DOI: 10.2298/CSIS101223034X

A Two-Tiered Reliable Application Layer Multicast

Xinchang Zhang[1], Meihong Yang[1], Guanggang Geng[2],
Wanming Luo[2], and Xingfeng Li[3]

[1] Shandong Key Laboratory of Computer Networks,
Shandong Computer Science Center,
Jinan, 250014, China
xinczhang@hotmail.com, yangmh@keylab.net
[2] Computer Network Information Center,
Chinese Academy of Sciences,
Beijing, 100190, China
{gengganggang, luowanming}@cnnic.cn
[3] The People's Bank of China,
Beijing, 100800, China
li_xingfeng@126.com

Abstract. This paper presents a two-tiered reliable application layer multicast (ALM) solution, called HRALM, to provide lossless ALM services. HRALM builds a domain-based multicast tree, and divides the members into two transport planes (i.e., Plane 1 and Plane 2) in terms of the tree. In any domain, the distance between each member and the domain header is below a given threshold, which improves the capability of topology-awareness. According to the loss detected by members in different planes, HRALM adopts different but correlated recovery solutions. In HRALM, a member duplicates and forwards the received recovery packet to each of its children if it has not received the data unit carried by the recovery packet before receiving the recovery packet, which can actively recover the loss at the downstream nodes. The simulation experiments show that HRALM has desirable transport and recovery performance.

Keywords: application layer multicast, negative acknowledgement, loss, recovery.

1. Introduction

In group applications (e.g., file distribution and multi-party game), multicast is the most efficient approach because it saves much bandwidth and greatly reduces the load of servers. Multicast functionality was originally implemented at the IP layer. IP multicast is an excellent approach to deliver multicast packets, without any unnecessary data duplication. However, IP multicast has some drawbacks that are the hurdles to its ubiquitous deployment. For examples, IP multicast depends on the support of multicast routers, and IP

multicast can make the entries of route forwarding tables increase rapidly. More problems of IP multicast can be seen in [1].

As an alternative of IP Multicast, application layer multicast implements the multicast functionality at application layer instead of IP layer. In ALM, network infrastructures need no additional modification, which addresses the problem of non-ubiquitous deployment of IP Multicast across wide-area. A major disadvantage of building ALM trees is that the members have no direct knowledge of the underlying topology, which brings some unavoidable performance penalties. In other words, ALM accelerates multicast deployment at the cost of acceptable performance penalties (such as additional traffic load and latency). Presently, ALM has been widely researched (see [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]).

In the delivery tree of IP multicast, branch nodes (other than the root) are multicast routers, and leaf nodes are group members. In contrast, no-leaf nodes in ALM trees are dynamic group members instead of multicast routers. Therefore the transport in ALM is obviously unreliable. In the following parts, reliable ALM refers to the ALM that can provide lossless service [16]. Providing end-to-end reliability through TCP is a choice of implementing the reliable ALM. However, it is difficult for this approach to obtain effective flow control in the viewpoint of group communication, because the end-to-end transmissions are asynchronous. For the same reason, the approach requires frequent data numbering and renumbering operations. Additionally, some end-to-end transmissions can be broken because group members can leave the group randomly [17]. LER [16] is one of few available studies on reliable ALM based on UDP. LER employs a lateral retransmission instead of a vertical retransmission from a host's ancestors. Since LER randomly divides hosts into several planes and independently builds an overlay tree in each plane, the capability of clustering nearby nodes is limited in some degree. Another limitation of LER is that it takes high measurement and computation overheads to select proper recovery neighbors [17].

This paper analyzes the transmission features of ALM delivery trees, and further proposes a two-tiered reliable application layer multicast solution HRALM. HRALM builds a domain-based multicast tree. A domain consists of three types of members, i.e., domain header (DH), domain agent (DA) and common host (CH). In each domain, the distance between each member and the domain header is below a given threshold. The distance-based domain can improve the capability of topology-awareness. HRALM divides the members into two transport planes (i.e., Plane 1 and Plane 2) in terms of the built delivery tree. Specifically, Plane 1 consists of all the domain headers, and Plane 2 consists of other members. HRALM employs different but correlated solutions to recover the loss detected by members in different planes. HRALM also uses an active recovery mechanism to improve the recovery performance. The members in the two planes play obviously different roles, and the approaches for recovering the loss at members in the two planes are also different. Therefore, HRALM can be considered as a two-tiered reliable ALM solution.

The rest of the paper is organized as follows. In Section 2, we introduce the related work. Section 3 presents an overview of HRALM architecture. HRALM′s tree building and loss recovery approaches are explained in Section 4 and 5, respectively. We evaluate HRALM performance by analyzing the simulation results in Section 6. Finally, we conclude our work in Section 7.

2. Related Work

ALM is a promising solution to provide the delivery service to group applications. Presently, ALM has been widely researched. In the delivery tree of IP multicast, branch nodes (other than the root) are multicast routers, and leaf nodes are group members. However, no-leaf nodes in ALM delivery trees are dynamic group members instead of multicast routers, which make the delivery of ALM unreliable. Therefore the reliable application layer multicast technology is an important research topic.

To improve the reliability, multiple-tree multicast approaches have been proposed, e.g., CoopNet [18], SplitStream [19], THAG [20] and NHAG [21]. Multiple-tree multicast constructs multiple paths between the root and each group member and delivers descriptions by using MDC [22] [23] to split the original streaming media into several descriptions. CoopNet proposes a centralized algorithm to facilitate deployment of multiple-multicast trees from different sources, and does not have explicit mechanisms to maximize bandwidth. SplitStream is a tree-based multicast algorithm based on structured overlay networks. THAG and NHAG can construct the node-disjoint multicast tree. Though the multiple-tree multicast approaches can improve the reliability of ALM, they are not reliable ALM solutions.

In ALMI [3], data distribution along the multicast tree occurs on a hop by hop fashion. Depending on the application, the data transfer between two adjacent members can be reliable or unreliable by deploying TCP or UDP, respectively. Yoid [8] also gives a similar TCP-based scheme for providing the reliable service. [17] explains some cases where TCP cannot provide good end-to-end reliability for group members in ALM environment, including: If a member leaves or fails, all the member's descendants need to reconnect to the remaining overlay and establish new TCP sessions from where they stopped; While it's easy to reconnect to the overlay, it's not guaranteed that the data flows can be restarted from where they are stopped; If the buffer of a member host has finite size, the packets needed by the newly established TCP session might not be in the buffer. [16] points out that (1) TCP-based reliable approach may not achieve high throughput due to TCP backoff mechanism, (2) the hosts at the leaves of the delivery tree may suffer from high delay, as a data segment has to be completely received before being forwarded downstream, and (3) it is not obvious to extend TCP in hop-by-hop, packet-by-packet manner for the reliable service.

