Computing and Informatics, Vol. 31, 2012, 1481–1510

A DISTANCE-HEURISTIC TREE BUILDING APPROACH IN APPLICATION LAYER MULTICAST

Xinchang Zhang, Weidong Gu, Meihong Yang

Shandong Key Laboratory of Computer Networks
Shandong Computer Science Center
No. 19, Keyuan Road, Lixia District
Jinan 250014, P.R. China
e-mail: {zhangxc, guwd, yangmh}@keylab.net

Guanggang Geng, Wanming Luo

Computer Network Information Center
Chinese Academy of Sciences
No. 4, South 4th Street, Zhongguancun
Beijing 100049, P.R. China
e-mail: {gengguanggang, luowanming}@cnnic.cn

Communicated by Ralf Klasing

Abstract. In the application layer multicast (ALM), clustering nearby nodes can effectively improve the multicast performance. However, it is difficult for the ALM solution to quickly and accurately position the newcomer, because group members have no direct knowledge of underlying network topology. Additionally, ALM delivery trees with different performances are built when group members join the group in different join sequences. To alleviate the above problems, this paper proposes a distance-heuristic tree building protocol (called DHTB). DHTB uses our proposed distance-constrained cluster model and close-member-first-receive (CF) rule. In the model, most nearby nodes are grouped into some distance-constrained clusters, with little cluster organization and maintenance overhead. The CF rule arranges or rearranges the locations of group members according to related distances, and effectively positions the newcomer with the help of on-demand landmarks. Both the distance-constrained cluster model and CF rule are distance-heuristic. Therefore DHTB can alleviate the join sequence problem, and build the ALM tree with desirable performance.

Keywords: Application layer multicast, multicast tree, join sequence, distance, multicast performance

Mathematics Subject Classification 2010: 94A11, 94C99

1 INTRODUCTION

In group applications (e.g., data dissemination and file sharing), multicast is the most efficient communication means because it can save significant bandwidth and greatly reduce the load on servers [1, 2, 3]. Originally, multicast functionality was envisioned at the IP layer. Servers would transmit to a group address and the routers themselves would take care of replicating and forwarding packets to downstream members of the multicast group. However, multicast routing represents a significant shift from traditional IP routing introducing numerous deployment and logistical hurdles for end-to-end multicast support. Despite the significant potential for multicast, numerous problems hindered its deployment enumerated in various publications such as [4].

As an alternative to IP Multicast, Application Layer Multicast (ALM) attempts to provide multicast-like functionality at the application level and end host rather than at the IP layer. Hence, ALM does not require modification to the network infrastructure allowing it to be deployed without end-to-end network support. However, the reliance on end hosts comes at several disadvantages that introduce numerous performance penalties. In a sense, ALM accelerates multicast deployment at the cost of acceptable performance penalties such as additional traffic load and latency.

Over the past decade, a wide variety of ALM approaches have emerged [5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]. For many of the works, clustering nearby nodes has been widely used to improve performance. Some ALM approaches (e.g., [6] and [7]) organize the overlay into a hierarchy of clusters. The hierarchy is usually organized by the bottom-top approach, i.e., cluster heads at the same level form next level clusters beginning from the lowest level, until there is only a single cluster at the highest level. In the above structure, the variations at some level might influence the construction of the higher levels, which means that cluster organization and maintenance overhead is high. Some agent-based ALM approaches (e.g., [9] and [10]) use some service domains to cluster nearby nodes. In these approaches, an agent can be considered as the center of the corresponding cluster (i.e., service domain). Agent-based ALM approaches need little cluster organization and maintenance overhead. However, providing agents fully distributed in the network is of high cost.

In most ALM protocols, the newcomer searches its location along the existing tree, and does not change the existing tree structure. Consequently, some join sequences of group members might result in generating the multicast tree with poor de-

livery performance. Some protocols (e.g., HMTP) alleviate the above join sequence problem through adjusting the existing tree in the improvement phase. However, the adjustment usually is of high cost because member hosts usually need traversing over a large number of tree branches to find the better locations.

In this paper, we propose a distance-constrained cluster model. In the proposed model, a cluster area is determined by the cluster center (i.e., cluster leader) and dynamic distance threshold, which means that the distance-constrained cluster need no cluster split and merge operations. If the newcomer is located in a cluster area and is positioned to the cluster in the join procedure, it can become a member of the cluster whenever it joins the group. Therefore the model can alleviate the join sequence problem. In addition, the variations of a distance-constrained cluster have little influence on other clusters, which greatly reduces the cluster maintenance overhead.

This paper also proposes a close-member-first-receive (CF) rule. Similar to some landmark-based positioning approaches (e.g., PIC [31]), the CF rule uses some landmarks to improve the accuracy of positioning the host. However, the landmarks in the CF rule are common member hosts instead of additional infrastructures. In the CF rule, the newcomer might make use of multiple landmarks before it joins the group, but it only employs one landmark at some moment. We use current landmark to denote the landmark being used. When some nodes compete for the child location of some existing node, the latter makes a decision in terms of the distances between nodes and their mutual current landmark, i.e., the node closest to the current landmark has priority to become a child of the existing node. Therefore the CF rule also alleviates the join sequence problem in some degree.

In this paper, we present a distance-heuristic ALM tree building solution (called DHTB) based on the distance-constrained cluster model and CF rule. DHTB constructs distance-constrained clusters by a self-organized way, and maintains the cluster with low cost. A DHTB multicast tree consists of a unique inter-cluster tree and multiple intra-cluster trees. In a given cluster, the CF rule might bring small gain because the nodes in the same cluster usually are close to each other. Consequently, the CF rule is only used in the inter-cluster tree building procedure.

The rest of the paper is organized as follows. In Section 2, we introduce the related work. Section 3 explains the two basic ideas of DHTB, i.e., the distance-constrained cluster model and CF rule. The protocol details of DHTB are presented in Section 4. We evaluate DHTB performance by analyzing the simulation results in Section 5. Finally, we conclude our work in Section 6.

2 RELATED WORK

The application layer multicast has been widely researched over the past decade (see [5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]). [30] gives a survey of application layer multicast protocols. In [19], application layer multicast protocols are classified into three different cate-

gories – mesh-first, tree-first and implicit approaches. The mesh-first approach (e.g., NARADA [13]) first organizes member hosts into the overlay mesh topology, then builds the multicast tree based the topology. In contrast, the tree-first protocol (e.g., HMTP [8]) directly constructs a data delivery tree. In the implicit approach (e.g., NICE [6]), the mesh and tree are simultaneously built.

Some existing protocols group member hosts into self-organized clusters, such as NICE and ZIGZAG [7]. NICE organizes the overlay into a hierarchy of clusters, and forms the multicast tree based on the hierarchy. The size of each cluster is between k and $3k-1$, which confines the scale of a cluster. If the size of a cluster is larger than $3k-1$, the cluster is divided into two equal-sized sub-clusters. In contrast, a cluster will be merged with a nearby cluster if the size of the cluster is under k. Each NICE cluster has a leader, which is the center of this cluster in the ideal situation. The cluster leaders at the same level form next level clusters initiating from the lowest level, until there is only a single cluster at the highest level. Similarly, ZIGZAG also organizes the overlay into a multi-layer hierarchy of clusters. The above two ALM solutions build the multi-layer hierarchy of clusters by the bottom-top approach, i.e., the cluster at some level (except the lowest level) is formed based on the clusters at lower level.

Some application layer multicast protocols employ designated agents to improve the multicast performance, e.g., [9], [10] and [22]. In OMNI, there exists a set of multicast service nodes (MSNs), each of which provides data distribution service to a set of member hosts. OMNI uses a decentralized scheme to organize the MSNs into an appropriate overlay structure. Similar to OMNI, TOMA uses proxies to form multicast service overlay network (called MSON). In addition, the hosts in a MSON domain also use centralized computation to build the corresponding delivery tree. Guo and Jha proposed an effective approach to address the issue of proxy placement in an overlay multicast network, i.e., decide an optimal placement of multicast proxies in the overlay multicast network to minimize the average end-to-end delay of the overlay skeleton tree [22].

A great many of ALM protocols (e.g., NICE, HMTP and Hostcast [14]) use distributed depth-first searching (DFS) approach to position the newcomer (see [20]). In the DFS approach, the newcomer searches down the multicast tree by exploring the branches of some existing nodes, as conventional depth-first traversal does. The protocol based on DFS employs some search criterion to select the appropriate branch in the traversal. For example, HMTP chooses the nearest node as next candidate parent (i.e., next node to contact).

There have been some positioning solutions based on network coordinates, e.g., PIC [31], Global Network Positioning [32] and binning scheme [33]. In these approaches, some designated nodes are chosen as landmarks to get network coordinates. With the help of landmarks, these solutions can accurately position hosts at the application layer. The main drawbacks of the approaches using public landmarks are as follows: (1) Choosing the ideal landmarks is difficult even impossible; (2) Public landmarks might become the bottleneck of ubiquitous deployment of the application layer multicast.

In the untrusted group environment, the selfishness problem directly influences the fairness and effectiveness of the application layer multicast. Laoutaris et al. expressed a node's "best response" wiring strategy as a k-media on asymmetric distance and proposed a neighbor selection model that can solve the selfishness problem to some extent (see [25]).

