, then $M_{\text{avail}} = M_{\text{avail}} \cup \{n\}$; otherwise $M_{\text{full}} = M_{\text{full}} \cup \{n\}$. If $d_{n'}(T) = 0$, then $M_{\text{full}} = M_{\text{full}} \cup \{n'\}$ and $M_{\text{avail}} = M_{\text{avail}} - \{n'\}$.
(4) If node n joins M_{avail} or node n' moves from M_{avail} to M_{full}, then each node n_0 in M_{off} recomputes its $\delta(n_0, T)$ according to formula (3) and updates its access node.
(5) If $M_{\text{off}} \neq \phi$, then each node $n_0 \in M_{\text{off}}$ recalculates its priority as follows:

$$P_T(n_0) = \alpha \times \frac{\delta_{\min}}{\delta(n_0, T)} + (1-\alpha) \times \frac{d_{no}(T)}{d_{\max}} \quad \forall n_0 \epsilon M_{\text{off}} \tag{4}$$

where α ($0 \leqslant \alpha \leqslant 1$) is a tunable parameter used to negotiate the weight ratio between edge delay and node degree, $\delta_{\min}$ and $d_{\max}$ are the minimum value of $\delta(n_0', T)$ and the maximum value of $d_{n_0'}(T)$ for all node $n_0'(n_0' \in M_0 ff)$, respectively.

Algorithm 1 gives the pseudocodes of the heuristics for ALM tree initialization.

Algorithm 1. The heuristic DCMOLST algorithm

for each $n \in V$
  if ($d_n < 1$) exit( );
  else $D_T = D_T + d_n$;
if ($D_T < 2(v-1)$) exit( );
$M_{\text{avail}} = \{s\}; M_{\text{off}} = M; M_{\text{full}} = \phi$;
while ($M_{\text{off}} \neq \phi$){
  for each $n \in M_{\text{off}}$
    for each $n_1 \in M_{\text{avail}}$
      if ($L_{n_1}(T) + l(n, n_1) < \delta(n, T)) \delta(n, T) = L_{n_1}(T) + l(n, n_1)$;
$\delta_{\min} = \min(\delta(n, T)); d_{\max} = \max(d_n(T))$;
  for each $n \in M_{\text{off}}$
    $P_T(n) = \alpha \times \frac{\delta_{\min}}{\delta(n,T)} + (1-\alpha) \times \frac{d_n(T)}{d_{\max}}$;
  let $n \in M_{\text{off}}$ be the node with maximum $p_T(n)$;
  Connect n into multicast tree via its access node n_1;
  $M_{\text{off}} = M_{\text{off}} - \{n\}$;
  $d_n = d_n - 1$;
  $d_{n_1} = d_{n_1} - 1$;
  if ($d_n = 0$) $M_{\text{full}} = M_{\text{full}} \cup \{n\}$;
  else $M_{\text{avail}} = M_{\text{avail}} \cup \{n\}$;
  if ($d_{n_1} = 0$)$M_{\text{full}} = M_{\text{full}} \cup \{n_1\}$;
}

The complexity analysis of this algorithm is straight-forward. The main time consumption of Algorithm 1 is to update the minimum ALM distance from each off-tree node to s (i.e., $\delta(n, T)$) after adding a new node to the partial tree. As described above, M_{off} and M_{avail} are two exclusive sets. For each outer iteration (i.e., for each $n \in M_{\text{off}}$), the inner iteration (i.e., for each $n_1 \in M_{\text{avail}}$) requires $|M| - |M_{\text{off}}|$ time to update $\delta(n, T)$. The total time of the heuristic algorithm is

$$T(|M|) = \sum_{i=0}^{|M|} i = \frac{|M| \times (|M|+1)}{2}$$

Thus, the time complexity of Algorithm 1 is $\Theta(|M|^2)$.

4.3. Local transformation operations

Since our heuristic DCMOLST algorithm is not a greedy one, there may have available degree on some branch nodes near the root after the initialization. Thus it is possible to further optimize the overall latency by changing the position of local nodes with these unused degrees and exchanging the position of nearby nodes. In this section, we define four local iterative optimizing operations for further improvement. Our philosophy of defining these optimization operations is "improving the overall performance of a multicast tree at the cost of sacrificing some local nodes' QoS". An operation is performed only if it could reduce the overall latency of the ALM tree. Each node in the ALM tree attempts to perform these operations periodically.