LER [16] is one of few available studies on reliable ALM based on UDP. LER employs a lateral retransmission instead of a vertical retransmission from

a host's ancestors. LER randomly divides hosts into several planes and independently builds an overlay tree in each plane. In a plane, a host acts as the multicast tree root (i.e., the plane source). The original source sends data to all the plane sources, which then distribute data along their own trees. Each host selects some hosts in other planes as its recovery neighbors, which are sorted according to the estimated recovery latency. A limitation of LER is that it takes high measurement and computation overheads to select proper recovery neighbors [17]. Clustering nearby nodes is a promising approach for building the delivery tree with low end-to-end delay and network traffic. However, randomly dividing the hosts into some planes weakens the above advantage to some extent.

In contrast, providing reliable service based on IP multicast has been widely studied (see [24, 25, 26, 27]). The NACK-based recovery mechanism is widely adopted in the existing approaches for reliable IP multicast. In the NACK-based mechanism (e.g., NORM [25]), the receiver (i.e., the group member other than the root in the delivery tree) sends a NACK message to request the receiver (of the message) to retransmit the recovery packet. The NACK-based recovery mechanism reduces the retransmission delay to some extent. Because of the above intrinsic difference, it is unwise for reliable ALM to directly leverage some existing approaches that work well in reliable IP multicast.

3. Overview of HRALM Architecture

The design objective of HRALM is to provide reliable (i.e., lossless) ALM services to the group members. Specifically, HRALM builds an ALM tree to distribute the data and recovers the loss in the distribution procedure. HRALM uses our proposed TCM model (see [28]) to build the multicast tree and divides the group members into two transport planes in terms of the built tree. In the structure of the HRALM tree, there are many distance-based domains, and most of group members each belong to a domain. Each domain has a center, called domain header (DH). In any domain, the distance between each member and the domain header is within a given cluster threshold (denoted by λ). In HRALM, there exist some members, called foreign hosts (FHs), which do not belong to any domain. Except DH and FH, there also are two types of members, i.e., domain agent (DA) and common host (CH). Fig. 1 illustrates the four types of members:

- Domain header: DH is the center of the corresponding domain, i.e., the distance between each member in the domain and the header is not more than the given domain threshold λ. The child of a DH member might be (1) a CA in the same domain, or (2) a DH in a different domain, or (3) a FH which does not belong to any domain.

- Domain agent: DA is a child of the DH of the corresponding domain. DA may accept three types of members (i.e., CHs in the same domain, FHs, and DHs in different domain) as its children.
- Common host: The parent of each CH is a DA node, and only other CHs in the same domain and FHs can become the children of a CH node.
- Foreign host: A FH (denoted by *r*) does not belong to any domain, and the parent of *r* might be a DH, DA, or CH. Let *h* mean the first upstream DH node in *r*'s root path, then the distance between *r* and *h* is more than λ. Only other FHs can become the children of a FH.

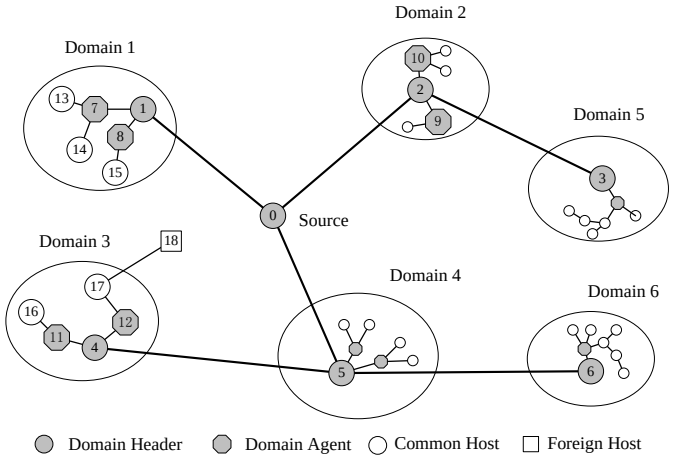

Fig. 1. Structure of the HRALM tree

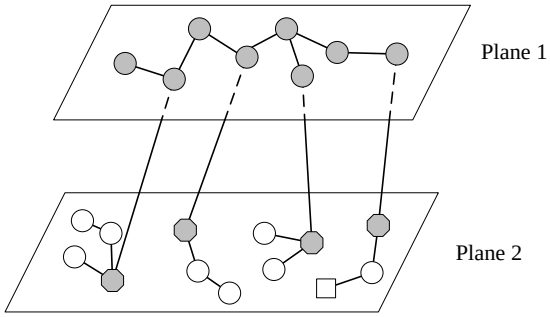

Fig. 2. The two-tiered transport planes of HRALM

According to the above description, we know that a domain consists of three types of members, i.e., DHs, DAs, and CHs. In the following parts, domain members represent the members (except the DH) in a certain domain. In HRALM, each member keeps the corresponding addresses and types of its children. If a child is *X* (DH, DA, CH, or FH) node, we call it *X*

child. Additionally, we say that a domain *D* is leaded by member *m* if *m* is the domain header of *D*.

The tree root (i.e., the data source) is a special DH member, and the domain leaded by the root only contains a single member (i.e., the DH). In HRALM, there are two types of transport planes, i.e., Plane 1 and Plane 2. All the DH members belong to Plane 1, while other members belong to Plane 2. Fig. 2 shows the structure of the above two-tiered transport planes.

In the application layer multicast, clustering nearby members can make the tree structure congruent to the network topology to some extent. Based on the above heuristic, HRALM adopts the above structure to cluster the members. Another important reason of using the above structure is to form the two-tiered transport planes in terms of the types of members.

In the reliable multicast, the confirmation entity is either packet or ADU, as IETF RFC 2887 [29] explains. HRALM uses ADU-level confirmation, which corresponds to ALM in nature. In HRALM, there are two types of negative acknowledgement packets, i.e., NACK1 (sent by members in Plane 1) and NACK2 (sent by members in Plane 2). Since the members in Plane 1 are usually in the top of the HRALM tree, HRALM attempts to recover the loss at members in Plane 1 through a quick and robust approach. The above quick and robust recovery ensures that the nodes in Plane 1 are relatively reliable. Based on these relatively reliable nodes, HRALM uses multi-round approach to recover the loss at members in Plane 2. Through the above hierarchical recovery solution, the recovery performance in HRALM is effectively improved.

HRALM also use an active recovery mechanism, i.e., any member duplicates and forwards the received recovery packet if it has not received the data unit carried by the packet before the arrival of the packet. In ALM, a loss at some tree node (denoted by *m*) must result in the same loss at the downstream nodes of node *m*. Therefore the above mechanism can actively recover the loss at the downstream nodes and reduce the recovery delay.