In the area of multicast (including IP multicast and ALM), reliable transmission is an important and relatively independent research subject (see [34]). For loss-tolerant applications (e.g., multimedia streaming), a small amount of loss is acceptable [26]. Banerjee et al. proposed a multicast data recovery scheme, called probabilistic resilient multicast (PRM), to improve the reliability of data delivery of ALM through sending some extra packets [21]. To provide reliable service based on the best-effort delivery of the ALM tree, ARQ (Automatic Repeat-reQuest) and FEC (Forward Error Correction) can be used to recover the data loss. [28] extensively analyzes epidemic loss recovery model with various topology, link noise, group size and message rate properties representing overlay network scenarios. Dán and Fodor revealed how and under what conditions overlays can benefit from the use of error control solutions, prioritization and taxation schemes [27]. In [29], a ARQ-based method (called LER) is presented to recover the found loss.

3 DISTANCE HEURISTICS IN DHTB

DHTB builds the ALM tree based on the distance heuristics, i.e., clusters nearby nodes according to the distances and builds the inter-cluster tree through the CF rule. This section will introduce the above two ideas, respectively.

3.1 Distance-Constrained Cluster Model

Similar to some previous works, such as ZIGZAG and NICE, DHTB divides the multicast group into many clusters. Each cluster in DHTB has a leader and n ($n \geq 0$) cluster members, as Figure 1 shows. In any cluster, the distance between each cluster member m and the leader is below a dynamic cluster threshold denoted by $\lambda(d_{max})$ (see Section 4.1). In this paper, the cluster in DHTB is called distance-constrained cluster. Different from the previous clustering approaches in ZIGZAG and NICE, DHTB (1) confines the cluster area by the distance metric instead of the number of members, (2) generates the cluster by a top-down approach, i.e., a member host is first selected as a cluster leader, then the cluster is determined in terms of the leader and cluster threshold, and (3) organizes the group into two-tier hierarchy, i.e., the whole group and distance-constrained clusters. The distance-constrained cluster model has the following advantages:

- The overhead of constructing and maintaining the clusters in the model is lower than that in ZIGZAG and NICE, because (1) distance-constrained clusters are only determined by cluster leaders and the cluster threshold, (2) there are no

cluster split and merge operations in the distance-constrained cluster model, and (3) the variation of some cluster has no influence on other clusters.

- If the newcomer is located in a cluster area and is positioned to the cluster in the join procedure, it can become a member of the cluster whenever it joins the group. In other words, the location at the cluster level is reserved in advance. Therefore the model can alleviate the join sequence problem.
- As noted previously, a member host becomes cluster leader if it cannot find an appropriate cluster to join. Otherwise, the host becomes the member of an existing cluster. Consequently, the cluster leader selection is easy in the model. Additionally, the leader is the center of the corresponding cluster in nature.

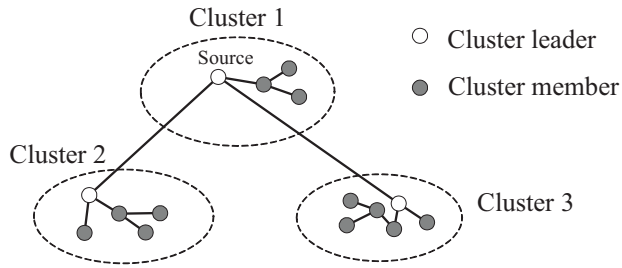

Fig. 1. The distance-constrained cluster model in DHTB

In some degree, cluster leaders act as the proxies used in the agent-based ALM protocols. However, the proxy-like leaders are automatically selected among group members, instead of being deployed in advance.

3.2 CF Rule

We propose a close-member-first-receive rule (called CF rule) to provide a distance-based heuristics for building the inter-cluster tree. In the heuristics, the reference point is also called landmark. A landmark is the multicast source or a common member host, and each member is able to become a landmark. The newcomer can employ $n(n \geq 1)$ landmarks before it joins the group, and the ith used landmark is denoted by i-landmark $(1 \leq i \leq n)$. Specially, the multicast source is the first landmark (1-landmark) for all the members. Additionally, different nodes might employ different sets of landmarks, depending on the existing nodes that they contact in the join procedures. Different from some existing landmark-based positioning approaches (e.g., PIC [31]), each host only uses one landmark at some phase. We use current landmark to denote the landmark that a host is using. When some hosts compete for the child location of some existing node, the latter makes a decision according to the distances between hosts and their mutual current landmark, i.e., the

host closest to the current landmark has priority to become a child of the existing node.

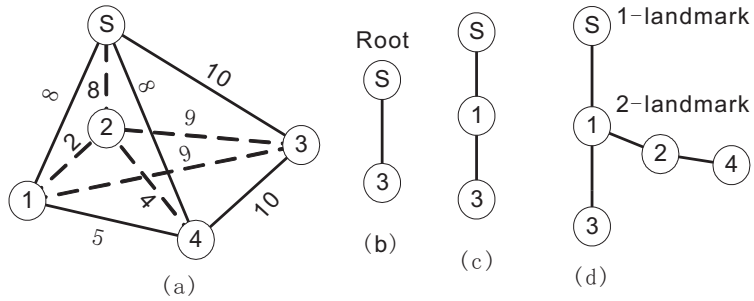

Fig. 2. An application example of the CF rule

We further explain the CF rule through an example shown in Figure 2. Figure 2 a) describes the distances of an overlay network. In the example, node 1 can accept at most two nodes as its children, and other nodes have at most one child. For simplicity, the example does not consider distance-constrained clusters. Assume that node 1 wants to join the group that has an existing tree shown in Figure 2 b); it first measures the distance to its first landmark – the root. Then node 1 sends join request message to the current landmark (the root), and competes the child location of node S with an existing node (i.e., node 3). Since node 1 is closer to the current landmark than node 3, it will become the unique child of node S, as Figure 2 c) shows. Note that node 3 becomes a child of 1. Similarly, nodes 2 and 4 also use the root as their first landmarks. However, they cannot replace node 1 because they each are not closer to S than node 1. Then nodes 2 and 4 go on the join procedure using node 1 as their 2-landmark. The final delivery tree is shown in Figure 2 d). We will explain how to switch the current landmark and combine the CF rule with distance-constrained cluster model in Section 4. In this part, we mainly care how the CF rule works at a certain phase.

The CF rule can effectively improve the multicast performance for the following reasons:

- The CF rule makes the positioning of the newcomer more accurate and adjusts the existing tree in terms of related distances, which can improve the consistency of the ALM tree and underlying network topology.
- With the help of the CF rule, the structure of built ALM tree heavily depends on the distances instead of the join order. Therefore the CF rule alleviates the join sequence problem in some degree.
- The CF rule pulls nodes around current landmark to the top of the subtree rooted by the landmark, and pushes nodes far from the landmark to the bottom of the subtree. The above operations can improve the ALM tree from overall point of view.

- In the CF rule, some common member hosts are automatically appointed as landmarks. Therefore the deployment of landmarks in the rule is flexible and convenient.

4 DHTB PROTOCOL DESCRIPTION

4.1 Overview

Similar to [8], we also assume that a rendezvous point (RP) is well-known to all members, and the RP knows the root of the delivery tree. Each host that intends to join the multicast group first contacts the RP for initiating the join process from the root. If a host can accept at most one host as its child, it will be pushed down to the bottom of the delivery tree. In this section, we do not discuss the details of the above procedure, and assume that each node can accept at least two hosts as its children.

DHTB employs the distance-constrained cluster model to group member hosts into many clusters. Let $d_{\max} = \max\{d(s,m)|m \in E\}$, where E denotes the set of the existing receivers of a multicast session, s means the multicast source, and $d(m,n)$ represents the distance from node m to node n. Then the dynamic cluster threshold in the model is denoted by $\lambda(d_{\max})$ and defined as

$$\lambda(d_{max}) = \tau \cdot \left\lfloor 1 + \mu \frac{d_{max}}{\tau} \right\rfloor \tag{1}$$

where τ and μ are two configuration parameters, which denote the minimum cluster threshold and adjustment coefficient, respectively. Similar to many existing ALM protocols, DHTB uses latency as the distance metric. Actually, the latency metric can effectively evaluate the physical distance between two hosts.

By Equation (1), the cluster threshold changes in terms of the different distributions of existing group members. Parameter τ is preferentially set to cover a local area (e.g., a common campus) because it is of little value for ALM to construct the cluster that covers smaller area. In the worst case, all the members will be in a unique cluster if group members are located in a local area. In the above case, DHTB can work but not try to optimize the multicast performance. Actually, it is not a wise choice to use ALM technology in the above situation. The value of parameter μ is between 0 and 1. The localization degree of the cluster decreases as the value of μ grows, and the number of clusters increases with smaller value of μ. Since each cluster leader is a potential landmark, the CF rule can provide better positioning capability as the number of clusters increases. However, too much clusters can reduce effectiveness of the distance-constrained cluster model. Therefore the selection of the value of the adjustment coefficient is a trade-off among the localization capability, positioning capability and effectiveness of the distance-constrained cluster model. Specifically, the value interval of μ such as $[0.2, 0.6]$ can be a reasonable choice. The experiment results in

Section 5.1 further verify the effectiveness of the above recommended value interval.

