A loss recovery approach for reliable application layer multicast
Xin-Chang Zhang\textsuperscript{a,\,*}, Mei-Hong Yang\textsuperscript{a}, Xiao-Jing Zhu\textsuperscript{b}, Wan-Ming Luo\textsuperscript{c}

\textsuperscript{a} Shandong Key Laboratory of Computer Networks, Shandong Computer Science Center, Jinan 250014, China
\textsuperscript{b} Chongqing Communication College, Chongqing 400035, China
\textsuperscript{c} Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China

\textbf{ARTICLE INFO}

Article history:
Received 22 April 2011
Received in revised form 7 January 2012
Accepted 7 January 2012
Available online 14 January 2012

Keywords:
Application layer multicast
Data loss
Recovery
Retransmission

\textbf{ABSTRACT}

In the application layer multicast (ALM), providing reliable (i.e., lossless) data delivery service is complex. This paper proposes a hierarchical loss recovery solution, called HR, for reliable application layer multicast. HR can be used as an extension to existing tree-based ALM protocols, to provide lossless ALM service. In the HR solution, the group members at the top of the ALM tree are in a recovery plane (i.e., Plane 1), and the members at the bottom of the ALM tree are in another recovery plane (i.e., Plane 2). HR employs a robust and quick approach to recover the losses at the group members in Plane 1, which can avoid the potential losses at the downstream nodes and provide many relatively reliable recovery sources. HR uses a loss recovery approach, which can effectively reduce the link load, to recover the loss found by the group member in Plane 2. In HR, the recovery packet is retransmitted by an area-constrained multicast means, which can recover the loss at one or multiple group members with little recovery diffusion. Through a cooperative and active way, HR can effectively address the error correlation problem of ALM.

© 2012 Elsevier Inc. All rights reserved.

1. Introduction

Multicast is the most efficient communication way for group applications because it saves much bandwidth and greatly reduces the load of the server. Traditionally, multicast functionality is implemented at the IP layer, called IP multicast. However, IP multicast has some drawbacks that are the hurdles to its ubiquitous deployment (Diot et al., 2000). As an alternative of IP multicast, the application layer multicast implements the multicast functionality at the 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. Most existing ALM solutions use tree structure to deliver the multicast data. In the delivery tree of IP multicast, branch nodes (other than the root) are multicast routers. In the ALM tree, branch nodes are dynamic group members instead of steady multicast routers. Consequently, the transport in ALM is relatively unreliable.

Multicast can provide two main types of services: reliable (i.e., lossless) service and loss-tolerant service. In the applications that require reliable services (e.g., the data distribution and multi-party game), the data sources distributes data to all the receivers, and any data loss has to be recovered. In the applications that require loss-tolerant services (e.g., video and audio streaming), it is unnecessary to recover each loss. Providing reliable service based on IP multicast is a challenging topic which has been researched for a long time. In the tree-based application layer multicast, there exists the so-called error correlation problem, i.e., the loss at a group member results in the corresponding losses at the downstream group members. The error correlation problem and the dynamics of group members make the data loss recovery in ALM more complicated.

To improve the reliability of the ALM service, some existing ALM solutions (e.g., Pendarakis et al., 2001; Zhang et al., 2005) use TCP to connect the adjacent nodes. Simon Wong et al. (2004) pointed 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. Jin et al. (2008) further pointed out that the ALM solution based on TCP cannot provide reliable ALM service.

In this paper, we propose a hierarchical loss recovery solution, called HR. The proposed solution can be used as an extension to existing tree-based ALM protocols using UDP, to provide lossless ALM service. Unlike other loss recovery solutions, HR divides the group members into two recovery planes (i.e., Plane 1 and Plane 2) in terms of the ALM tree, and employs different but correlated approaches to recover the losses at the members in different planes. Different from the loss recovery solutions that attempt to reduce
the error correlation in the recovery procedure (e.g., LER (Simon Wong et al., 2004)). HR uses a series of ways to recover the data losses in terms of the error correlation. This paper consists of three main contributions: (1) a hierarchical loss recovery strategy is proposed to obviously improve the whole recovery performance, (2) a special retransmission way (i.e., area-constrained multicast) is presented to recover the loss at one or multiple group members with little recovery diffusion, and (3) a cooperative and active recovery way is proposed to address the error correlation problem without reducing the delivery performance of ALM.