For providing lossless delivery service, the data source in HRALM buffers all the sent ADUs. However, the member (other than the root) buffers the latest received ADUs in terms of a fixed buffer size designed by the member.

4. Tree Construction of HRALM

In this section, we will introduce the TCM-based tree building approach used in HRALM. In the approach, the group information (e.g., the address of the root) is memorized by a Rendezvous Point (RP). When the newcomer wants to join the group, it first contacts the RP and gets the address of the data source. Then the newcomer joins the group in terms of the join algorithm shown in Fig. 3.

In the join algorithm, *candi* represents a candidate node which might become the parent of the newcomer. The join algorithm uses round trip time (rtt) as the distance metric. In this paper, we use $d(m,n)$ to denote the round

trip time from *n* to *m*. In the algorithm, JoinDomain(*m*,*n*,1) means host *n* joins the domain whose header is *m*, and JoinDomain(*m*,*n*,2) represents *n* become a downstream FH node of *m*. We will give more explanation on the above two procedures in next part. In terms of Line 3 in the join algorithm, *n* cannot become a domain member of the domain leaded by *s* if $d(s,n) \leq \lambda$. Line 4 in the algorithm is executed if (1) $d(m,n)>\lambda$ and $f(n) \geq 2$, or (2) $d(m,n) \leq \lambda$ and $m=s$. In HRALM, there are three types of join messages, i.e., JoinRequest (*t*,0), JoinRequest (*t*,1) and JoinRequest (*t*,2), where *t* denotes the receiver of the join messages. If a node accepts a join request from a member of type A, then the new child of the node is marked with symbol A.

Procedure Join (*s*,*n*)

// Newcomer *n* joins the group. *s* denotes the data source.

1: Initialize：*candi*←*s*; *S* is allocated for storing past *candi* nodes. All members have no any label by default. // *S* is a stack
2: Query *candi* to discover all its DH children. Measure the rtts from *candi* and its DH children to *n*. Let Ω mean the set of *candi* and its DH children.
3: Find the nearest node (*m*) among nodes in Ω. If $d(m,n) \leq \lambda$ and $m \neq s$, then JoinDomain(*m*,*n*,1); If $d(m,n)>\lambda$ and $f(n)<2$, then JoinDomain(*m*,*n*,2).
4: Find the nearest member (denoted by *m'*) among nodes (without the *full* labels) in Ω. If all nodes in Ω are marked with a *full* label, then pop the top element *p* of *S*, *candi*←*p*, go to Line 2.
5: If *m'* is not the current candidate node, push *candi* onto the stack *S*, *candi*←*m'*, go to Line 2.
6: Send a join request JoinRequest (*candi*,0) to *candi*. If the join request is refused, then mark *candi* with a *full* label and go to Line 4. Otherwise, *candi* becomes *n*'s parent node, and *n* mark itself with a DH label.

Fig. 3. Join algorithm

In this paper, we use $f(m)$ to denote the fanout of member *m,* i.e., the maximum number of children which host *m* is willing to accommodate in the multicast tree [2]. Similarly, we use $f'(m)$ to denote the remnant fanout of *m*. $f'(m)$ is equal to the number of existing children subtract $f(m)$. The fanout of member *m* is obtained by sending some detecting packet before *m* joins the group. Let $\sigma_1(k)$, $\sigma_2(k)$, $\sigma_3(k)$ and $\sigma_4(k)$ denote the number of DH children of *k*, the number of DA children of *k*, the number of CH children of *k* and the number of FH children of *k*, respectively. Then a DH (donated by *d*) responds to a join request sent by *t* in terms of the following cases:

(1) If $f'(d) \geq 2$, *d* accepts join request of any type.
(2) If $f'(d)=1$, $\sigma_2(d) \geq 1$ and *d* receives a JoinRequest(*d*,0) message, then *d* accepts *t* as its DH child.

(3) If $f'(d)=1$, $\sigma_1(d) \geq 1$, and d receives a JoinRequest(d,1) message, then d accepts t as its DA child.
(4) If $f'(d)=0$, $\sigma_4(d) \geq 1$, and d receives a JoinRequest(d,0) message, then d: accepts t as its DH child; randomly selects a FH child and tells it to rejoin the group starting from d.
(5) If $f'(d)=0$, $\sigma_4(d) \geq 1$, and d receives a JoinRequest(d,1) message, then d: accepts t as its DA child; randomly selects a FH child and tells it to rejoin the group starting from d.
(6) Otherwise, d rejects the join request.

According the first three cases, we can see that the DH node reserves the child location to DA and other DH nodes. From case (4) and (5), we also know that DH and DA nodes have priority over FH nodes when they compete for the same child location.

Procedure JoinDomain(*m*,*n*,*k*)
// Newcomer *n* joins a domain leaded by *m*. *k*=1 or 2.
1: Initialize：*candi_D*←*m*; stack *D* is allocated for storing past *candi_D* nodes.
2: Query *candi_D* to discover all its children. Let Ψ mean the set of *candi_D*, DA and CH children, and FH children if *k*=2. Measure the rtt from each member in Ψ to *n*.
3: Find the nearest member among nodes without *full* label in set Ψ. If all nodes in Ψ are with a *full* label, then pop the top element *p* in *D*, *candi_D*←*p*, go to Line 2.
4: If the nearest member is not the current candidate node, push *candi_D* into the stack *D*, *candi_D*←the nearest member, go to Line 3.
5: Send a JoinRequest (*candi_D*,*k*) request to *candi_D*. If refused, mark *candi _D* with a *full* label and go to Line 3. Otherwise, *candi_D* becomes *n*'s parent node, and mark itself by a DA label if *candi_D*=*m* and *k*=1, or A CH label if *candi_D*≠*m* and *k*=1, or A FH label if *k*=2.

Fig. 4. Join the group to become a domain member or FH

The CM or FH node (denoted by b) accepts join request only if $f'(b) \geq 1$. The DA node (donated by g) responds to a join request sent by t in terms of the following cases:

(1) If $f'(g) \geq 2$, g accepts join request of any type.
(2) If $f'(g)=1$, $\sigma_4(g) \geq 1$ and g receives a JoinRequest(g,1) message, then: g accepts t as its CH child; g randomly selects a FH child of g and tells it to rejoin the group starting from g.
(3) Otherwise, g rejects the join request.

In the application layer multicast, clustering nearby members makes the tree structure congruent to the network topology to some extent, which can effectively improve the performance of the ALM tree. The TCM-based tree building approach tries to divides most of the group members into many domains in terms of the distance, which is helpful for HRALM to cluster the

members. HRALM positions the newcomer by searching the existing tree. The searching procedure stops when it reaches a leaf node, or a node that is closer to the newcomer than the related neighbors. Therefore, we can say that the HRALM tree is topology-aware in some degree.