4.3.1. Parent–child position swap

This is a child-driven operation which swaps the position of a parent node and a child node in the tree. This operation requires the child node has unused degree. Assume node g and p are the grandparent node and parent node of node c, respectively. The parent–child swap operation will be performed if and only if formula (5) is true:

$$N_p(T) \times l_p(T) > (L_g(T) + l(g, c)) \times N_p(T) + (N_p(T) - N_c(T)) \times l(c, p) \tag{5}$$

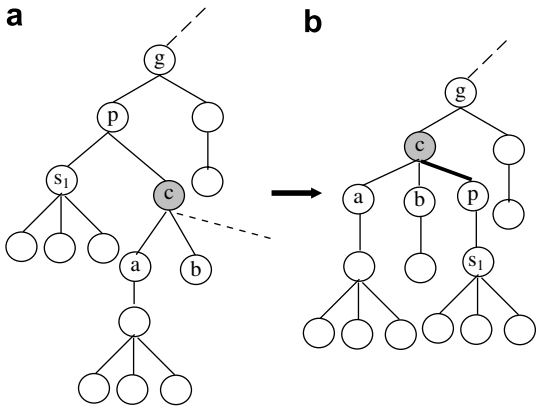

Fig. 3. An example of parent–child position swap.

Fig. 3 demonstrates a parent–child position swap operation. In Fig. 3a, since c has free degrees, it initiates a parent–child position swap test according to formula (5) and sends a swap request to its parent node p. Fig. 3b shows the resulted tree after swapping.

Actually, formula (5) could be deduced as follows:

$$\begin{aligned} & N_p(T) \times L_p(T) > (L_g(T) + l(g,c)) \times N_p(T) + (N_p(T) - N_c(T)) \times l(c,p) \\ & \quad \Rightarrow N_p(T) \times L_p(T) - (L_g(T) + l(g,c)) \times N_p(T) > (N_p(T) \times l(c,p) - N_c(T)) \times l(c,p) \\ & \quad \Rightarrow N_p(T) \times (L_p(T) - L_g(T) - l(g,c) - l(c,p)) > -N_p(T) \times l(c,p) \\ & \quad \Rightarrow N_p(T) \times (l(g,p) - l(g,c) > (N_p(T) - N_c(T)) \times l(c,p) \end{aligned} \tag{6}$$

From the above inequation, we can see that the distance between p and g must be larger than that of c and g. This is a precondition of a parent–child position swap operation.

For a parent–child position swap operation, only the children of parent node are involved. Therefore, the complexity of this operation is correlated to the average node degree $\overline{d(T)}$.

4.3.2. Grandchild node promotion

This is a grandfather-driven operation. If node g has residual degrees, then one of its grandchildren will be promoted as a its child on condition that this operation can reduce the aggregate subtree latency for the DCM-OLST. If multiple grandchildren of g are eligible for the promotion, the grandchild which maximally reduces the overall latency is selected. Assume C is the node set consisting of all the grandchildren of g, i.e., $C = \{c | G_c(T) = g\}$. Node g chooses a node from C using Algorithm 2.

Algorithm 2. Grandchild selection algorithm

```
CanNode = null; CanNode represents the candidate node, initially it is null
totalCost = 0;
for each c ∈ C
  if (L_c(T) − L_g(T) > 0)
    if ((L_c(T) − L_g(T)) × N_c(T) > totalCost) {
      totalCost = (L_c(T) − L_g(T)) × N_c(T);
      CanNode = c;
  }
```

According to Algorithm 2, the complexity of a grandchild node promotion is determined by the size of its grandchildren. The complexity equals to $\Theta(\overline{d(T)}^2)$.

Fig. 4 shows an example of grandchild node promotion operation. In this figure, since the promotion of c_2 can maximally reduce the maximum subtree latency, it is moved from node a to node g.