In DHTB, each member (including the multicast source) belongs to a certain distance-constrained cluster. Each cluster contains a cluster leader and n ($n > 0$) cluster members. The distance between each of cluster members and the leader is below the dynamic cluster threshold. The nodes in a cluster are further classified into three categories – cluster leader, cluster agent (CA for short) and common cluster member (CM for short).

Cluster leader: The CL node is the center of the corresponding cluster. The child of CL L is either a CA node of the cluster that L belongs to or another CL node.

Cluster agent: Any CA node (denoted by A) is a child of the cluster leader (denoted by L) of the cluster that A belongs to. The child of a CA node is either a CM node of the cluster that L belongs to or a CL node other than L.

Common cluster member: The parent node of any CM node (denoted by M) is either a CA or a CM, which is in the cluster that M belongs to.

Note that both CA and CM are called cluster members, and that a cluster might only have a cluster leader. Figure 3 illustrates the three types of nodes.

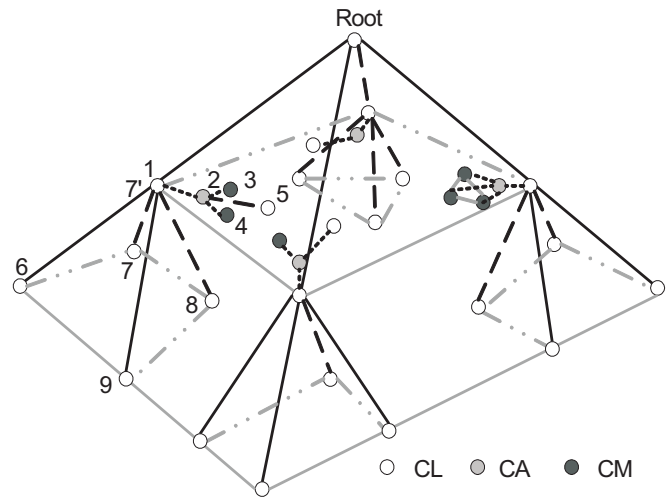

Fig. 3. A DHTB tree structure

The DHTB tree consists of two parts – an inter-cluster tree and multiple intra-cluster trees. The inter-cluster tree consists of all the cluster leaders (including the root) and related links, while an intra-cluster tree is the delivery tree within the corresponding cluster. Clearly, the union of the inter-cluster tree and all the intra-cluster trees forms the whole multicast tree. DHTB builds the inter-cluster

tree based on the CF rule, which effectively improves the multicast performance. Since the nodes in a cluster usually are relatively close to each other, the CF rule might bring no obvious gain. Therefore the CF rule is not used in the procedure of building the intra-cluster tree.

In the ALM tree, the fanout (or degree) of a node denotes the maximum number of children the node is willing to accommodate. The remaining fanout of node n is denoted by $f(n)$ and defined as

$$f(n) = \text{the fanout of } n - \text{the number of } n\text{'s existing children.} \tag{2}$$

In ALM, data replicating and forwarding are performed by member hosts. Since most member hosts tend to be on access links rather than at network core, it is necessary for an ALM solution to build a degree-constrained tree in terms of the bandwidth of members and the multicast application. Specially, the fanout of a member host usually is small in the streaming applications. DHTB is a degree-constrained ALM tree building approach, and the fanout of any node is constrained in terms of the node's bandwidth and the specific application.

As noted previously, providing reliable multicast service is a relatively independent research subject in the multicast area. Similar to most ALM protocols, DHTB is designed for building the delivery structure of the application layer multicast, and does not ensure reliable delivery services. In the application layer multicast, the loss is inevitable when a branch node leaves the multicast session without any notification. DHTB attempts to reduce the above loss during the rejoin procedure through building temporary connection. DHTB also uses the temporary connection to provide uninterrupted data delivery in the process of adjusting the tree structure. In DHTB, the data units sent by the source is sequentially numbered, which is necessary for the member host to sequentially deliver the received data from the buffer to the application. To provide reliable service, DHTB can be extended by adding additional loss recovery module. This paper does not discuss the details of the above extension.

4.2 Newcomer Join

As noted above, the DHTB tree consists of the inter-cluster tree and all the intra-cluster trees. The DHTB tree is progressively built as newcomers join the multicast group. In this section, we will introduce related messages, functions and algorithms.

4.2.1 Messages and Functions

We first introduce some basic concepts used in the following part. If a CL (denoted by L) has a CL child that uses L as the last landmark, the CL child is marked with symbol $*$, denoted by CL* (see Algorithm 1). From Algorithm 1, we can notice that the CL* child occupies its location with low priority. Note that symbol $*$ only appears in the children list of a CL node. In the following parts, the CL child is

distinguishable from the CL* child. We use e, $e*$ and i symbols to represent the CL, CL* and no-CL child types. According to the proposed cluster model, each no-CL child of a CL node is a CA node, and each no-CL child of a CA (or CM) node is a CM node.

In this paper, $lm(n)$ denotes the current landmark of node n if n is joining the group, or the last used landmark if n is an existing node. Additionally, $d_{lm}(n)$ means the distance from $lm(n)$ to n, $C_t(p)$ means the set of $p's$ children of type t ($t = e, e*$ or i), and $C_{t_1,t_2}(n)$ represents the union of $C_{t_1}(n)$ and $C_{t_2}(n)$.

The main types of the messages used in the join procedure are as follows:

- *join*$(p, lm(n), d_{lm}(n))$: Newcomer n sends the message, with $lm(n)$ and $d_{lm}(n)$, to its current candidate parent p. Note that only cluster leaders can receive the message.
- *child*(n, t): The current candidate parent sends the message, with its children list of type t, to newcomer n. The message is used to tell n to choose next candidate parent among the nodes in the children list.
- *redirect*(n, p): When node p finds that it should become the new current landmark of newcomer n, it sends the message to tell n to change the current landmark.
- *agent*$(n, C_i(p))$: Node p sends the message to tell node n to select the closest node among $p's$ no-CL children as its next candidate parent.
- *joinc*(h, n): Newcomer n sends the message to request h to accept itself as a CA child.
- *success*(n, h): h sends the message to tell n to end the join procedure because it has been accepted as a child.
- *clusterchild*(n, h): h sends the message to notify n of its CA children list.

Next we define the following four functions, which are used in the join algorithm of DHTB.

$$t(p,n) = \begin{cases} \lambda(d_{max}) & \text{if } lm(n) = p \\ \lambda(d_{max}) + d_{lm}(p) & \text{otherwise} \end{cases}. \tag{3}$$

$$f_i(n) = \begin{cases} f(n) & \text{if } f(n) > 0 \\ 0 & \text{if } f(n) = 0,\ |C_i(n)| \neq 0 \\ 1 & \text{if } f(n) = 0,\ |C_i(n)| = 0 \end{cases}. \tag{4}$$

$$f_e(n) = \begin{cases} f(n) & \text{if } f(n) > 0 \\ |C_{i,e}(n)| - 1 & \text{if } f(n) = 0 \end{cases}. \tag{5}$$

$$f_{e*}(n) = \begin{cases} f(n) & \text{if } f(n) > 0 \\ |C_i(n)| - 1 & \text{if } f(n) = 0,\ |C_i(n)| > 1 \\ 1 & \text{if } f(n) + |C_{i,e*}(n)| = 0 \\ 0 & \text{otherwise} \end{cases}. \tag{6}$$

4.2.2 Join Algorithm

Algorithm 1 describes the join procedure. In the algorithm, G is the identification of the multicast group, n means the newcomer, and s indicates the tree root (i.e., multicast source). In DHTB, the tree root saves the current cluster threshold. From Line 2 of the algorithm, we can notice that the newcomer contacts the tree root and measures the distance from the root. In the *join* message sent to the root s, $d_{lm}(n) = d(s, n)$. Therefore the root can adjust the current cluster threshold in terms of Equation (1). The root notifies the newcomer of the current cluster threshold, and the following *join* message will carry the threshold to tell the receiver of the message to use the new threshold. For the sake of simplicity, Algorithm 1 does not depict the above procedure.

Algorithm 1 uses the following two operations on the linked list (denoted by L):

- SortInsert$(L, v, d(v, n)$: Node v is inserted into L, denoted by $(a_1, \cdots, a_{i-1}, v, a_i, \cdots, a_m)$, such that $d(a_j, n) \leq d(a_k, n)$ $(1 \leq i, k \leq m, j \neq k)$.
- GetHead(L): This operation gets and deletes the first node k of L. Note that k is the closest node (to the newcomer) among the nodes in L.

Algorithm 2 explains the Accept procedure used in Algorithm 1. Once newcomer n is positioned to the cluster that it belongs to, it initiates the cluster join procedure (i.e., ClusterJoin in Algorithm 1) to become a CM or CA of the cluster, beginning from the corresponding cluster leader h. Algorithm 3 describes the ClusterJoin procedure. Next we will introduce the Replace procedure.

If the push-down condition holds, the Replace procedure (also called push-down procedure) is performed. The push-down condition is denoted by $P(p, n, t)$ and defined as

$$P(p, n, t) = (f(n) \geq \gamma) \wedge \exists j((j \in C_t(p)) \wedge (d_{lm}(n) \leq d_{lm}(j) - \lambda(d_{\max}))), \qquad (7)$$

where γ $(\gamma \geq 2)$ is a configuration parameter, t represents the type of a child. We use function $r(p, t)$ to denote p's child of type t that should be replaced. Let $r(p, t) = k$, then k satisfies the following Boolean expression:

$$k \in C_t(p) \wedge (d_{lm}(k) = \max\{d_{lm}(j) | j \in C_t(p)\}). \qquad (8)$$

The following are the basic steps of Replace(p, n, t, c):

Step 1: n adds $r(p, t)$ to its children list, and labels $r(p, t)$ by denotation T.