In HRALM, each host periodically sends the message including current root path to its children, and instantly sends the message when it finds that its root path is changed. Each member *m* also periodically sends the heartbeat messages to keep its neighbor nodes active. When a DH node leaves the group gracefully, it will actively tell the neighbors to cope with its leave. If a member leaves the group without any notification, all the children of the member rejoin the group starting from a closest and active upstream node in its root path. As to the other maintenance procedures, such as structure update, partition recovery, loop detection and resolution, HRALM can use the approaches of some tree-based ALM protocols. We do not care these details in this paper.

5. Loss Recovery of HRALM

In this section, we first introduce the transmission characters of ALM, which is the basis of the design of the loss recovery in HRALM. Then we introduce the related packet and timer types. Finally, we explain the loss recovery solutions for the members in Plane 1 and 2, respectively.

5.1. Transmission Characters of ALM

In IP multicast tree, the loss at a group member (other than the root) has no direct influence on other group members. In contrast, if a node (i.e., group member) in the ALM tree could not receive some correct packet for any reason, all its downstream nodes in the tree would lose the corresponding correct packet because group members take on the forwarding functionality of multicast routers. In ALM, from the root to each member, there is one unique loop-free path along the multicast tree. The member list of this path is called root path [2]. Therefore, there is a high error correlation among the nodes in a root path. Clearly, the group member in multicast session can leave randomly, and has limited capability of forwarding the data because of some reasons (e.g., network congestion and resource exhaust). Therefore the forwarding functionality of the member host is unreliable in ALM.

In a delivery tree (including n+1 nodes) whose maximum node degree is *k*, we can easily conclude that the node at level *i* has at least $n+1-\frac{(k^{i-1}-1)}{k-1}-k^{i-1}$ downstream nodes.

In a given delivery tree, if a node *n* at level *i* loses a correct packet but all its upstream nodes in its root path receive the correct packet, we say that an

interrupt event happens at the node *n,* denoted by $I(n)$. We use $p_j(n)$ to mean the *j*th upstream node in *n*'s root path. For examples, $p_1(n)$ means the parent of node n, and $p_2(n)$ denotes *n*'s grandfather. Assume that the event, that a node cannot correctly receive each packet sent by its parent in the delivery tree, is independent distributed. Then we have:

Lemma 1. The interrupt event happens with higher probability at lower level in the delivery tree. Note that the root is at the lowest level (i.e., level 1).

Proof. Let a node *n* cannot receive the correct packet sent by its parent with probability of α_n, then the interrupt event happens at a node n with probability of $\Pr(I(n))$, $\Pr(I(n)) = \alpha_n \prod_{j=1}^{i-1}(1-\alpha_{p_j(n)})$, where *i* is the level of node *n* in the delivery tree. Therefore we can easily prove the lemma.

According to the above description, we can notice that the interrupt event usually has heavy negative influence on the reliability of the application layer multicast.

5.2. Packet and Timer Types

There are five types of messages (i.e., NACK1, NACK2, NACK1_T, NACK2_A and RECOVERY) and two types of timers (i.e., T_NACK1 and T_NACK2) in the recovery procedure of HRALM, as Table 1 and 2 show.

Table 1. The packet types related to the loss recovery

Packet type	Description
NACK1	Negative acknowledgement sent by a DH
NACK2	Negative acknowledgement sent by a member in Plane 2
NACK1_T	NACK1 acknowledgement packet
NACK2_A	NACK2 acknowledgement packet
RECOVERY	Recovery packet

Table 2. The timer types related to the loss recovery

Timer type	Description
T_NACK1	NACK1 retransmission timer
T_NACK2	NACK2 retransmission timer

The NACK1 and NACK2 messages are the retransmission requests sent by members in Plane 1 and Plane 2, respectively. When a DH node received a NACK1 message from member *m*, it sends the NACK1_T message to tell *m* to restart its T_NACK1 timer if the expected ADU is not in its buffer and it will forward the NACK1 message to other DHs. The NACK1_A message is used

to tell the receiver to contact the next recovery source. HRALM can distinguish the RECOVERY message from the normal data packet by identification in the application data unit. A member forwards the received recovery packet if it has not received the data unit carried by the packet before the arrival of the packet. More detail on the above messages can be seen in the following section.

The time intervals of T_NACK1 and T_NACK2 timers are denoted by t_{NACK1} and t_{NACK2}, respectively. When a member in Plane 1 finds that it has lost a correct ADU, it sends the NACK1 messages to some nodes by the unicast means and starts the T_NACK1 timer. Similarly, a member in Plane 2 sends a NACK2 message to a DH by the unicast means and starts the T_NACK2 timer when the member detects a loss. Let *rtt* (*m*) means the round trip time from the data source to member *m*, then t_{NACK1} and t_{NACK2} each are larger than *rtt*(*m*).

5.3. Loss Recovery for DHs

In this paper, we say that a member is at level *k* if there are (*k*-1) upstream nodes in its root path. In HRALM, each DH *m* at level *k* keeps a recovery list (denoted by $L_{m,k}$).The recovery list saves some member nodes (called recovery neighbors) which are potential loss recovery sources for *m*. Note that the tree root is not included in any recovery list. Let $v(L_{m,k})$ denote the number of recovery neighbors in $L_{m,k}$, then

$$v(L_{m,k})=max\{\text{int}(|\alpha-\log k|), \beta\}\ , \quad (1)$$

where α and β are two configuration parameters. Specifically, α and β denote the upper bound and lower bound of the number of recovery neighbors, respectively. The upper bound is used to confine the maximum number of NACK1 messages in the loss recovery procedure, while the lower bound ensures that the loss recovery for DHs is robust. The default values of α and β are 4 and 2, respectively. According to the rule of hierarchical recovery, Eq. (1) also considers the influence of the level of a member in some degree. In HRALM, the address of a DH is periodically delivered to the group members by the multicast way. Then DH node *m* randomly selects $v(L_{m,k})$ DH nodes as its recovery neighbors in terms of the above address information. Additionally, a DH node periodically sends a heartbeat message to each recovery neighbor to acknowledgement the living of the recovery neighbor. Once finding that some recovery neighbor is not active, the DH node replaces it with an active DH node.