The rest of the paper is organized as follows. In Section 2, we introduce the related work. Section 3 presents the detailed design of HR. We evaluate the performance of HR by analyzing the simulation results in Section 4. Finally, we conclude our work in Section 5.

2. Related work

Application layer multicast is a promising solution to provide the delivery service for the group application, and has been widely researched in the past decade. So far there have been many ALM protocols which address how to use tree structure to deliver the multicast data from the source to all the other group members (e.g., Banerjee et al., 2006a; Francis, 2000; Li and Striegel, 2007; Pendarakis et al., 2001; Struve et al., 2006; Tran et al., 2003; Zhang et al., 2002). Some existing ALM protocols use hierarchical tree building approaches, e.g., ZIGZAG (Tran et al., 2003) and NICE (Zhang et al., 2002). ZIGZAG and NICE each organize the overlay into a hierarchy of clusters, and form the ALM tree based on the hierarchy. Our proposed solution (i.e., HR) also uses the hierarchy strategy. However, HR attempts to use the strategy to effectively recover the data loss in the data distribution (based on the existing ALM tree) instead of building the ALM tree. The hierarchical loss recovery of HR can be seen in Section 3. Note that the loss recovery and delivery tree construction are two relatively independent research topics and that HR is not directly related to the specific approach for building the ALM tree.

So far there have been some studies on the reliability of ALM. For examples, Özkasap (2009) and Jin et al. (2008) presented some surveys of the reliability of ALM. Next we further introduce some typical existing approaches for improving the reliability of ALM.

Banerjee et al. (2006b) proposed a randomized forwarding technique (called PRM) to improve the reliability of ALM. In PRM, each member randomly forwards packets to some extra children (besides its own children) with a certain probability, which can actively recover some potential data loss. Clearly, the randomized forwarding way cannot ensure that all the data losses are recovered. Some multiple-tree ALM approaches have been proposed to provide the delivery service with desirable reliability, e.g., CoopNet (Padmanabhan et al., 2002), SplitStream (Castro et al., 2003), THAG (Tian et al., 2005), and NHAG (Kobayashi et al., 2009). Multiple-tree ALM approaches construct multiple paths between the root and each group member and deliver descriptions using MDC (Alasti et al., 2001; Goyal, 2001). CoopNet uses 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. However, the above multiple-tree ALM approaches cannot provide lossless ALM services.

In ALMI (Pendarakis et al., 2001), 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. When TCP is used, a connection has to be established between two adjacent nodes. Void (Francis, 2000) also presents a similar TCP-based scheme for providing the reliable service. As mentioned in Section 1, TCP-based approaches have some shortcomings on providing the reliable ALM service.

LER (Simon Wong et al., 2004) is one of few available studies on reliable ALM based on UDP. LER employs a lateral retransmission instead of a vertical retransmission from a member’s ancestors. LER randomly divides members into several planes and independently builds an overlay tree in each plane. In each plane, a member 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 member selects some members 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 (Jin et al., 2008). Clustering nearby nodes is a promising approach for building the delivery tree with low end-to-end delay and network traffic. Therefore randomly dividing the members into some planes weakens the above advantage to some extent.

In contrast, the reliable transport over IP multicast (RTIPM for short) has been widely studied for a long time (e.g., Paul et al., 1997; Wonyong et al., 2006). The existing RTIPM protocols are usually based on UDP. In RTIPM, the tree-based ACK and NACK-based mechanisms are widely adopted. The tree-based ACK mechanism arranges the group members into a tree whereby the group member generates ACKs to a parent node, which aggregates those ACKs to its parent in turn. In the NACK-based mechanism (e.g., NORM (Adamson et al., 2004)), the receiver sends a NACK message for every data packet or ADU (application data unit) it discover it did not receive, which reduces the retransmission delay to some extent. Xiao and Birman (2001) presented a randomized RTIPM protocol which improves the robustness of tree-based reliable multicast protocols by diffusing the responsibility of error recovery among all members in a designed group. As mentioned in Section 1, the branch nodes of the ALM delivery tree are common member instead of multicast routers, which brings some new problems (e.g., the error correlation problem) to reliable ALM. Therefore it is unwise for reliable ALM to directly leverage the above approaches proposed to provide reliable transport over IP multicast.