Step 2: n replaces $r(p, t)$ in $p's$ children list, and becomes $p's$ child of type c.

Step 3: $r(p, t)$ rejoins the group beginning from p.

Step 4: If $r(p, t)$ receives repeated data unit from n and its new parent node, it sends a request to tell n to delete $r(p, t)$ in the children list.

In the above procedure, the child with denotation T is a temporary child. In the following tree building procedure, the temporary children are ignored. Under

Algorithm 1 Join algorithm

procedure GROUPJOIN(G, s, n)
  initialize: $lm(n) \leftarrow s; d_{lm}(n) \leftarrow d(s,n); p \leftarrow s$; create a linked list L.
  send $join(p, lm(n), d_{lm}(n))$ to p.
  if n receives $redirect(n,p)$ **then**
    $lm(n) \leftarrow p$; $d_{lm}(n) \leftarrow d(p,n)$; initialize L.
    send $join(p, lm(n), d_{lm}(n))$ to p.
  end if
  if n receives $child(n, C_t(p))$ **then**
    for all $v \in C_t(p)$ **do**
      measure the distance $d(v,n)$; SortInsert$(L, v, d(v,n))$.
    end for
    $k \leftarrow$ GetHead(L); send $join(k, lm(n), d_{lm}(n))$ to p.
  end if
  If n receives $success(n,p)$, then n joins the group successfully.
end procedure
procedure RESPONSE(p, n) // p is an existing CL node
  if p receives $join(p, lm(n), d_{lm}(n))$ **then**
    if $d_{lm}(n) < t(p,n)$ **then**
      If $lm = p$, then ClusterJoin(p,n). Otherwise, send $redirect(n,p)$ to n.
    else if $lm = p$ and $lm \neq s$ **then**
      if $f_{e*}(n) > 0$ **then**
        Accept$(n, m, e*)$
      else if $|C_{e*}(p)| \neq 0$ **then**
        If $P(p,n,e*) = 1$ holds, then run the $Replace(p,n,e*,e*)$ procedure. Otherwise, send $child(n, C_{e*}(p))$ to n.
      else // $f_{e*}(n) = 0$
        send $agent(n, C_i(p))$ to n.
      end if
    else
      if $f_e(p) > 0$ **then**
        Accept(p,n,e)
      else if $P(p,n,e) = 1$ **then**
        Replace(p,n,e,e) // p accepts n as a CL child
      else
        send $child(n, C_e(p)$ to n.
      end if
    end if
  end if
end procedure

the premise that a node successfully forwards the received packet to its normal children (i.e., CL, CL* and no-CL children), the node also forwards the received packet to its temporary children. In general, the new member n has no child or only has few children before $r(p,t)$ finds its new parent node. Therefore the above steps effectively solve the data loss caused by the structure adjustment. When CL k rejoins the group, it sends the *join* message to previous parent with $d_{lm}(k)$, while no-CL m sends the *joinc* message to $lm(m)$. Then the rejoin procedure goes on according to Algorithm 3 or Algorithm 1. In DHTB, the data units sent by the source are sequentially numbered (see 4.1). Therefore the repeated data can be detected in terms of the sequence number of the data unit. Note that the repeated data is not forwarded to downstream nodes.

Algorithm 2 m accepts n as a child of type t

procedure ACCEPT(m,n,t) // $f_t(m)>0, t\in\{e,e*,i\}$
  if $f(m)>0$ **then**
    m accepts n as its child of type t.
  else if $t=i$ **then** // m receives $joinc(m,n)$
    If $|C_{e^*}(m)|>0$, then Replace$(m,n,e*,CA)$. Otherwise, Replace(m,n,e,CA).
  else if $t=e$ **then** // m is not the current landmark of n
    If $|C_i(m)|>1$, then Replace(m,n,i,CL). Otherwise, Replace$(m,n,e*,CL)$. // $m=lm(k)$
  else // m is current landmark of n
    If $|C_i(m)|>1$, then Replace$(m,n,i,CL*)$. Otherwise, Replace$(m,n,e,CL*)$.
  end if
end procedure

Algorithm 3 Join the cluster

procedure CLUSTERJOIN(h,n)
  n sends $joinc(h,n)$ to h;
  If n receives $clusterchild(n,C_i(h))$, then the join process goes on as HMTP.
  If n receives $success(n,h)$, then n joins the cluster successfully.
end procedure
procedure RESPONSE(h,n)
  if h receives $joinc(h,n)$ **then**
    If $f_i(h)>0$, Accept(h,n,i). Otherwise, send $clusterchild(n,C_i(h))$ to n.
  end if
end procedure

From Equations (4)–(6) and Algorithm 2, we can notice that the cluster leader rearranges the children of different types as the following two rules:

- If there are multiple nodes (including the CL node) in a given cluster area, the CL node must provide at least one child location to the cluster member.
- The CL node accepts as many CL children as possible, which can produce more potential landmarks for the following newcomers.

When two nearby nodes send *join* messages to some existing node, the existing node might accept the two newcomers as its CL (or CL*) children, respectively. Consequently, each newcomer needs to determine whether or not its parent has some CL (or CL*) child which is close enough for the newcomer to join as a cluster member. If its parent has such a child, the newcomer joins the corresponding cluster.

Figure 3 depicts an example of building a DHTB tree. When node 3 wants to join the group, it contacts the first landmark – the root (denoted by s), and gets the distance $d(s,3)$. The root has accepted four nodes as its children (up to maximum accepting capacity), and the distance $d(s,3)$ is not short enough for node 3 to replace anyone of its existing children. Therefore node 3 has to select the closest node (node 1) as its next candidate parent. Node 1 finds that $d(s,3)$ is approximate to $d(s,1)$, then it tells the newcomer to use it as the current landmark. Finally, provided that $d(1,3) < \lambda(d_{max})$, node 3 becomes a CM of the corresponding cluster. Suppose that nodes 1–5 have not joined the group, and that the original location of node 7 is marked by $7'$. When node 1 wants to join the group, it first contacts the root, and gets the distance $d(s,1)$. According to Algorithm 1, node 1 occupies the location of $7'$ (i.e., 7 is replaced by 1). Then node 7 rejoins the group beginning from node s, and becomes a CL child of node 1.

We use symbol 1-CL to mean the CL that only uses the root as its landmark, and s to denote the root of the multicast tree. Let A_k denote the set of existing 1-CLs before k joins the group and $d_k^m = max\{d(s,j)|j \in A_k \bigcup \{k\}\}$, we get the following theorem.

Theorem 1. Assume that 1-CL k joins the group in terms of current cluster threshold λ, then k is at level $O(\lfloor \frac{d(s,k)}{\lambda} \rfloor)$, and at level $\lfloor \frac{d_k^m}{\lambda} \rfloor$ in the worst case.

Proof. Suppose that node k passes $s, n_1, n_2, \cdots, n_m$. Then

$$\begin{aligned} d(s,k) \;\geq\; & d(s,n_1) + (d(s,n_2) - d(s,n_1)) + \cdots \\ & + (d(s,n_m) - d(s,n_{m-1})) + (d(s,n_k) - d(s,n_m)). \end{aligned}$$

When $d(s,n_i) - d(s,n_{i-1}) = \lambda$, we can get the max m which satisfies the above inequality. Note that the level of the root is 0. The theorem has been proven. □

4.3 Improvement and Maintenance

DHTB can generate a delivery tree with desirable performance in the initial tree building stage. However, the initial DHTB tree is not optimal (actually the optimization problem is NP-hard, as [35] proves). Consequently, we use the new parent switch procedure to further improve the multicast performance.

In the improvement procedure, a node (denoted by i) chooses a random node (denoted by k) in its root path, and asks node k for $lm(k)$. If $k's$ current landmark is not the same as $i's$, node i contacts $lm(k)$ to get the distance $d(lm(k), i)$. Then node i rejoins the group beginning from $lm(k)$. The above rejoin procedure is similar to Algorithm 1, except that next candidate parent is randomly selected with probability α $(0 < \alpha < 1)$, and is selected in terms of Algorithm 1 with probability $(1-\alpha)$. The transition from old parent p to new parent p' is performed if $d(p, i) \geq d(p', i)+\varphi$, where φ is the cost of the transition. After i becomes a member of the group, its root path might change because the multicast group is dynamic. Consequently, DHTB selects next candidate parent in terms of Algorithm 1 with probability $(1 - \alpha)$, which is beneficial for the DHTB tree for fast converging and keeping the desirable structure.

According to Algorithm 1 and the new parent switch procedure, we have the following theorems.

Theorem 2. Let φ and the cluster threshold each be equal to zero; then the DHTB trees of a designed group tend to the same structure in different join sequences.