The algorithm shown in Fig. 5 explains the procedure of recovering the loss at a DH node. In this paper, *NACK1*(*n*,*m*) denotes the NACK1 message that is used to tell the receiver of the message to retransmit ADU *n* to member *m*, *NACK1_T*(*n*) means the NACK1_T message that is used to tell the receiver of the message to restart the T_NACK1 timer for ADU *n*, and *RECOVERY*(*n*) represents the RECOVERY message that carries ADU *n*. When member *m* in Plane 1 (i.e., DH *m*) finds that it has lost ADU *n*, it instantly sends a NACK1 message to each recovery neighbor in the recovery list by the unicast means,

for requiring the latter to retransmit ADU *n* to *m*, and starts the T_NACK1 timer.

Procedure RecoverDH(*m*, *n*,*v*($L_{m,k}$))
// Recover the lost ADU *n* of DH *m*.
1: *m* sends *NACK1*(*n*,*m*) to each node in *v*($L_{m,k}$), to request the latter to retransmit ADU *n*; *m* starts the T_NACK1 timer.
2: If *m* receives *NACK1_T*(*n*), it restarts the T_NACK1 timer.
3: If *m* receives *RECOVERY*(*n*) before the T_NACK1 timer expires, the loss at *m* is recovered. Additionally, it duplicates and forwards the message to its children if it has not previously received the ADU.
4: If *m* have not received *RECOVERY*(*n*) before the T_NACK1 timer expires, it sends *NACK1*(*n*,*m*) to the root.

Fig. 5. The recovery algorithm for the DH node

When DH *p* receives *NACK1*(*n*,*m*), it retransmits ADU *n* to *m* if ADU *n* is in its buffer; Otherwise, *p* sends *NACK1*(*n*,*m*) to each recovery neighbor in the recovery list and sends *NACK1_T*(*n*) to *m*. Any member duplicates and forwards received *RECOVERY*(*n*) to all its children if it has not received ADU *n* before the arrival of the recovery packet. In the application layer multicast, the loss at a member results in the same loss at all downstream nodes of the member. Therefore the above solution can actively recover the loss at downstream members. However, the active recovery might produce some repeated ADUs. To address the above problem, the members detect and discard the repeated ADUs.

In the practical application, it is seldom that a DH node sends a NACK1 message but receives no response. For providing complete reliable loss recovery, HRALM also copes with the above situation, i.e., a DH node sends the NACK1 message to the tree root if there is no response to the previous NACK1 message (see Line 4 of the algorithm shown in Fig. 5).

From the algorithm shown in Fig. 5, we can notice that the possible recovery source (i.e., *m*'s recovery neighbors, recovery neighbors of *m*'s recovery neighbors, and so on) exponentially increases in the recovery procedure. Therefore the loss recovery for the DH node is quick and robust.

Fig. 6 illustrates two examples of recovering the loss at a DH node. In Fig. 6a, member *m* sends the NACK message (i.e., *NACK1*(*n*,*m*)) to each node in its recovery list (including n_1, n_2 and n_3). Once receiving *NACK1*(*n*,*m*), node n_3 retransmits ADU *n* to *m* because ADU *n* is in its buffer. In the example, member *m* receives no response from node n_1 and n_2 for some reasons, such as the network congestion and member departure. Fig. 6b depicts another recovery procedure. In this case, n_1 sends *NACK1_T*(*n*) to *m* and sends *NACK1*(*n*,*m*) to each node in its recovery list. Finally, *m* gets *RECOVERY*(*n*) from n_4.

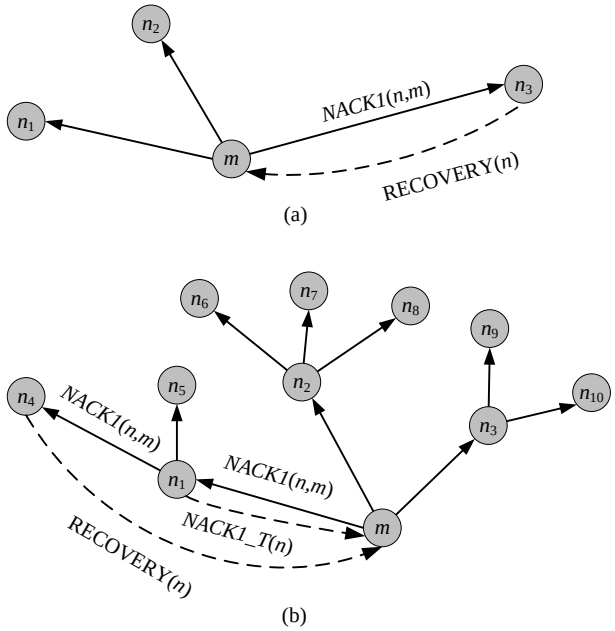

Fig. 6. Examples of recovering the loss at a DH node

5.4. Loss Recovery for Members in Plane 2

As noted previously, only the members in Plane 2 (i.e., DAs, CHs and FHs) can send the NACK2 messages. HRALM employs a multi-round procedure to recover the loss triggered by NACK2. In this situation, a recovery round starts when a member sends a NACK2 message to one of its upstream nodes in the corresponding delivery tree, and ends when the member receives the expected ADU (carried by RECOVERY message) or T_NACK2 timer expires.

Assume that there are $n_r(m)$ nodes between m and the tree root in m's root path, and $n_h(m)$ nodes between m and the first (i.e., closest) DH node in m's root path, then we define a recovery source selection function as

$$U_i(m)=\min\{n_r(m), n_h(m)+i\}\ , \qquad (2)$$

When member m in Plane 1 finds that it has lost a correct ADU (denoted by n), it waits a random time interval between 0 and $rtt(m)$, then sends a NACK2 message (i.e., $NACK2(n,m)$) to the $U_1(m)$th upstream node in m's root path, for requiring the latter to retransmit ADU n, and starts the T_NACK2 timer. When the $U_1(m)$th upstream node receives the NACK2 message, it retransmits ADU n to m if ADU n is in its buffer; Otherwise, it sends a NACK2_A message to m to acknowledge the receipt of the NACK2 message. Once (1) m receives the NACK2_A message before the timer expires or (2) the T_NACK2 timer expires, it sends the NACK2 message to the $U_2(m)$th upstream node in m's root path and restarts the T_NACK2 timer. The above

procedure goes on until *m* receives the expected recovery packet. Fig. 7 gives an example of recovering the loss at the member in Plane 2.

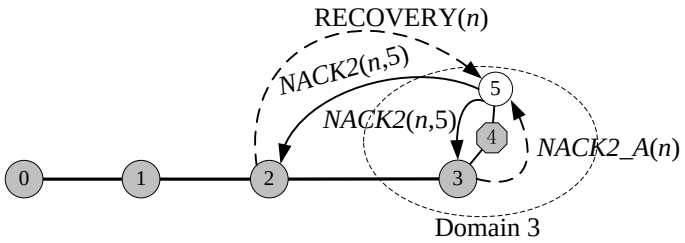

Fig. 7. Example of recovering the loss at the member in Plane 2

According to structure of the HRALM tree, the members in Plane 2 usually are at the middle or bottom of the ALM tree. Therefore the loss recovery operations for members in Plane 2 are effectively reduced by the active recovery mechanism and NACK2 suppression. From Eq. (2), we can see that the recovery source selection is based on a linear function, and each recovery source is a DH node. As noted above, the loss at a DH node can be quickly and robustly recovered. Consequently, the loss at member *m* also can be quickly recovered in most cases.