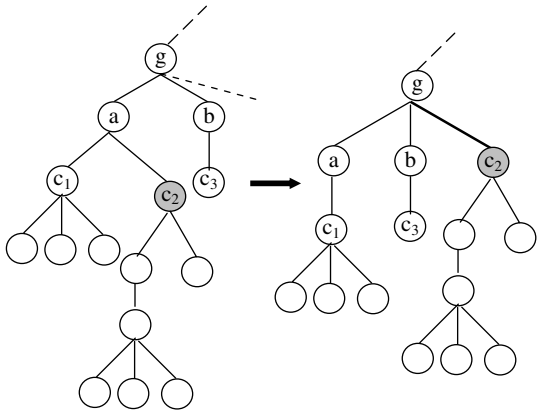

Fig. 4. An example of grandchild node promotion.

4.3.3. Nephew node movement

Node n is a nephew of node u, so u is the uncle of n, if $G_n(T) = P_p(T)$ and $P_n(T) \neq u$. Nephew node movement is an uncle-driven operation which selects one of its nephew nodes to act as its child. Similar to the grandchild node promotion operation, if multiple nephew nodes are eligible, the one which can maximally reduce the subtree latency is chosen. Algorithm 3 gives the pseudocodes of nephew node selection. Note that S is a set consisting of all the nephew nodes of node u. Similar to the grandchild node promotion, the complexity of the nephew node movement is $\Theta(\overline{d(T)}^2)$.

Algorithm 3. Nephew selection algorithm

```
CanNode = null; CanNode represents the candidate node, initially it is null
totalCost = 0;
for each n ∈ S
  p = P_n(T);
  if (L_p(T) + l(p,n) − L_u(T) − l(u,n) > 0)
      if ((L_p(T) + l(p,n) − L_u(T) − l(u,n)) × N_n(T) > totalCost) {
        totalCost = (L_p(T) + l(p,n) − L_u(T) − l(u,n)) × N_n(T);
        CanNode = n;
  }
```

Fig. 5 shows an instance of this operation. In this figure, u has free node degrees and select node n_1 as the eligible node for the nephew movement operation.

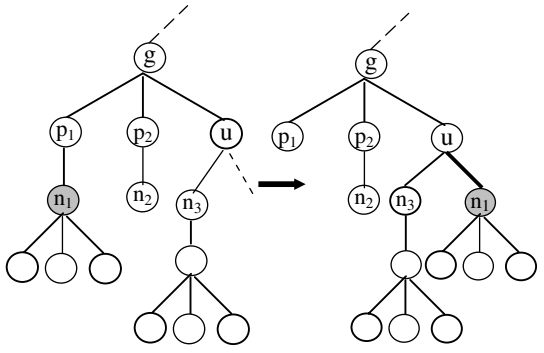

Fig. 5. An example of nephew node movement.

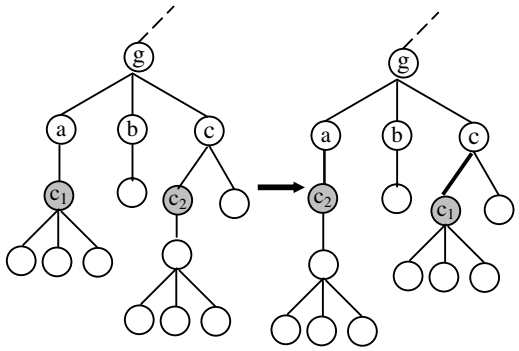

Fig. 6. An example of cousin node swap.

All the above three operations are initiated by the node with residual degree. In order to avoid operation collision and improve optimizing performance, we define the priority of these operations as follows: parent–child position swap, grandchild node promotion and nephew node movement. Each node tries to perform these operations sequentially.

4.3.4. Cousin node swap

Node c_1, c_2, who have different parents, are cousin nodes of each other, if they share the same grandfather. Cousin node swap operation exchanges the position of a pair of cousin nodes. The cousin node swap operation is performed if and only if the following inequation is true:

$$N_{c_1}(T) \times L_{c_1}(T) + N_{c_2}(T) \times L_{c_2}(T) > (L_a(T) + l(a, c_2)) \times N_{c_2}(T) + (L_c(T) + l(c, c_1)) \times N_{c_1}(T) \tag{7}$$

Since each node may have at most $\overline{d(T)} \times (\overline{d(T)} - 1)$ cousin nodes, the complexity of this operation is $\Theta(\overline{d(T)}^2)$. Note that although this operation can reduce the overall latency, it may come at the cost of sacrificing the aggregate subtree latency rooted at one of the cousin nodes. Fig. 6 illustrates an example of this operation. We can see that, before the swapping, $P_{c_1}(T) = a$, $P_{c_2}(T) = c$; After performing this operation, $P_{c_1}(T) = c$ and $P_{c_2}(T) = a$.