Proof. Suppose that $T(V, E)$ is a steady DHTB tree; we consider another tree $T(V, E')$ built in some join sequence, where V denotes the set of member nodes, E and E' denote the sets of tree edges of $T(V, E)$ and $T(V, E')$, respectively. Let f_S mean the fanout of the root (i.e., S), then the closest f_S nodes to the root become $S's$ children in any join sequence. Thus $T(V, E')$ is consistent with $T(V, E)$ at level 1. Suppose that $T(V, E')$ is consistent with $T(V, E)$ at level i. Next we prove that the structures at level $(i + 1)$ of $T(V, E)$ and $T(V, E')$ are consistent with each other. According to the DHTB solution, node c will finally choose node k as its parent if k is at level i in $T(V, E)$ and has a child whose parent is not k in $T(V, E')$. The theorem has been proven. □

Since φ and the cluster threshold each are larger than zero, DHTB trees in different join sequences are somewhat different. However, the stable DHTB trees of a designated group have similar gross performance in different join sequences.

When a member (denoted by m) leaves a group gracefully, it notifies its parent and children of its departure. After receiving the above notification, m's parent simply deletes the node from its children list, and m's children have to find new parents. For rejoining the group, a child (denoted by n) of m first finds the closest active node (denoted by j) in its root path. Then the child requests the active node (denoted by p) for its current landmark $lm(p)$. Assume that the n's current landmark is $lm(p)$; then n rejoins the group initiating from j by cluster join algorithm if it is a CM or CA node, or rejoins the group as Algorithm 1 does if it is a CL node. If n's current landmark is not lm, it rejoins the group initiating from $lm(j)$ as Algorithm 1 does. Note that the above procedure does not break the CF rule. For reducing the data loss, n requests an active candidate parent node, whose remaining fanout is larger than zero, to accept it as a temporary child in the above rejoin procedure. The details on adding and deleting the temporary child can be seen in Section 4.2.2.

As the group size grows, the root endures more and more stress. To address the above problem, the root periodically multicasts a special message that includes a list of its CL children. When a member receives the message, it saves or updates the corresponding list. Therefore the newcomer can get the list from anyone of existing nodes. The newcomer randomly chooses a node in the list, and joins the group starting from the node.

As to the other maintenance procedures (e.g., partition recovery, loop detection and resolution), DHTB can use the approaches as some tree-first protocols do. We ignore more details in this paper.

5 SIMULATION RESULTS

We used the GT-ITM Generator [36] to generate transit-stub graphs as underlying network topologies. Different topologies were generated with different parameters, and the number of nodes in each topology was between 4 500 and 5 200. For each topology, we also generated 1 000 nodes as member hosts and a node as the server. Each host node was connected to a random stub-domain node in the corresponding topology, and each stub-domain node could connect at most one host node. Unless otherwise specified, the fanout of a host node was between 2 and 5. In our experiments, we used latency as the distance metrics. The fixed parameter configurations were as follows: $\tau = 60\,\text{ms}$; $\varphi = 200\,\text{ms}$; $\gamma = 2$. The value of τ is slightly larger than all the average diameter of the stub-domains. Note that the diameter of a stub-domain denotes the maximum node-to-node delay in the domain. By default, $\mu = 0.25$. We used the NS-2 simulator ([37]) to simulate DHTB and three existing ALM protocols – HMTP, OMNI and ZIGZAG. Except for some results from a single run (Figures 9, 12, 13 and 14), data points in the following graphs represent averages over 100 runs with 95% confidence interval. Note that the join orders in different runs of an ALM protocol were different, and different ALM protocols used the same join orders in the same scenario.

5.1 Distance-Constrained Clusters

The practical performance of OMNI is closely related to the deployment of MSNs. In this part, we randomly selected a member host, among the member hosts in each stub-domain, as the MSN that provides service to the member hosts in the stub-domain. OMNI with the above optimized deployment of MSNs (OMNI-OD for short) can well cluster nearby hosts because each local area (i.e., stub domain) has a service agent. Similar to OMNI, DHTB also can explicitly cluster member hosts. However, DHTB builds the clusters self-organizedly. In our experiments, we investigated the self-organized clustering capability of DHTB using OMNI-OD as reference. Note that each multicast group contains 1 000 receivers in this part.

Let u and d be two members in the same stub-domain (denoted by D), then d is said to be a continuous downstream node of u in an ALM tree if (1) d is

a downstream node of u in the ALM tree and (2) each node in the path from u to d is in D. Clearly, u and its continuous downstream nodes can be thought of as being practically clustered. We use M_D to represent the set of the members in D, and use $h(M_D)$ to denote the member that is in M_D and has the most continuous downstream nodes. Assume that member m is in M_D; we say that m is not clustered as OMNI-OD if m is not a continuous downstream node of $h(M_D)$. In this paper, we use DHTB/OMNI-OD clustering failure ratio to denote the ratio of the members that are not clustered as OMNI-OD to the total number of the members in the multicast group. Similarly, the DHTB/OMNI-OD cluster ratio means the ratio of the number of clusters of DHTB to that of OMNI-OD.

Figures 4 and 5 illustrate the DHTB/OMNI-OD clustering failure ratios and cluster ratios with different values of the adjustment coefficient. In DHTB, the distance-constrained cluster covers wider area with a larger value of μ. In other words, a member in M_D is not a continuous downstream node of $h(M_D)$ with higher probability as the value of μ increases. Therefore the DHTB/OMNI-OD clustering failure ratio grows as the value of μ increases (see Figure 4). For the same reason, less clusters are generated as the value of μ increases (see Figure 5). As noted previously, the selection of the value of the adjustment coefficient is a trade-off among the localization capability, positioning capability and effectiveness of the distance-constrained cluster model. Figures 4 and 5 show that $[0.25, 0.6]$ should be the desirable value range of the adjustment coefficient in our experiments. We further investigated the DHTB/OMNI-OD clustering failure ratio with fixed value (0.25) of the adjustment coefficient under eight topologies. The results show that the DHTB/OMNI-OD clustering failure ratio slightly changed under different topologies (see Figure 6).

5.2 Tree Cost, Stress and Delay

This section will present some performance (tree cost, stress and delay) comparisons of DHTB and HMTP. In these experiments, all the ALM trees were built under the same topology.

5.2.1 Tree Cost

In this part, we use tree cost ratio (the ratio of the cost of ALM tree to that of SPST) to evaluate the tree cost. Note that SPST denotes the shortest path source tree. From Figure 7, we can see that the tree cost ratios of DHTB and HMTP each slowly increases as the group size grows. Figure 7 also shows that the tree cost ratio of HMTP monotonously increase as the group size grows. In contrast, the tree cost ratio of DHTB sometimes drops, mainly because more and more nearby nodes are clustered as more hosts join the group. On the whole, DHTB can build ALM trees with lower tree cost than HMTP, because (1) the CF rule and landmark-based positioning make DHTB tree conform to the underlying network topology to some extent, and (2) the member hosts in a local area can be well grouped into a cluster.

