Subgroup-based Source Recovery or Local Recovery for Reliable Multicasting

Danyang Zhang*
New Mexico Highlands University, Las Vegas, NM 87701, USA

Sibabrata Ray†
University of Alabama, Tuscaloosa, AL 35487, USA

Rajgopal Kannan† and S. Sitharama Iyengar†
Louisiana State University, Baton Rouge, LA 70802, USA

Abstract

A traditional approach to design scalable reliable multicast algorithms is using proxies to recover lost packets. The proxies store the packets from the source and retransmit them to the requesting end receivers. Proxy placement in a multicast tree is a well-researched problem. However, given that the proxies require significant storage and processing resources, it is not possible to place them in arbitrary locations. Therefore, one may have to be satisfied with suboptimal proxy placement. Clearly, suboptimal proxy placement may waste significant bandwidth for recovery purposes. In addition, many proxy placement algorithms depend on the knowledge of per link packet loss probabilities, which are costly to estimate and change over time. In this paper, we propose a mechanism to alleviate the problem of suboptimal proxy placement.

The usual approach to recover a packet to the receivers under a proxy is either to multicast the packet to all receivers or to unicast the packet to the receivers who lost it. We propose to partition the receivers under the proxy to several subgroups. The proxy multicasts the lost packet to the subgroup from which a NACK (negative acknowledgement) has been received. We designed two algorithms, i.e., first-fit and all-or-one to compute the subgroups.

Our recovery method requires significantly less recovery bandwidth, which is a problem for sub-optimal proxy placement. In addition, our method does not depend on the knowledge of per link packet loss probabilities and needs very little network resources. Our method may also be used for source recovery (for small multicast groups or if no proxy is available).

Keywords: Reliable multicast, distributed algorithms, source recovery, local recovery, first-fit, all-or-one.

1 Introduction

Reliable multicast has drawn considerable attention during recent years. Many researchers have proposed various scalable and efficient reliable multicast schemes. These proposed recovery schemes can be broadly classified into two categories, i.e., recovery-without-retransmission (using FEC) schemes and recovery-with-retransmission schemes.

In recovery-without-retransmission schemes, FEC (Forward Error Correction) is the representative method [5, 14, 15]. FEC is a technique that proactively transmits redundant packets together with regular data packets for recovery purposes. FEC improves multicast reliability without (acknowledgement) ACK/NACK implosion problem. However, the implementation of FEC is quite costly. FEC wastes bandwidth by sending many (sometimes unnecessary) duplicate packets for recovery. In addition, FEC cannot guarantee fully reliable multicast, which is not appropriate for applications like software distribution, internet stock quoting, etc. Therefore, in many cases, recovery-with-retransmission schemes are necessary.

The recovery-with-retransmission schemes can be divided into three classes with respect to the responsibility of retransmission. These three categories are source-based, server/proxy-based and peer-based recovery schemes.

In source-based recoveries, the source exclusively retransmits the lost packets to the requesting receivers. This mechanism guarantees that one recovery attempt is enough for each request, and thus reduces the overhead incurred by failed recovery attempts. However, it needs to handle ACK/NACK implosion problems. In [18], protocols A, N1 and N2 belong to this category.

Server/proxy-based recovery schemes usually partition group members into local groups hierarchically and/or geographically and allocate one recovery server/proxy for each local group to detect recovery requests and recover the lost packets. RMTP [9], IRMA [6], MTCP [13], RMX [1], ARM [7], HRM (hierarchical reliable multicast) [4], et al., fall in this class. In peer-based recovery schemes, it is up to the receivers to detect packet loss and send the request to other receivers [2, 8, 16] or third-party [17] which will either retransmit the packet if they have received it or send the request further otherwise.

The existing source-based/proxy-based recovery schemes often perform the recovery by retransmitting the lost packet to all group members/ group members in the locality (recovery by multicast) or by sending the lost packet to the recipients not receiving the packets one-by-one (recovery by unicast). None

* Dept. of Computer and Mathematical Sciences.
† Dept. of Computer Science.

of the recovery methods are bandwidth efficient. In this paper we propose a new bandwidth-efficient recovery mechanism for source-based/proxy-based recovery schemes.

The remainder of this paper is organized as follows. In the next section we introduce some existing server/proxy-based recovery schemes and present the performance benefits of our schemes. In Section 3, we formulate the subgrouping problem and prove that computing optimal subgroups is NP-hard. In Section 4, we give an example on why optimal subgrouping problem is time-costly and present two heuristics, i.e., *First-fit* and *All-or-one*, for constructing the subgroups. In Section 5, we present and analyze our simulation results. Our conclusion and future work are described in Section 6.