5. Group membership management

One of the main difference between ALM tree and IP multicast tree is that the overlay multicast network is dynamic due to the autonomy attribute of end-hosts. In this section, we present a membership management mechanism to support the join operation of new node and repair partitions caused by member leaving and/or failure.

5.1. Node join

When a new host wants to join a multicast session, it first contacts the source node s by sending a join request. The source node s in turn broadcasts the join request message to the whole multicast tree. In response to the request, all members with residual degrees return their overlay latency in the ALM tree to the new host. Then the new host computes its minimum ALM distance to s (as Definition 5) and attaches to the multicast tree through the determined access node.

5.2. Node leave

There are two different cases of node leave: friendly leaving and abrupt failure. In the former case, the leaving node notifies its neighbors before exiting. Abrupt failure is an exception caused by some unpredictable reasons and should be detected locally and propagated to the rest of the group. Many distributed failure detection

algorithms have been proposed in recent years [2,3]. In this paper, we are only concerned with the case of friendly leaving.

A leaf node should only inform its parent node to stop forwarding data packets to itself when leaving a multicast session. However, it is not as simple for no-leaf nodes. Once a non-leaf node leaves the system, the multicast tree is broken into several parts. To remerge these partitions, we employ a local repair scheme where only the children of the leaving node participate in the reconstruction and all the subtrees rooted at these children still keep their current topologies and states. Suppose node g is the leaving node, node p is the parent of g and C is a set made up of g's children. To repair the multicast tree, our algorithm first selects a node $c_1 \in C$ with the minimum distance from p to replace the position of g, and then updates the overlay latency of all the nodes in the subtree $S_{c_1}(T)$. Following that, we try to connect all the other subtrees $S_{c_i}(T)$ $(c_i \in C)$ to the existing component tree T through available degree within subtree as $S_{c_1}(T)$ possible. In the case of no sufficient degree, the unconnected subtrees will contact source node s so as to determine their access nodes.

6. Performance evaluation

In this section, we evaluate the performance of our heuristic routing algorithm. We generate a transit-stub network with 1000 network routers using the GT-ITM topology generator. End-hosts are randomly connected to routers of different stub domains. Our simulation scenarios consist of 200 nodes. The IP unicast delay between two nodes follows a uniform distribution with a finite interval [10 ms, 200 ms], i.e., $d(e) \sim Un$ (10 ms, 200 ms). The node degree is normally distributed with a mean of 5 and a standard deviation of 3 (the minimal and the maximal value are 1 and 20, respectively), i. e., $d_n(T) \sim N(5,3)$. Table 1 lists the default parameters of the performance evaluation.

Fig. 7 shows the performance comparison with different α. The mean latency corresponding to each α is the averaged value of several trails. Each trail is conducted with a different average node degree. We see that $\alpha = 0.4$ is preferable to either $\alpha = 1$ or $\alpha = 0$. This result proves that we could produce a better tree by giving more consideration to both transmission delay and node degree. When $\alpha = 1$, the heuristic DCMOLST algorithm is similar to the MDDBST algorithm [9] and gives priority to nodes with minimal latency from the source node. When $\alpha = 0$, the heuristic gives priority to nodes with larger available degree and chooses the one with minimal latency preferentially when they have the same available degrees.