Since a domain member first sends a NACK2 message to the header of the domain that it belongs to and rechooses the next recovery source by a linear function, the NACK2 explosion problem is effectively alleviated. In addition, the NACK2 suppression further alleviates the NACK2 explosion problem.

6. Simulation Experiments

We used the GT-ITM Generator [30] to generate a 5000-node transit-stub graph as our underlying network topology. Each node represents a router, and the average degree of router nodes was about 3. We also generated 1000 nodes as member hosts, which were connected to stub-domain router nodes randomly. Each stub-domain node connected a host node at most. The fanout of 960 member hosts were assigned by a random value between 2 and 5, and the fanout of other member hosts were assigned by 1. The server was located in a random stub-domain. In the simulations, packets are randomly dropped in each link with probability of an interval μ (called reliability interval). By default, μ=[0.01,0.2], $\lambda=0.25t_{max}$, where t_{max} is the maximum value of rrts between the members and the server. We simulated the related protocols with NS-2 ([31]).

In our experiments, we first used HRALM and LER to build the ALM tree with 1000 receivers, respectively. Table 3 gives the distribution of members of different types in HRALM. From the table, we know that about 10% members are located in Plane 1. Fig. 8 compares the load of the main links, of which

each connects two nodes in different stub-domains, of HRALM and LER trees. Physical link load (stress) means the number of identical copies of a packet that traverse a physical link. As noted previously, LER randomly divides member hosts into several independent planes, which results in that some nearby nodes cannot be well clustered. Therefore LER has higher link load than HRALM (see Fig. 8). In addition, there are more related main links in LER. Note that the related links denote the links that connect the nodes in different stub-domains and transport the data packets of the group application.

Table 3. The distribution of different types of receivers

DHs	DAs	CHs	FHs
109	245	614	32

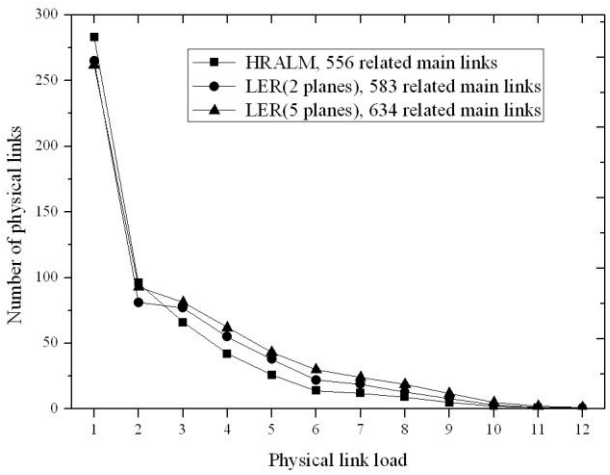

Fig. 8. Physical link load of HRALM and LER

Fig. 9 shows the mean numbers of recovery rounds of HRALM. In each scenario of these experiments, a HRALM tree with 1000 receivers was built, and then 100 ADUs are distributed along the tree with 3 reliability intervals, respectively. In this part, a loss recovery round means a recovery round in the recovery procedure for a loss at the member in Plane 2, or a NACK1 diffusion phase (i.e., the NACK messages are sent by some node for recovering a certain loss) in the recovery procedure for a loss at the member in Plane 1. In particular, the number of loss recovery rounds for a loss is zero if the loss is actively recovered. From Fig. 9, we can notice that the mean number of loss recovery rounds is low in each scenario, which means that HRALM can quickly recover the loss.

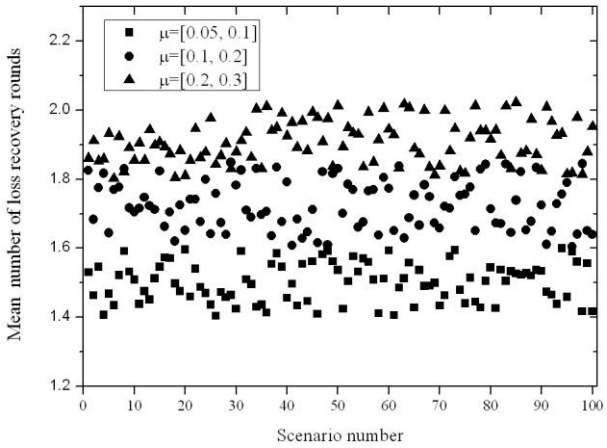

Fig.9. Loss recovery rounds of HRALM

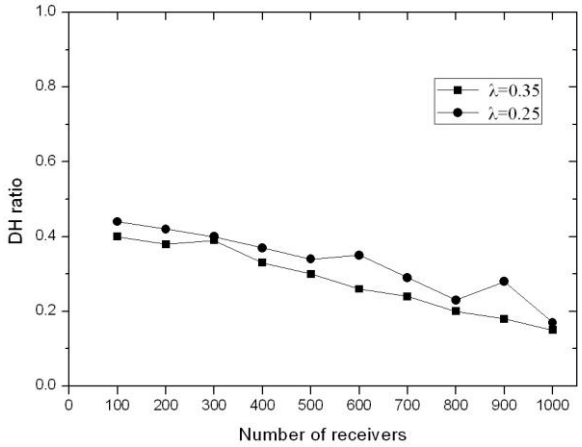

Fig.10. The number of DHs of HRALM

Fig.10 shows the number of DHs in 10 groups with different group sizes. In the figure, DH ratio denotes the ratio of the number of DHs to the number of all members. From the figure, we can see that DH ratio has a dropping trend as the group size grows, which means more and more nodes are clustered with the growth of group size. Since HRALM uses the distance-based domain to contain the domain members, the DH ratio would continue to decrease if more members joined the same group session.