3. HR solution

In this section, we describe the proposed HR solution. Section 3.1 presents an overview of the solution. Section 3.2 explains how to construct the two recovery planes. The loss recovery procedures for the losses at the members in the two recovery planes are described in Section 3.3 and Section 3.4, respectively.

3.1. Overview

As noted in Section 1, HR divides the group members into two recovery planes, i.e., Plane 1 and Plane 2, in terms of the ALM tree. In this paper, the group member in Plane 1 is called core node (CN), and the group member in Plane 2 is called periphery node (PN). The basic design principle of HR is to quickly and robustly recover the losses at CNs, and to recover the losses at PNs with low link load and reasonable recovery delay. The CNs are usually at the top of the ALM tree. Therefore the quick and robust loss recovery for CNs can avoid the potential losses at the downstream nodes and provide many relatively reliable recovery sources for the downstream nodes. Note that the recovery source denotes the node which can retransmit needed recovery packet. HR retransmits the recovery packet by an area-constrained multicast means, and employs a cooperative and active way (including retransmission request suppression, active recovery, etc.) to address the error correlation problem.
In the reliable multicast, the confirmation entity is either packet or ADU, as IETF RFC 2887 \cite{Handley2000} explains. HR uses ADU-level confirmation, which corresponds to ALM in nature. In the loss recovery procedures, HR uses six main types of messages, i.e., NACK1, NACK2, NACK1_N, NACK2_N, NACK2_A, and RECOVERY. NACK1 and NACK2 denote the retransmission request messages sent by CN and PN, respectively. NACK1_N and NACK2_N denote the negative responds to NACK1 and NACK2, which means that the receivers of NACK1 and NACK2 have not the corresponding ADUs in their buffers. NACK2_A is used to suppress sending the NACK2 message. RECOVERY means the recovery packet that contains a retransmitted ADU. Note that HR can distinguish the RECOVERY packet from the normal data packets by some special identification in the application layer data.

In HR, each member employs three types of timers, i.e., T\_NACK1, T\_NACK2, and T\_NACK2\_A. Note that T\_NACK2 and T\_NACK2\_A do not run simultaneously. The time intervals of T\_NACK1, T\_NACK2 and T\_NACK2\_A are denoted by t\_NACK1, t\_NACK2, and t\_NACK2\_A, respectively. Let \( rtt_t(m) \) mean the round trip time between the member \( m \) and the data source, then: \( t\_NACK1 = \xi_1 rtt_t(m) \), \( t\_NACK2 = \xi_2 rtt_t(m) \). Additionally, \( t\_NACK2\_A = \xi_3 t\_NACK2\). Note that \( \xi_1, \xi_2, \) and \( \xi_3 \) are three configuration parameters, which each are larger than 1.

### 3.2. Recovery plane construction and maintenance

In ALM, the data losses at the nodes at the top of the ALM tree have worse influence on the whole delivery reliability than that at the bottom of the ALM tree because the common group members are responsible for forwarding received data packets to their downstream nodes. Therefore it is very helpful for a loss recovery solution to robustly and quickly recover the losses at the top of the ALM tree. Fig. 1 shows the structure of the two recovery planes of HR. From the figure, we can see that some nodes at the top of the ALM tree are selected as core nodes (CNs), and other nodes of the ALM tree are selected as periphery nodes (PNs). Next we introduce how to dynamically divide group members into two recovery planes.

![ALM tree and two recovery planes](image)

**Fig. 1.** The ALM tree and two recovery planes.

Let \( C_x \) denote the set of children of the member \( x \). We introduce a function to indicate the height of the subtree rooted by \( x \), which is denoted by \( h(x) \) and defined by

\[
h(x) = \begin{cases} 
0, & \text{if } |C_x| = 0 \\
\max\{h(c) + 1 | c \in C_x\}, & \text{otherwise}.
\end{cases}
\]

In HR, each group member saves a subtree-weight value for each child of the member. Let the subtree-weight of \( m \)'s child (denoted by \( c \)) be denoted by \( w(m, c) \), then \( w(m, c) = h(c) \). Algorithm 1 presents the solution to obtain the subtree-weight values of the children of group members. In the algorithm, the message \( \text{HeightUpdate}(h(x)) \) is used to inform \( p(x) \) of \( h(x) \), where \( p(x) \) denotes the parent node of \( x \). From the algorithm, we can notice that the subtree-weight values are dynamically assigned in terms of the ALM tree. Clearly, the tree maintenance is an indispensable part of a desirable ALM solution. Through the maintenance, the tree structure can be adjusted when some nodes fail. According to Algorithm 1, we know that the subtree-weight values can be updated when the tree structure changes.