Fig. 8 depicts the effect of iterative optimizing operations and compares the overall latency over different averaged node degree. Note that we suppose no node joins/exits the multicast tree in this experiment and nodes perform these optimizing operations periodically. From this figure, we can see that the overall latency decreases until converging to a stable level over time. Since our heuristics is not a greedy algorithm, the available degree near the root provides more opportunities to perform iterative optimizing operations after the initialization. Thus, initially the overall latency drops rapidly over optimizing operations. Meanwhile, these operations exhaust available degrees gradually, so the curve reaches to a converged state ultimately. This result indicates that the iterative optimizing operations can improve the overall performance effectively.From this figure we may observe that, higher average node degree could result in a lower overall latency ALM tree. The reason is twofold. First, large average node degree provides more feasibility to create a broad ALM tree through available degrees near the root. Therefore, the overlay latency from the source node to each node in the initial ALM tree is less than the case of small average node degree. Second, more node degree provides more possibility to further shrink the depth of multicast tree through residual degrees.

Table 1
Experiment parameters

Parameter	Value
$\|M\|$	250
α	0.4
$d(e)$	Un(10 ms, 200 ms)
$d_n(T)$	$N(5,3)$

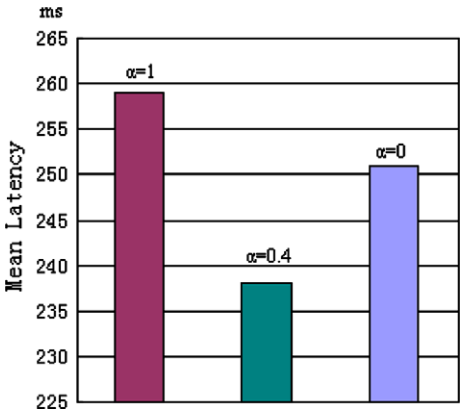

Fig. 7. Performance comparison of different weight ratio between edge delay and node degree, i.e., α.

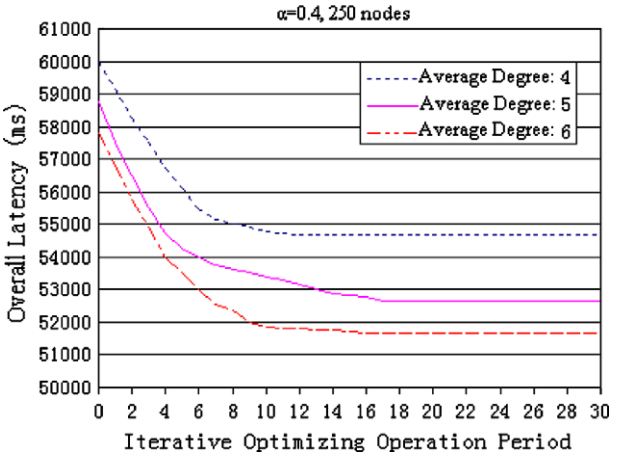

Fig. 8. Effect of different averaged node degree.

We also examine our solution under dynamic environment. To simulate the node join scenario, we first generate a multicast tree consisting of 120 nodes with DCMOLST algorithm. Then we periodically inject new nodes into the tree, inserting 30 nodes at one time. Fig. 9 illustrates the impact of node join. We can see that

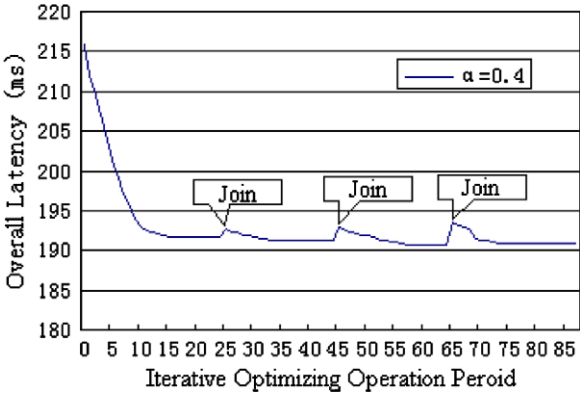

Fig. 9. Impact of node join.