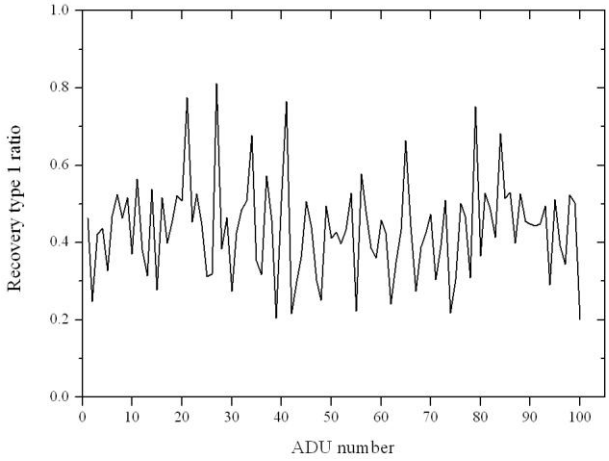

Fig.11. The recovery type 1 ratio of HRALM

In next experiments, we divided the recovery packets into two types, i.e., recovery type 1 and recovery type 2. If a recovery packet is sent to the related members because of the loss recovery launched by a DH node, then the packet is identified by type 1. Otherwise, the recovery packet is identified by type 2. Fig.11 plots the ratio of the number of the recovery packets of type 1 to the number of total recovery packets. From the figure, we can see that the recovery launched by the members in Plane 1 plays an important role through only about 10% group members are DHs.

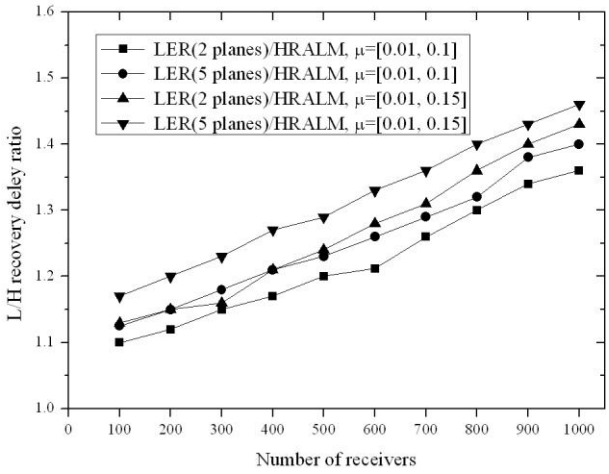

Fig.12. Recovery delay comparison of HRALM and LER

Fig.12 and Fig.13 illustrate the recovery delay and of recovery load HRALM and LER in 10 groups with different group sizes. In these experiments, we used two planes and five planes to build LER multicast tree, respectively. L/H

recovery delay ratio means the ratio of average recovery delay of LER to that of HRALM. Let $l(n)$ mean the number of links that transport the recovery packet n, and $R(A)$ mean the set of recovery packets in solution A. Then L/H recovery load ratio (denoted by *LHLR*) is defined as

$$LHLR = \sum_{n \in R(LER)} l(n) \Big/ \sum_{n \in R(HRALM)} l(n) \quad . \tag{3}$$

According to the above definition, we can notice that *LHLR* can evaluate the load of the loss recovery. From Fig.12, we can see that L/H recovery delay ratio is more than 1 in each group, and that ratio increases as the group size grows. We attribute the desirable performance to quick loss recovery for DHs and the active recovery mechanism. From Fig.13, we notice that L/H recovery load ratio is also more than 1 in each group, and has an increasing trend. Fig.12 and Fig.13 tell us that HRALM can obviously improve the recovery performance.

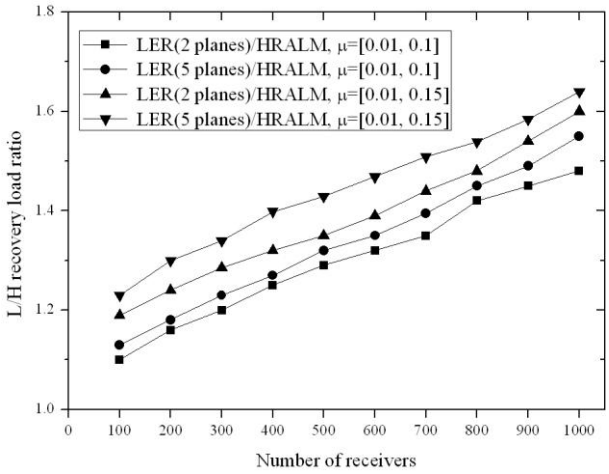

Fig.13. Recovery load comparison of HRALM and LER

7. Conclusion

In this paper, we proposed a two-tiered reliable application layer multicast solution HRALM, which can provide lossless services. In HRALM, most of members belong to the distance-based domains. In the domain, the distance between each member and the domain center is below a given threshold. The distance-based domain can cluster nearby members well. According to the structure of the domain-based tree, HRALM divides the members into four types of members, i.e., domain header (DH), domain agent (DA), common host (CH) and foreign host (FH). The DH node is the center of the corresponding domain, and all the DHs constitute a transport plane, i.e.,

Plane 1. Another plane, named Plane 2, consists of all the DAs, CHs and FHs.

HRALM uses NACK-based mechanism to recover the loss. However, the NACK message is sent to different member (or set of members) in different recovery phase or round because of the unreliability of member hosts. Since the members in Plane 1 are usually in the top of the HRALM tree, HRALM recovers the loss at members in Plane 1 through a quick and robust approach, which ensures that the nodes in Plane 1 are relatively reliable. Based on these relatively reliable nodes, HRALM uses multi-round approach to recover the loss at members in Plane 2. Through the above hierarchical recovery solution, the recovery performance in HRALM is effectively improved.

HRALM also use an active recovery mechanism. In the mechanism, any member duplicates and forwards the received recovery packet to its children if it has not received the data unit carried by the recovery packet before receiving the recovery packet. Since the loss at a member must result in the same loss at downstream node, the above mechanism can effectively improve the recovery performance.

Acknowledgments. This work was supported by the National Natural Science Foundation of China under Grant No. 61070039 and No. 61005029，the National Basic Research Program of China under Grant No. 2009CB320502，the Outstanding Young and Middle-aged Scholars Foundation of Shandong Province of China under Contract No. 2008BS01019.

References

1. Diot, C., Levine, B., Lyles, B., Kassem, H., Balensiefen, D.: Deployment Issues for the IP Multicast Service and Architecture. IEEE Network, 14(1), 78-88. (2000)
2. Zhang, B., Jamin, S., Zhang, L.: Host multicast: A framework for delivering multicast to end users. In Proceedings of IEEE INFORCOM, 1366-1375. (2002)
3. Pendarakis, D., Shi, S., Verma, D., Waldvogel, M.: ALMI: An Application Level Multicast Infrastructure. In Proceedings of the 3rd Usenix Symposium on Internet Technologies & Systems, 49-60. (2001)
4. Wang, F., Xiong, Y. Q., Liu, J. C.: mTreebone: A Collaborative Tree-Mesh Overlay Network for Multicast Video Streaming. IEEE Transactions on Parallel and Distributed Systems, 21(3), 379-392. (2006)
5. Yue, J., Wu, C.: A Trees-Mesh Based Application Layer Multicast Using Collaborative Sub-streams. In Proceedings of the Second International Conference on Future Networks, 29-33. (2010)
6. Wei, X. J., Jia, R. S., Liu, G., Yan, X. H.: Research on P2P-Based Application Layer Multicast Technology for Streaming Media. In Proceedings of the Second International Workshop on Education Technology and Computer Science, 341-345. (2010)
7. Banerjee, S., Kommareddy, C., Kar, K., Bhattacharjee, B., Khuller, S.: Construction of an Efficient Overlay Multicast Infrastructure for Real-Time Applications. In Proceedings of IEEE INFOCOM, (2003)