**Algorithm 1.** Get subtree-weight values

1. Initialize: For the newcomer \( r, h(r) = 0 \).
2. When the newcomer \( r \) joins the group, \( r \) sends \( \text{HeightUpdate}(h(r)) \) to \( p(r) \).
3. Once a group member (denoted by \( m \)) receives the \( \text{HeightUpdate}(h(c)) \) message from one of its children (denoted by \( c \)), \( m \) does as follows: If \( c \) is a new child, then \( m \) initializes the subtree-weight value for \( c \) (i.e., \( w(m, c) = 0 \)); Otherwise, \( m \) updates the subtree-weight value of \( c \) (i.e., \( w(m, c) = h(c) \)) and recomputes \( h(m) \). The member \( m \) sends \( \text{HeightUpdate}(h(m)) \) to \( p(m) \) if the value of \( h(m) \) changes.
4. If the member \( m \) deletes an existing child, it recomputes \( h(m) \). Similar to the above step, \( m \) sends \( \text{HeightUpdate}(h(m)) \) to \( p(m) \) if the value of \( h(m) \) changes.

In ALM, from the root to each node, there is one unique loop-free path along the multicast tree. The node list of this path is called root path \cite{Zhang2002}. In the ALM solutions which do not use the root path, each member can easily get its root path through a simple extension. This paper ignores the detail of the above extension, and assumes that each member saves and updates its root path. Let \( r(m) \) denote the root path of the member \( m \), and \( |r(m)| \) mean the number of the nodes in \( r(m) \). We define the following gain function:

\[
\text{chg}(m) = \delta \cdot (h(m)/|r(m)|),
\]

where \( \delta \) is a configure parameter. When the member \( m \) receives a \( \text{HeightUpdate} \) message, it computes \( \text{chg}(m) \) in terms of the latest information (i.e., root path and subtree-weight values of \( m \)'s children). If \( \text{chg}(m) > 1 \), then \( m \) becomes a CN node. Otherwise, \( m \) becomes a PN node. Clearly, the higher the parameter \( \delta \) is, the more CNs are produced. The growth of the numbers of CNs can reduce the recovery delay and improve the robustness of the loss recovery. However, it is unnecessary to produce too much CNs because the losses at PNs be effectively recovered when enough CNs be produced \cite{see Section 3.4}. By default, \( \delta = 1 \).

From the above description, we can notice that HR dynamically divides group members into two recovery planes in terms of the ALM tree. Therefore the two recovery planes are automatically maintained in the case where some nodes fail. According to the loss recovery approaches described in the following two sections, we can notice that the failures of some nodes (except the data source) do not affect the correctness of recovering the losses at other nodes, and that the loss at an active member can always be recovered regardless of the type (i.e., CN or PN) of the member.
3.3. Loss recovery for CNs

In this paper, we say that a node is at the level \( k \) if there are \((k-1)\) upstream nodes in its root path. Note that the root of the ALM tree is at the highest level.

Let \( L^0(m) \) denote a set of nodes and have at most \( \gamma \) elements, where \( \gamma \) is a configuration parameter. Specially, \( L^{(1)}(m) = \{ \text{the treeroot} \} \). In HR, each CN (denoted by \( m \)) at the level \( k \) keeps a recovery vector, which is denoted by \( R_{m,k} \) and defined by

\[
R_{m,k} = (L^{(1)}(m), L^{(2)}(m), \ldots, L^{(v(k))}(m)),
\]

where \( v(k) \) denotes the number of components of \( R_{m,k} \), which is defined by

\[
v(k) = \begin{cases} \log k, & \text{if } \exists x(2^x + 1 = k) \\ \log k + 1, & \text{otherwise} \end{cases}.
\]

Let the set of the candidate recovery sources for the CN \( m \) at level \( k \) be denoted by \( \Delta(m, k) \), then

\[
\Delta(m, k) = \bigcup_{i=1}^{v(k)} L^{(i)}(m).
\]