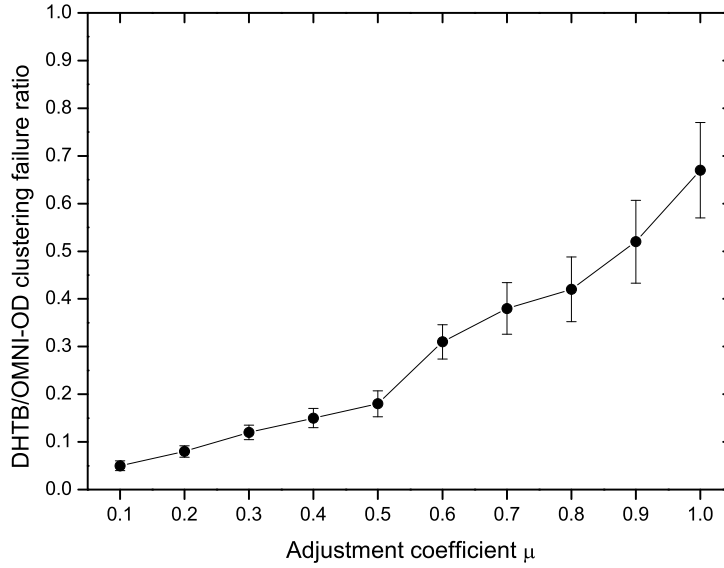

Fig. 4. DHTB/OMNI-OD clustering failure ratios with different values of μ

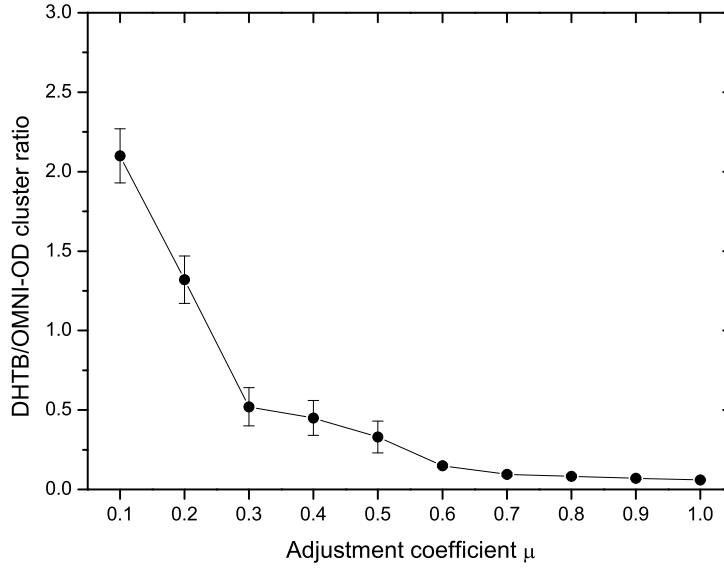

Fig. 5. DHTB/OMNI-OD cluster ratios with different values of μ

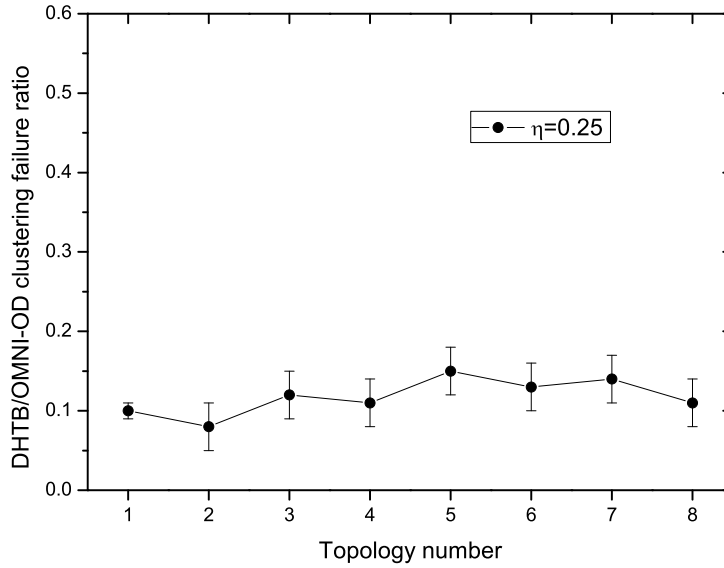

Fig. 6. DHTB/OMNI-OD clustering failure ratios under different topologies

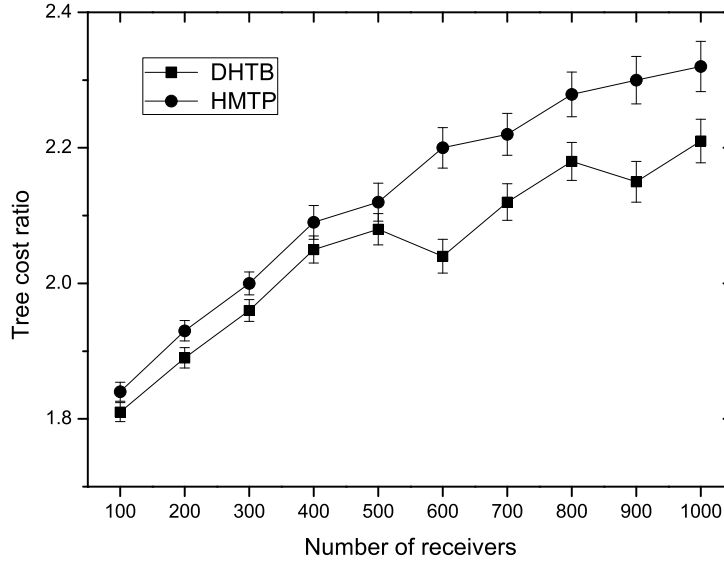

Fig. 7. Tree cost comparison of DHTB and HMTP

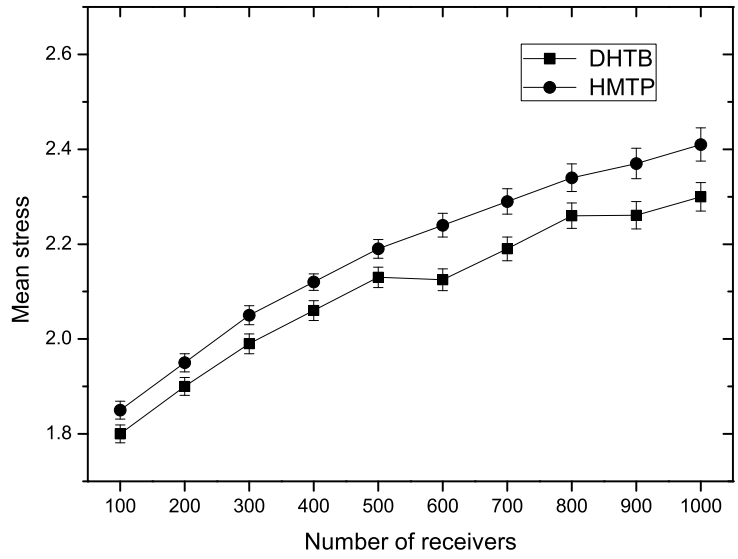

Fig. 8. Link load comparison of DHTB and HMTP

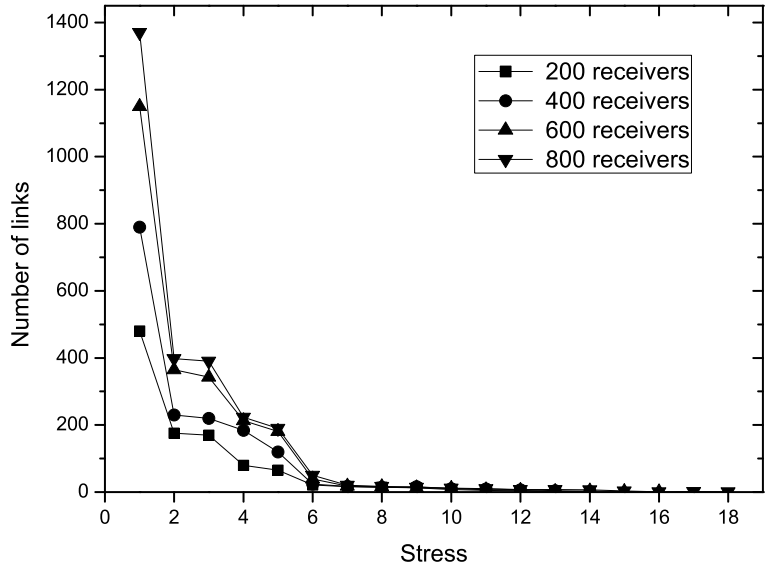

Fig. 9. Stress distribution of DHTB

5.2.2 Stress

The stress metric is defined per-link and counts the number of identical packets sent by a protocol over each underlying link in the network [6]. Figure 8 gives the stress comparison of DHTB and HMTP. From the figure, we can observe that the mean stress slowly increases as the group size grows in both DHTB and HMTP, which is advantageous to keep the scalability of multicast applications. We also can see that DHTB brings less stress to the links than HMTP. We attribute the above advantage to the better topology-awareness of DHTB.

Figure 9 depicts the stress distributions of DHTB in four different groups. The horizontal axis represents stress, and the vertical axis indicates the number of physical links with a given stress. From the figure, we can notice that each distribution curve has a heavier tail, and most links have low stress. In this paper we do not present the stress distributions of HMTP, which are similar to the corresponding distributions of DHTB.

5.2.3 Delay

Figure 10 plots the mean delay ratios in DHTB and HMTP, as a function of the group size. A member's delay ratio means the ratio of the latency from the root to the member along the ALM tree to that along SPST under the same topology. From Figure 10, we can observe that the mean delay ratio tends to increase with larger group size in both DHTB and HMTP. However, the mean delay ratio in DHTB is lower than that in HMTP, and sometimes drops at some group sizes. The main reasons of the above dropping are as follows: When more hosts (including some nodes close to the root) joined the group, the nodes with high delay ratios left their current parents, and found closer nodes as their new parents. In these experiments, the minimum delay ratios in DHTB and HMTP each were equal to 1 in each group, and the corresponding amount of variance of members' delay ratios in each group can be seen in Figure 11.

5.3 Convergence

Both DHTB and HMTP explicitly improve initial ALM trees. In this part, we give the comparison of convergences of DHTB and HMTP. In these simulations, we first built initial DHTB and HMTP trees with the same 1 000 receivers. Then 100 random member hosts launched the corresponding improvement procedure to improve DHTB and HMTP trees. In this paper, the above improvement phase is called improving unit. In our experiments, initial DHTB and HMTP trees were improved by 50 improving units. Figures 12–14 show related results. From the figures, we can notice that the convergence spend of DHTB is higher than that of HMTP. We attribute the above advantage of DHTB to the proposed distance-constrained cluster model and CF rule.