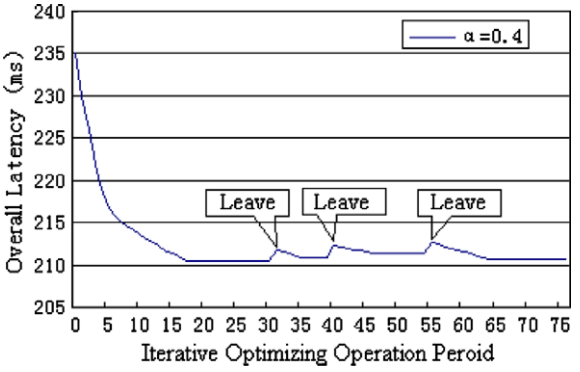

Fig. 10. Impact of node leave.

the average-latency increases immediately in response to join operations, and then gradually drops to a balanced state. Since most of the top level nodes near the root are saturated under steady state and new nodes can only attach to the lower nodes, the average-latency increases initially. Later, the optimizing operations work and improve the performance. Fig. 10 shows the impact of node exit. The initial multicast tree includes 250 nodes. A few nodes are randomly selected to leave the session friendly. Fig. 10 indicates that node leaving has negative impact on the overall performance, but this situation could be improved partially by the optimizing operations.

7. Conclusions

Different from previous work on QoS routing, we take a new angle of view for multicast tree optimizing problem in this paper. Especially, we study the degree-constrained minimum overall latency spanning tree problem with the aim to optimize the overall performance of ALM tree. Our solution divides the optimizing procedure into different stages and presents corresponding algorithms for them. Concretely, we provide a heuristic DCMOLST algorithm for multicast tree initialization and define a set of distributed iterative optimizing operations for further improvement. The philosophy of our heuristic DCMOLST algorithm is to avoid QoS degradation caused by single metrics through capturing node priority based on both edge delay and node degree simultaneously, while the idea of iterative local transformation operations is to improve the overall performance gains at the cost of sacrificing partial nodes' benefits. Simulation results demonstrate that the proposed degree-constrained QoS-aware routing algorithm is reasonable and promising for lower delay, higher performance application layer multicast services.

References

[1] E. Brosh, Y. Shavitt, Approximation and heuristic algorithms for minimum delay application-layer multicast trees, in: Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM'04), Hong Kong, China, 2004, pp. 2697–2707.
[2] M. Castro, P. Drushel, A-M. Kernarrec, A. Rowstron, Scribe: a large-scale and decentralized application-level multicast infrastructure, IEEE Journal on Selected Areas in Communications 20 (2002) 1489–1499.
[3] Y. Chu, S.G. Rao, S. Seshan, H. Zhang, A case for end system multicast, IEEE Journal on Selected Areas in Communications 20 (2002) 1456–1471.
[4] W.J. Jia, W.Q. Tu, J. Wu, Distributed hierarchical multicast tree algorithms for application mesh networks, IEICE Transaction on Information and Systems E89-D (2006) 654–662.
[5] X. Jia, A distributed algorithm of delay-bounded multicast routing for multimedia applications in wide area networks, IEEE/ACM Transaction on Networking 6 (1998) 828–837.
[6] V. Kompella, J. Pasquale, G. Polyzo, Multicast routing for multimedia communication, IEEE/ACM Transactions on Networking 1 (1993) 286–292.
[7] L.Y. Li, C.L. Li, A QoS multicast routing protocol for dynamic group topology, Information Sciences 169 (2005) 113–130.
[8] Z. Li, P. Mohapatra, QRON: QoS-aware routing in overlay networks, IEEE Journal on Selected Areas in Communications 22 (2004) 29–40.

[9] S.Y. Shi, J.S. Turner, Routing in overlay multicast networks, in: Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM'02), New York, NY, 2002, pp. 1200–1208.
[10] B. Suman, C. Kommareddy, K. Kar, et al., OMNI: An efficient overlay multicast infrastructure for real-time applications, Computer Networks 50 (2006) 826–841.
[11] B. Wang, J. Hou, Multicast routing and its QoS extension: problems, algorithms, and protocols, IEEE Network Magazine 14 (2000) 22–36.
[12] J. Wu, R.H. Hwang, J. Lu, Multicast routing with multiple QoS constraints in ATM networks, Information Sciences 124 (2000) 29–57.
[13] E.W. Zegura, K. Calvert, S. Bhattacharjee, How to model an internetwork, in: Proceedings of the 15th Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM'96), San Francisco, CA, 1996, pp. 594–602.
[14] Q. Zhu, M. Parsa, J. Garcia-Luna-Aceves, A source-based algorithm for delay-constrained minimal-cost multicasting, in: Proceedings of the 14th Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM'95), Boston, MA, 1995, pp. 377–384.