In HR, the root path of each CN node is periodically multicasted to the group by a special message, called RPATH. Then the CN \( m \) gets \( R_{m,k} \) as the following steps:

- **Step 1**: Assume that the length of \( m \)’s root path (i.e., \( n(m) \)) be \( e \), then \( m \) adds the \( j \text{th} \) \((j = 2^{k-1})\) node in \( n(m) \) into \( L^{(1)}(m) \) if \( |L^{(1)}(m)| < \gamma \). Additionally, \( m \) adds eth node into \( L^{(v(k))}(m) \) if \( 2^r < e < 2^{n+1} \) and \( |L^{(v(k))}(m)| \leq \gamma \). \( R_{m,k} \) is updated if \( m \)’s root path changes. Note that the root and \( m \)’s parent are the first and last nodes in \( m \)’s root path, respectively.

- **Step 2**: When \( m \) receives a RPATH message, it gets the corresponding root path (denoted by \( rp \)). Assume that the length of \( rp \) be \( e' \), then: (1) \( m \) deletes the existing nodes in \( L^{(1)}(m) \) if the nodes are upstream or downstream nodes of the last node of \( rp \), (2) \( m \) adds the \( j \text{th} \) \((j = 2^{k-1})\) node in \( rp \) into set \( L^{(1)}(m) \) if \( |L^{(1)}(m)| \leq \gamma \) and \( 3 \text{)} m \) adds eth node into \( L^{(v(k))}(m) \) if \( 2^{r'} < e' < 2^{n+1} \) and \( |L^{(v(k))}(m)| \leq \gamma \).

**Lemma 1.** There do not exist two nodes (denoted by \( a \) and \( b \)) in \( L^{(1)}(m) \) such that \( a \) is a upstream node of \( b \).

**Proof.** Assume that there exist two nodes (denoted by \( a \) and \( b \)) in \( L^{(1)}(m) \) such that \( a \) is a upstream node of \( b \). Consider the following two cases:

- Case 1: The structure of the ALM tree is steady. In this case, \( a \) always is in the root path of \( b \). From the above computing procedure of \( R_{m,k} \), we know that the two nodes on the same root path cannot be added into \( L^{(1)}(m) \) simultaneously. Thus the above assumption does not hold.

- Case 2: The structure of the ALM tree is unsteady. In this case, the root path of each node is variable. Since two nodes on the same root path cannot be added into \( L^{(1)}(m) \) simultaneously, \( a \) and \( b \) must be added to \( L^{(1)}(m) \) in terms of two different root paths (denoted by \( r_a \) and \( r_b \)). However, \( b \) or \( a \) cannot simultaneously be in \( L^{(1)}(m) \) according to Step 2 in the above procedure. Thus the above assumption does not hold.

Now this lemma has been proven. □

**Algorithm 2.** Recover loss at the CN \( m \) at the level \( k \)

1. Once \( m \) finds that it has lost a correct ADU \( n \), it instantly sends a NACK1 \( n \) message to each node in \( L^{(1)}(m) \) and starts the \( T_{\text{NACK1}} \) timer. \( T_{\text{NACK1}} \), count = \( v(k) \).

2. If \( \text{waitfor}(\text{RECOVERY}(n)) = \text{true} \), then: ADU \( n \) is recovered; \( m \) forwards \( \text{RECOVERY}(n) \) to each of its children if \( m \) has not received ADU \( n \); end the procedure.

3. If \( \text{received}(\text{RECOVERY}(n)) = \text{false} \) or \( m \) receives a NACK1 \( n \) message from each node in \( L^{(1)}(m) \), then: count = \( \text{max}(1, \text{count} - 1) \); \( m \) sends a NACK1 \( n \) message to each node in \( L^{(1)}(m) \); restart \( T_{\text{NACK1}} \); goto Line 2.