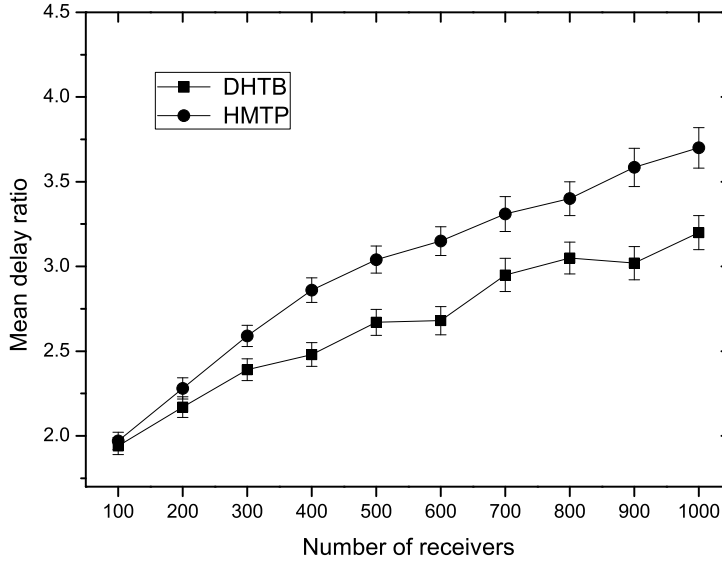

Fig. 10. Delay comparison of DHTB and HMTP

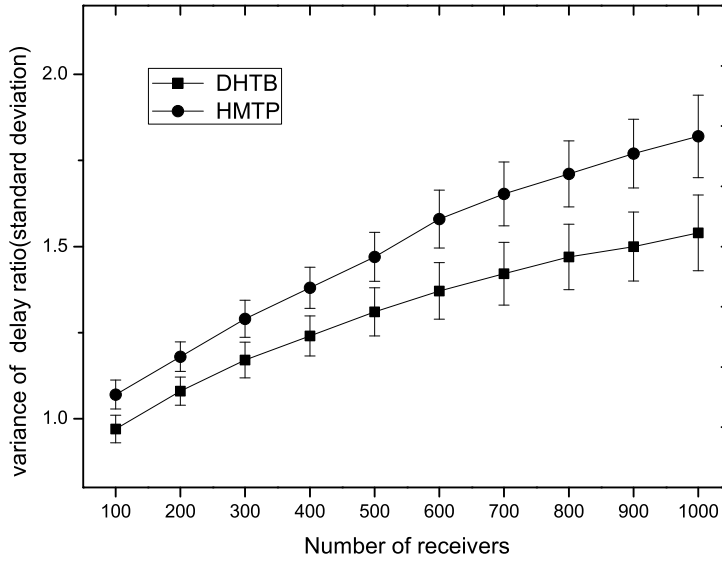

Fig. 11. Variance comparison of members' delay ratios of DHTB and HMTP

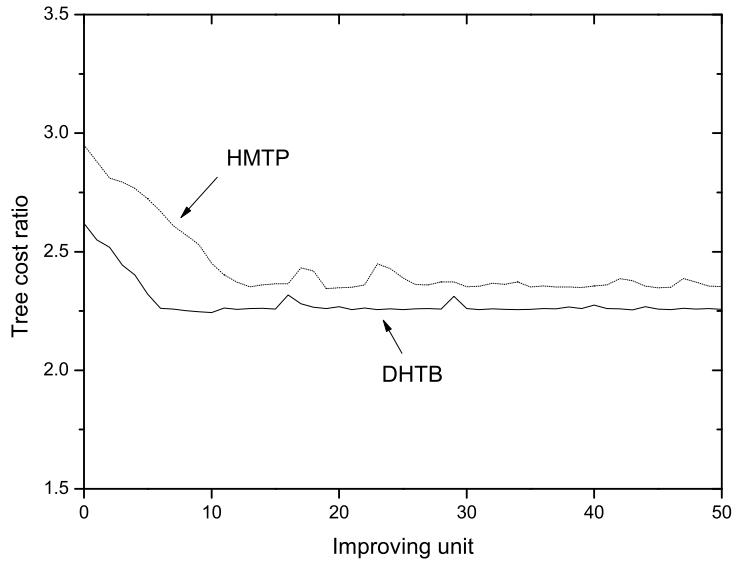

Fig. 12. Tree cost improvements of DHTB and HMTP

5.4 Stability

In these experiments, 1 000 designed receivers joined the multicast session in 300 different join sequences in terms of DHTB, HMTP and ZIGZAG. Note that these experiments were based on a unique topology. In the experiments on ZIGZAG, parameter k ($k > 3$) is assigned by 4. The worst-case fanout of a node in ZIGZAG heavily depends on the overlay structure. Specially, the worst-case fanout of the associate-head or head of the ZIGZAG tree is at most $6k - 3$ ($k > 3$), and that of the common node (neither the head nor the associate-head) is at most $3k - 1$. Consequently, the fanout of each host node in the experiments on ZIGZAG was only confined by the upper bound 21. The main goal of these experiments was to compare the stabilities of the above three ALM protocols when the members join the group in different join orders. Table 1 gives the statistical results of these experiments. Through the distance-constrained cluster model and CF rule, DHTB attempts to make the delivery tree conform to the underlying topology, which alleviates the join sequence problem. Therefore the tree cost ratio, mean stress and mean delay ratio of DHTB each are steadier than that of HMTP and ZIGZAG (see Table 1).

6 CONCLUSIONS

In this paper, we presented a distance-heuristic tree building protocol (called DHTB) based on the proposed distance-constrained cluster model and CF rule. The clusters in the cluster model need (1) little cluster construction and maintenance overhead

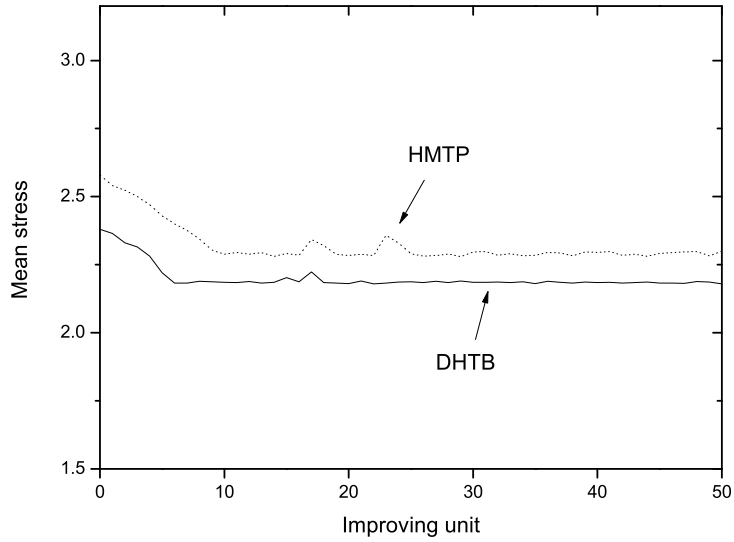

Fig. 13. Stress improvements of DHTB and HMTP

Method	Tree cost			Mean stress			Mean delay ratio		
	Min	Max	s	Min	Max	s	Min	Max	s
DHTB	2.11	2.35	0.069	2.08	2.29	0.057	2.84	3.8	0.17
HMTP	2.15	2.53	0.081	2.13	2.44	0.063	2.94	4.2	0.23
ZIGZAG	2.13	2.59	0.102	2.19	2.53	0.079	2.81	4.1	0.21

Table 1. Stability comparison of DHTB, HMTP and ZIGZAG (s denotes standard deviation)

and (2) no cluster split and merge operation. In the CF rule, the landmark-based positioning approach improves the accuracy of positioning the newcomer in the join procedure. Additionally, the CF rule arranges (or rearranges) the members' locations in the ALM tree by a distance-based approach, which can improve the ALM tree from overall point of view. Since DHTB builds the ALM tree heavily depending on the distance heuristics, the join sequence problem can be effectively alleviated. The theoretical and experimental results show that DHTB can build the ALM tree with desirable multicast performance, and build ALM trees with similar gross performance in different join sequences.

Acknowledgments

This work was supported by the National Basic Research (973) Program of China under Contract No. 2009CB320502, the National Science Foundation of China under

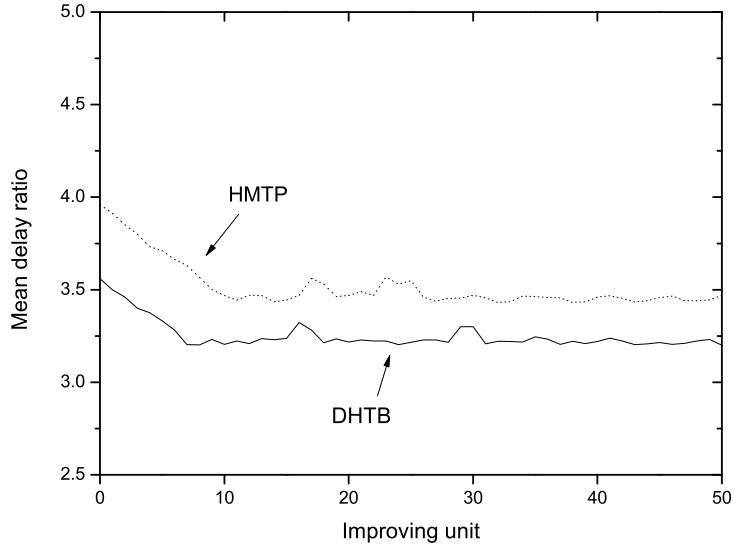

Fig. 14. Delay improvements of DHTB and HMTP

Contract No. 61272433 and 61005029, and the Outstanding Young and Middle-aged Scholars Foundation of Shandong Province of China under Contract No. BS2011DX-033. The authors would like to thank the Executive Editor and anonymous referees for their constructive comments on the revisions of this paper.

REFERENCES

[1] Paul, P.—Raghavan, S. V.: Survey of Multicast Routing Algorithms and Protocols. In Proc. of the 15th international conference on Computer communication, Washington, 2002, pp. 902–926.

[2] Obraczka, K.: Multicast Transport Protocols: A Survey and Taxonomy. IEEE Communications Magazine, Vol. 36, 1998, No. 1, pp. 94–102.