8. Francis, P.: Yoid: Extending the Multicast Internet Architecture. http://www.aciri.org/yoid
9. Strufe, T., Günter, S., Chang, A.: BCBS: An Efficient Load Balancing Strategy for Cooperative Overlay Live-Streaming. In: IEEE ICC, 1-12. (2006)
10. Li, X. L., Striegel, A. D.: A case for Passive Application Layer Multicast. Computer Networks, 51(11), 3157-3171. (2007)
11. Banerjee, S., Lee, S., Bhattacharjee, B., Srinivasan, A.: Resilient Multicast Using Overlays. IEEE/ACM Transactions on Networking, 14(2), 237-248. (2006)
12. Tian, R. X., Zhang, Q., Xiang, Z., Xiong, Y. Q., Li, X., Zhu, W. W.: Robust and Efficient Path Diversity in Application-Layer Multicast for Video Streaming. IEEE Transactions on Circuits and System for Video Technology, 15(8), 961-972. (2005)
13. Levine, B. N., Paul, S., Garcia-Luna-Aceves, J. J.: Organizing multicast receivers deterministically by parcket-loss correlation. Multimedia Systems, 9(1), 3-14. (2003)
14. Yoon, W., Lee, D., Youn, H. Y.: Comparison of tree-based reliable multicast protocols for many-to-many sessions. IEE Proc., Commun., 15(6): 923-93. (2005)
15. Zhang, X., Li, X., Luo, W., Yan, B.: An Application Layer Multicast Approach Based on Topology-Aware Clustering. Computer Communications, 32(6), 1095-1103. (2009)
16. Simon Wong, K.-F., Gary Chan, S.-H., Wong, W., Zhang, Q., Zhu, W., Zhang, Y.: Lateral Error Recovery for Application-Level Multicast. In: IEEE Infocom, 2708-2718. (2004)
17. Jin, X., Yiu, W. –P. K., Chan, S. –H. G.: Loss Recovery in Application-Layer Multicast. IEEE Multimedia, 15(1), 18-27. (2008)
18. Padmanabhan, V. N., Wang, H. J., Chou, P. A., Sripanidkulchai, K.: Distributing streaming media content using cooperative networking. In: ACM Int. Workshop Network and Operating System Support for Digital Audio and Video (NOSSDAV), 177-186. (2002)
19. Castro, M., Druschel, P., Kermarrec, A.-M., Nandi, A., Rowstron, A., Singh, A.: Splitstream: High-bandwidth content distribution in cooperative environments. In: Int. Workshop Peer-to-Peer Systems (IPTPS), 298-313. (2003)
20. Tian, R., Zhang, Q., Xiang, Z., Xiong, Y., Li, X., Zhu, W.: Robust and efficient path diversity in application-layer multicast for video streaming. IEEE Trans. Circuits Syst. Video Technol., 15(8), 961-972. (2005)
21. Kobayashi, M., Nakayama, H., Ansari, N., Kato, N.: Robust and efficient stream delivery for application layer multicasting in heterogeneous networks. IEEE Trans. Multimedia, 11(1), 166-176 (2009)
22. Alasti, M., Sayrafian-Pour, K., Phremides, A., Farvardin, N.: Multiple description coding in networks with congestion problem. IEEE Trans. Inf. Theory, 47(3), 891-902. (2001)
23. Goyal, V. K.: Multiple description coding: Compression meets the network. IEEE Signal Process. Mag., 18(5), 74-93. (2001)
24. Paul, S., Sabnani, K. K., Lin, J. C., Bhattacharyya, S.: Reliable Multicast Transport Protocol (RMTP). IEEE Journal on Selected Areas in Communications, 15(3), 407-421. (1997)
25. Adamson, B., Bormann, C., Handley, M., Macker, J.: Negative-acknowledgment (NACK)-Oriented Reliable Multicast (NORM) Protocol. IETF RFC 5740. (2009)
26. Wonyong, Y., Dongman, L., hy Youn, Lee, S.: A Combined Group/Tree Approach for Scalable Many-to-many Reliable Multicast. Computer Communications, 29(18), 3863-3876. (2006)
27. Yavatkar, R., Griffioen, J., Sudan, M.: A reliable dissemination protocol for interactive collaborative applications. In: ACM Multimedia, pp.333-344. (1995)

28. Zhang, X., Wang Z., Luo W., Yan Baoping. A Study on Topology-aware Application Layer Multicast Scheme. Journal of Software, 21(8), 2010-2022. (2010) (in Chinese)
29. Handley, M., Floyd, S., Whetten, B., Kermode, R., Vicisano, L., Luby M.:The Reliable Multicast Design Space for Bulk Data Transfer. RFC 2887. (2000)
30. Calvert, K., Zegura, E., Bhattacharjee, S.: How to Model an Internetwork. In: Proceedings of IEEE INFOCOM, 594-602. (1996)
31. The Network Simulator-ns2. Available: http://www.isi.edu/ns-nam/ns.

Xinchang Zhang received the Ph.D. degree from Computer Network Information Center of Chinese Academy of Sciences, Beijing, China, in 2010. Now, he is working at Shandong Computing Center, Shandong Academy of Sciences. His research interests include network protocols and architectures, multicasting, and software testing.

Meihong Yang received the M.S. degrees from the Shandong University. Now, she is the associate technology officer and professor of Shandong Computing Center of Shandong Academy of Sciences. In the recent years, she has received the scientific and technological progress award four times. Her research interests include software engineering and computer network.

Guganggang Geng is a Research Assistant at Computer Network Information Center, Chinese Academy of Sciences. He obtained Ph.D. degree from Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2008. His current research interests include Machine learning, Web Search.

Wanming Luo is an Associate Professor at Computer Network Information Center, Chinese Academy of Sciences. He obtained Ph.D. degree in Computer Science from Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2001. His current research interests include computer networks, performance evaluation, and network security analysis.

Xingfeng Li received the Ph.D. degree from Computer Network Information Center of Chinese Academy of Sciences, Beijing, China, in 2009. Now, he is working at The People's Bank of China. His current research interests include computer networks, system reliability.

Received: December 23, 2010; Accepted: January 20, 2011.