**Algorithm 2** explains the loss recovery procedure for the CN node. When the CN \( m \) at the level \( k \) finds that it has lost a correct ADU \( n \), it instantly sends NACK1 \( n \) (i.e., the NACK1 message that requests for the retransmission of ADU \( n \) to each node in \( L^{(1)}(m) \)). If \( m \) receives RECOVERY \( n \) (i.e., the RECOVERY message that carries ADU \( n \)) before \( T_{\text{NACK1}} \) expires, then waitfor(RECOVERY\( n \)) returns true. Otherwise, waitfor(RECOVERY\( n \)) returns false. If \( m \) has not received RECOVERY \( n \) before \( T_{\text{NACK1}} \) expires, then received(RECOVERY\( n \)) = false. The CN node responds to the NACK1 \( n \) message as follows: It sends RECOVERY \( n \) to the sender of NACK1 \( n \) if ADU \( n \) is in its buffer; Otherwise, it sends NACK1 \( n \) to the sender of NACK1 \( n \).

To summarize, HR employs a multi-round procedure to implement the loss recovery triggered by sending the NACK1 message. A recovery round starts when a CN (denoted by \( m \)) sends NACK1 to each node in \( L^{(1)}(m) \), and ends when \( m \) receives the corresponding RECOVERY message or re-sends NACK1 to each node in \( L^{(1)}(m) \).

**Fig. 2.** Loss recovery for the CN \( m \) at level \( k \). \( \gamma = 3, u = v(k) \).

**3.4. Loss recovery for PN**

As noted previously, only PNs can send the NACK2 message. HR also employs a multi-round procedure to implement the loss
recovery triggered by sending the NACK2 message, as Algorithm 3 shows. In this situation, a recovery round starts when a PN sends a NACK2 message to one of the upstream nodes in the corresponding ALM tree, and ends when the PN receives the corresponding ADU (carried by RECOVERY) or sends a new NACK2 message to another upstream node.

Assume that (1) there are \( u \) nodes between the PN \( m \) and the first (i.e., closest) CN node in \( m \)'s root path \( r(m) \), and (2) there are \( v \) CNs in \( r(m) \), then a recovery source choosing function is denoted by \( U_i(m) \) and defined by

\[
U_i(m) = \begin{cases}
    \text{the}(2^{i+1}\text{UD}), & i \leq \tau \\
    \text{the}(u + i - \tau \text{UD}), & \tau + v > i > \tau \\
    \text{thereeroot}, & i > \tau + v 
\end{cases}
\]  

(6)

where \( \log u \leq \tau < 1 + \log v \), and \( UD \) means the upstream node of \( m \).

**Algorithm 3.** Recover loss at the PN \( m \)

1. Once \( m \) finds that it has lost a correct ADU \( n \), it instantly sends a NACK2(n) message to \( U_1(m) \), and starts the \( T_{\text{NACK2}} \) timer \( T_2 \); \( \text{upc} \leftarrow 1 \).
2. If waitfor(RECOVERY(n)) = true, then: ADU \( n \) is recovered; \( m \) forwards RECOVERY(n) to each of its children if \( m \) has not received ADU \( n \); end the procedure.
3. If \( m \) receives a NACK2_A(n) message, then \( m \) cancels \( T_2 \) if it exists; starts or restarts \( T_{\text{NACK2}} \) timer \( T_2 \); forwards the message to its children.
4. If received2(RECOVERY(n)) = false or \( m \) receives a NACK2_N(n), then: \( \text{upc} \leftarrow \text{upc} + 1; m \) sends the NACK2(n) message to \( U_{\text{upc}}(m) \); restart \( T_2 \) or \( T_3 \); goto Line 2.

When a PN (denoted by \( m \)) finds that it has lost a correct ADU \( n \), it instantly sends NACK2(n) (i.e., the NACK2 message that requests for the retransmission of ADU \( n \)) to \( U_1(m) \), to trigger the recovery procedure. When \( U_i(m) \) receives NACK2(n), it sends RECOVERY(n) to \( m \) if ADU \( n \) is in its buffer, or sends NACK2_A(n) to \( m \) if it has not received ADU \( n \) or sends NACK2_N(n) to \( m \) if it has received ADU \( n \) but ADU \( n \) is not in its buffer. If \( m \) has not received RECOVERY(n) before \( T_2 \) or \( T_3 \) expires, then received2(RECOVERY(n)) = false.