[3] Luo, J.—Ye, D.—Liu, X.—Fan, M.: A Survey of Multicast Routing Protocols for Mobile Ad-Hoc Networks. IEEE Communications Surveys & Tutor, Vol. 11, 2009, No. 1, pp. 78–91.

[4] Diot, C.—Levine, B.—Lyles, B.—Kassem, H.—Balensiefen, D.: Deployment Issues for the IP Multicast Service and Architecture. IEEE Network, Vol. 14, 2000, No. 1, pp. 78–88.

[5] Castro, M.—Druschel, P.—Kermarrec, A. M—Rowstron, A: Scribe: A Large-Scale and Decentralized Application-Level Multicast Infrastructure. IEEE Journal on Selected Areas in Communications, Vol. 20, 2002, No. 8, pp. 1489–1499.

[6] Banerjee, S.—Bhattacharjee, B.—Kommareddy, C.: Scalable Application Layer Multicast. In Proc. of ACM Sigcomm, Pittsburgh, 2002, pp. 205–220.

[7] Tran, D. A.—Hua, K. A.—Do, T. T.: ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming. In Proc. of IEEE INFOCOM, San Francisco, 2003, pp. 1283–1292.

[8] Zhang, B.—Jamin, S.—Zhang, L.: Host Multicast: A Framework for Delivering Multicast to End Users. In Proc. of IEEE INFOCOM, New York, 2002, pp. 1366–1375.

[9] Banerjee, S.—Kommareddy, C.—Kar, K.—Bhattacharjee, B.: Construction of an Efficient Overlay Multicast Infrastructure for Real-Time Applications. In Proc. of IEEE INFOCOM, 2003, pp. 1521–1531.

[10] Li, L.—Cui, J. H.—Gerla, M.—Chen, S. G.: A Scalable Overlay Multicast Architecture for Large-Scale Applications. IEEE transaction on parallel and distributed systems, Vol. 18, 2007, No. 4, pp. 449–459.

[11] Liebeherr, J.—Nahas, M.—Si, W.: Application-Layer Multicasting With Delaunay Triangulation Overlays. IEEE Journal on Selected Areas in Communications, Vol. 20, 2002, No. 8, pp. 1472–1488.

[12] Pendarakis, D.—Shi, S.—Verma, D.—Waldvogel, M.: ALMI: An Application Level Multicast Infrastructure. In Proc. of the 3rd Usenix Symposium on Internet Technologies & Systems, San Francisco, 2001, pp. 49–60.

[13] Chu, Y. H.—Rao, S. G.—Zhang, H.: A Case for End System Multicast. ACM SIGMETRICS Performance Evaluation Review, Vol. 28, 2000, No. 1, pp. 1–12.

[14] Li, Z.—Mohapatra, P.: Hostcast: A New Overlay Multicasting Protocol. In Proc. of IEEE International Communications Conference, Alaska, 2003, pp. 11–15.

[15] Mathy, L.—Canonico, R.—Simpson, S.—Hutchison, D.: Scalable Adaptive Hierarchical Clustering. IEEE Communication Letter, Vol. 6, 2002, No. 3, pp. 117–119.

[16] Zhao, Q.—He, Y.—Zhang, J. Z.: A Hybrid Approach for Overlay Multicast. In Proc. of the First International Multi-Symposiums on Computer and Computational Sciences, 2006, pp. 496–502.

[17] Li, X. L.—Striegel, A. D.: A Case for Passive Application Layer Multicast. Computer Networks, Vol. 51, 2007, No. 11, pp. 3157–3171.

[18] Strufe, T.—Schäfer, G.—Chang, A.: BCBS: An Efficient Load Balancing Strategy for Cooperative Overlay Live-Streaming. In Proc. of IEEE ICC, 2006, pp. 304–309.

[19] Banerjee, S.—Bhattacharjee, B.: Comparative Study of Application Layer Multicast Protocols. Available on: `http://wmedia.grnet.gr/P2PBackground/a-comparative-study-ofALM.pdf`.

[20] Tan, S. W.—Waters, G.—Crawford, J.: A Performance Comparison of Self-Organising Application Layer Multicast Overlay Construction Techniques. Computer Communications, Vol. 29, 2006, No. 12, pp. 2322–2347.

[21] Banerjee, S.—Lee, S.—Bhattacharjee, B.—Srinivasan, A.: Resilient Multicast Using Overlays. IEEE/ACM Transactions on Networking, Vol. 14, 2006, No. 2, pp. 237–248.

[22] Guo, J.—Jha, S.: Placing Multicast Proxies for Internet Live Media Streaming. In Proc. of the 32nd IEEE Conference on Local Computer Networks, 2007, pp. 149–156.

[23] Kobayashi, M.—Nakayama, H.—Ansari, N.—Kato, N.: Reliable Application Layer Multicast Over Combined Wired and Wireless Networks. IEEE Transactions on Multimedia, Vol. 11, 2009, No. 8, pp. 1466–1477.

[24] Li, J.—Wang, Y.—Xue, M.—Tang, Z.: A Short Delay Degree-Constrained Hierarchical Application Layer Multicast Protocol. In Proc. of the International Conference on Web Information Systems and Mining, 2009, pp. 674–681.

[25] Laoutaris, N.—Smaragdakis, G.—Bestavros, A.—Byers, J. W.: Implications of Selfish Neighbor Selection in Overlay Networks. In Proc. of IEEE INFOCOM, 2007, pp. 6–12.

[26] Jin, X.—Yiu, W. P. K.—Chan, S. H. G.: Loss Recovery in Application-Layer Multicast. IEEE Multimedia, Vol. 15, 2008, No. 1, pp. 18–27.

[27] Dán, G.—Fodor, V.: Stability and Performance of Overlay Multicast Systems Employing Forward Error Correction. Performance Evaluation, Vol. 67, 2010, No. 2, pp. 80–101.

[28] Özkasap, Ö.: End-to-end Epidemic Multicast Loss Recovery: Analysis of Scalability and Robustness. Computer Communications, Vol. 32, 2009, No. 4, pp. 668–678.

[29] Wong, K. F. S.—Chan, S. H. G—Wong, W.—Zhang, Q.—Zhu, W.—Zhang, Y.: Lateral Error Recovery for Application-Level Multicast. In Proc. of IEEE Infocom, 2004, pp. 2708–2718.

[30] Hosseini, M.—Ahmed, D. T.—Shirmohammadi, S.—Georganas, N. D.: A Survey of Application-Layer Multicast Protocols. IEEE Communications Surveys & Tutorials, Vol. 9, 2007, No. 3, pp. 58–74.

[31] Costa, M.—Castro, M.—Rowstron, A.—Key, P.: PIC: Practical Internet Coordinates for Distance Estimation. In Proc. of ICDCS '04, Tokyo, 2004, pp. 178–187.

[32] Eugene, N. T.—Zhang, H.: Predicting Internet Network Distance with Coordinates-Based Approaches. In Proc. of IEEE INFOCOM, 2002, pp. 170–179.

[33] Ratnasamy, S.—Handley, M.—Karp, R.—Shenker, S.: Topologically-Aware Overlay Construction and Server Selection. In Proc. of IEEE INFOCOM, 2002, pp. 1190–1199.

[34] IETF rmt group. Available on: `http://datatracker.ietf.org/wg/rmt/`.

[35] Malouch, N.—Liu, Z.—Rubenstein, D.—Sahu, S.: A Graph Theoretic Approach to Bounding Delay in Proxy-Assisted, End-System Multicast. In Proc. of the 10th IEEE Int. Workshop on Quality of Service (IWQoS), 2002, pp. 1–10.

[36] Calvert, K.—Zegura, E.—Bhattacharjee, S.: How to Model an Internetwork. In Proc. of IEEE INFOCOM, 1996, pp. 594–602.

[37] Ns-2 Network Simulator. Available on: `http://www.isi.edu/nsnam`.

Xinchang Zhang received the Ph. D. degree from the Computer Network Information Center of Chinese Academy of Sciences, and the M. Sc. degree from the Shandong University of Science and Technology, China. Currently he is working at the Shandong Computer Science Center, Shandong Academy of Sciences, China. His research interests include network protocols and architectures, multicasting, and software testing. He has over 20 papers in research journals and international conference proceedings in these areas.

Weidong Gu is the Director of the Shandong Computer Science Center of Shandong Academy of Sciences, China. He received the M. Sc. degree from the Tianjing University, China. In the recent years, he has received the scientific and technological progress award four times. His research interests include supercomputing and computer networks.

Meihong Yang received the M. Sc. degree from the Shandong University, China. Currently she is the Associate Technology Officer and Professor of the Shandong Computer Science Center of Shandong Academy of Sciences, China. Her research interests include software engineering and cloud computing.

Guanggang Geng is a Research Assistant at the Computer Network Information Center, Chinese Academy of Sciences. He received his Ph. D. degree in pattern recognition and intelligent systems from Institute of Automation, Chinese Academy of Sciences. Currently he is interested in machine learning, adversarial Information Retrieval on the Web, and Web Search.

Wanming Luo is an Associate Professor at the Computer Network Information Center, Chinese Academy of Sciences. He obtained Ph. D. degree in Computer Science from the Institute of Computing Technology, Chinese Academy of Sciences in 2001. His current research interests include computer networks, modeling, and mobile computing. He has co-authored over 30 papers in research journals and international conference proceedings in these areas.