Next we explain why the \( T_{\text{NACK2}} \) timer and NACK2_A message are used in the above recovery process. From Algorithm 3, we can notice that a node sends the NACK2 message if and only if the node receives the NACK2 message from one of its downstream nodes and the node has not received the corresponding ADU. If the node, that has sent a NACK2_A(n) message, receives RECOVERY(n), it will forward RECOVERY(n) to its children. By the area-constrained multicast, \( m \) finally receives RECOVERY(n). Therefore we use the \( T_{\text{NACK2}} \) timer and NACK2_A message to suppress unnecessary retransmission requests. Additionally, the NACK2 suppression procedure can effectively address the NACK2 explosion problem. Through the above cooperation and active recovery way, the error correlation problem is further alleviated.

From Eq. (6), we can see that PN's recovery source choosing procedure uses different functions. In the first phase, an exponent function is used to choose the recovery source, to reduce the worst recovery delay. The main reason of using exponent function is to provide better capability of local recovery. In the second phase, a linear function is used to choose the next recovery source. From Eq. (6), we also can notice that the first recovery source, denoted by \( r \), of the second phase is a CN node, which means each upstream node of \( r \) is also a CN node. As noted above, the loss at a CN node can be quickly recovered. Therefore we use the linear function to choose the next recovery source after a CN node is selected as the recovery source. Note that the tree root is the final candidate recovery source for any node (PN or CN).

4. Experiment results

We used the BRITE Generator (Medina et al., 2001) to generate a 5000-node power-law network topology. Each node in the above topology represented a router, and the average fanout degree of router nodes was about 3. Additionally, we generated 1000 nodes as the member hosts and a node as the server, which were connected to router nodes randomly. Note that each router node connected at most one host node. In the simulation experiments, each member host lost each correct packet with probability of a random value in an interval (denoted by \( \mu \)). By default, \( \mu = [0.01, 0.2] \). The experiments used the following parameter configurations: \( \gamma = 3 \), \( \xi_1 = \xi_2 = \xi_3 = 2 \). In these experiments, we applied HR to NICE (Zhang et al., 2002), to form a reliable ALM solution called HR-NICE. Specifically, we used NICE to build the ALM tree and used HR to recover the data loss. We simulated the related protocols with NS-2 (i.e., The Network Simulator version 2). Except for some results from a single run (i.e., Figs. 3–6), data points in the following graphs represent averages over 100 runs with 95% confidence interval. Note that the group members joined the group session in a random order in each run.

We first used HR-NICE and LER to build the ALM trees with 1000 members, respectively. Fig. 3 gives a comparison of the link load of HR-NICE and LER. In the figure, the main link denotes the link that connects two different router nodes, and the related main link

![Fig. 3. Link load of HR-NICE and LER.](image-url)

**Fig. 4.** Mean recovery distance ratio of HR-NICE.
represents the main link that transports the multicast data packets. Physical link load (stress) means the number of identical copies of a packet that traverse a link. As noted previously, LER randomly divides hosts into several independent planes, which makes some nearby nodes be located in different planes. Therefore LER brought higher link load than HR-NICE in our experiments, as Fig. 3 shows.

We use recovery distance ratio (RDR) to represent the ratio of the length of the recovery path to the length of the path from the source to the corresponding member. The RDR of recovering lost ADU n at the member m is denoted by RDR(m, n) and defined by

\[ RDR(m, n) = \frac{h(r(m, n), n)}{h(s, m)}, \tag{7} \]

where \( r(m, n) \) means the node that successfully retransmits ADU n to m, \( h(k, m) \) indicates the number of router nodes in the path from k to m, and s denotes the data source. If \( RDR(m, n) < 1 \), then the corresponding recovery source is closer to m than the data source. The lower the value of \( RDR(m, n) \) is, the closer the corresponding recovery source is. According to the above description, we can notice that the RDR metrics can evaluate the degree of localization of the loss recovery to some extent. Note that the above localization degree is a relative measurement using the loss recovery, which gets the recovery packet from the data source, as a reference.

Fig. 4 depicts the mean recovery distance ratio of HR-NICE. Note that the interval \( \mu \) in this experiment is \([0.05, 0.2]\). From the figure, we can see that the recovery distance ratios of HR-NICE are low in most scenarios, which means that HR-NICE can well keep the local recovery. The high values in Fig. 4 can be explained as follows: The losses at some members could not be recovered in the first few recovery rounds (see Section 3.3 and Section 3.4) because of the losses of related messages (i.e., NACK1, NACK2 and RECOVERY messages), which resulted in that these members finally obtained corresponding recovery packets from some remote nodes. Note that the hosts usually lose the packets, which are sent to them, with lower probability in real networks than in our simulation.

Fig. 5 shows mean number of loss recovery rounds of HR-NICE. From the results, we can notice that few loss recovery rounds were needed in our experiments. In some degree, the above results show that HR-NICE can quickly recover the data losses. We attribute the desirable performance to the NACK2 suppression, quick recovery source searching procedure and the area-constrained multicast.

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 nodes because of the receipt of a NACK1 message, then the packet was identified by type 1. Otherwise, the recovery packet was identified by type 2. Fig. 6 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 loss recovery launched by the nodes in Plane 1 plays an important role.

Figs. 7 and 8 illustrate the loss recovery delay and load of HR-NICE and LER in 10 groups with different group sizes. In these experiments, we used two planes and five planes to build LR multicasts, respectively. In Fig. 7, \( L/H \) recovery delay ratio means the ratio of average recovery delay of LER to that of HR-NICE. Let \( l(n) \) mean the number of links that transport the recovery packet n, and \( R(A) \) denote the set of recovery packets in the solution A. Then \( L/H \) recovery load ratio (denoted by LHRLR) is defined by

\[ LHRLR = \frac{\sum_{n \in R(LER)} l(n)}{\sum_{n \in R(HR-NICE)} l(n)}. \tag{8} \]

According to the above definition, we can notice that LHRLR can evaluate the load of the loss recovery in some degree. From Fig. 7, we can see that \( L/H \) recovery delay ratio is more than 1 in each group, and that \( L/H \) recovery delay ratio increases as the group size grows. We attribute the desirable performance to quick recovery source searching procedure and the active recovery way. Fig. 8 tells us that \( L/H \) recovery load ratio also is more than 1 in each group and has an increasing trend. As noted previously, HR-NICE employs NACK2_A to suppress the retransmission requests, and uses the area-constrained multicast to retransmit the


Xin-Chang Zhang received the Ph.D. degree from the Computer Network Information Center of Chinese Academy of Sciences. Presently, 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 30 papers in research journals and international conference proceedings in these areas.

Mei-Hong Yang is the associate technology officer and professor of the Shandong Computer Science Center of Shandong Academy of Sciences, China. She received the M.S. degrees from the Shandong University, China. In the recent years, she has received the scientific and technological progress award four times. Her research interests include software engineering and computer networks.

Xiao-Jing Zhu is a lecturer at the electronic information engineering department, Chengqing Communication College, China. She obtained Master degree in Information and Communication Engineering from Tsinghua University in 2010. Her current research interests include computer networks, modeling, and signal processing.

Wan-Ming Luo obtained Ph.D. degree in Computer Science from the Institute of Computing Technology, Chinese Academy of Sciences in 2001. Now, he is an Associate Professor at the Computer Network Information Center, Chinese Academy of Sciences. His current research interests include computer networks and mobile computing.

Fig. 8. Loss recovery load of HR-NICE and LER.

recovery packets along the ALM tree. Therefore HR-NICE produced low recovery load than LER in our experiments.

5. Conclusions

In ALM, the reliability of the data delivery is low because dynamic group members forward received data packets to other members. In this paper, we proposed a hierarchical loss recovery solution HR, as an extension to existing tree-based ALM protocols, to provide lossless ALM service. HR divides the group members into two recovery planes (i.e., Plane 1 and Plane 2) in terms of the ALM tree. HR employs different but correlated solutions to effectively recover the data losses. Specially, the losses found by nodes in Plane 1 are robustly and quickly recovered, and the losses found by nodes in Plane 2 are recovered with low link load and reasonable recovery delay. The quick and robust loss recovery for the losses at nodes in Plane 1 can avoid potential data losses at the bottom of the ALM tree and provide reliable recovery sources. HR retransmits the recovery packet by an area-constrained multicast means, which recover the loss at one or multiple members with little recovery diffusion. Through a cooperative and active recovery way, the error correlation problem is addressed without reducing the delivery performance of ALM.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant No. 61070039, the National Basic Research Program of China under Grant No. 2009CB320502 and the Outstanding Young and Middle-aged Scholars Foundation of Shandong Province (in China) under Grant No. 2008BS01019 and No. BS2011DX033. The authors would like to thank the executive editor and anonymous referees for their constructive comments on the revisions of this paper.

References