2 Survey On Some Existing Recovery Schemes

In existing server/proxy-based recovery schemes, recovery servers/proxies and downstream routers from recovery servers/proxies collect NACKs from clients under it. Simultaneously, it also updates its subscription bitmap that indicates which outgoing links subscribed for which repair packets. This subscription bitmap is usually stored in routers soft-state cache [7]. By looking up the subscription information, the local recovery server is able to retransmit the repair packet only to those requesting links. RMTP [9], IRMA [6], MTCP [13], RMX [1] and ARM [7] explicitly state this local retransmission technique. Among them, ARM [7] describes this problem in detail and also points out that if no cache is available the router just retransmits the repair down all links.

However, this subscription information in cache is hard or costly to maintain since whenever a NACK or a repair comes, the cache of each router on the way from the requesting client to the recovery server needs to be updated. Further, the router caching memory is always at a premium and this technique may occupy a large amount of router memory given that every router (not just the proxy) in the multicast tree has to remember per packet loss information. Such per packet loss information may be very large for high-speed networks as packet loss usually occurs due to network congestion and therefore temporally correlated. In addition, when the cache is flushed, all the subscription information is gone, then the recovery server can only multicast the repair to the whole local group, which may cost too much extra bandwidth.

HRM [4] organizes the receivers into subgroups. A proxy is assigned to each subgroup to detect packet loss and recover lost packets. Reference [4] gives algorithms to compute the optimal placement of proxies within a multicast group. While HRM presents an interesting body of theory, it suffers from several practical problems.

1) HRM strongly depends on the loss characteristics of the underlying multicast tree, i.e., it needs to know the per-link loss probability for computing optimal or approximate placement of proxies. But per-link loss probability is difficult to estimate. Loss probability on each link is a dynamic value. It changes from time to time according to the network traffic or the link load. Further, estimating per-link loss probability requires sending several packets over every link. Thus, per link loss-probabilities may not be computed (even approximately) without incurring significant overhead.
2) Proxies require significant resources (memory, bandwidth and processing power) to perform the recovery for the group under it. Therefore, arbitrary members of a multicast group may not serve as a proxy. Restricted placement of proxies may lead to degraded performance.
3) In addition, if a proxy in HRM receives recovery request(s) from one or more of its subgroup members and this proxy has the repair packet(s), it will either multicast the repairs to the whole subgroup or unicast those requesting receivers respectively. By multicasting, it may cost extra bandwidth since the loss is normally a partial loss. By unicast, it also wastes bandwidth by transmitting the repair packet on the same link many times.
4) The proxy placement algorithms proposed in [4] are of high complexity and are not amenable to distributed implementation.

HRM does not discuss how proxies can recover lost packets to achieve minimum recovery latency in the case that the proxies also lost the packets, while our paper [20] proposes a distributed algorithm for proxies to recover lost packets with minimal delay and low bandwidth usage, which can be combined with this paper to form a two-tier reliable multicast recovery scheme.

In this paper we propose a recovery mechanism for source/proxy based recovery schemes free from the drawbacks of the existing recovery mechanisms. Our proposed scheme partitions the receivers under a proxy into several static subgroups. A NACK from a receiver causes the proxy to retransmit the lost packet to the subgroup containing the sender of the NACK. As we shall see later, the all-or-one heuristic ensures that no router (except possibly the proxies) stores any additional information (per packet or per subgroup) beyond the original multicast address entry in its routing table. Such static subgroup based recovery schemes do not incur any overhead to form new dynamic multicast groups for recovery as happens for [1, 6, 7, 9, 13]. Our schemes require less recovery bandwidth in comparison to unicast or multicast based recovery. The subgrouping schemes alleviate the extra overhead that may be incurred by suboptimal proxy placement. Further, our subgrouping schemes do not require the knowledge of exact per link loss probabilities as HRM [4] or other optimal proxy placement algorithms do.

In this paper we also prove that partitioning group members into subgroups such that the bandwidth usage for recovery is minimized is a NP-hard problem. Subsequently, we propose two heuristics; First-fit and All-or-one to achieve lower bandwidth usage than simply multicast or unicast the repair packets. Furthermore, First-fit costs little router caching memory with the fact that only the recovery servers need to store and update per packet loss information and all other intermediate routers just remember the subgrouping information which is obtained at multicast tree construction phase. All-or-one is more significant in that it does not need any router memory and can still achieve low bandwidth usage. Our simulation results in Section 4 support these conclusions.

3 Subgrouping Scheme

3.1 NP-Hardness of Optimal Subgrouping Problem

Consider a multicast tree rooted at source (or the recovery server, see Figure 1). The clients (receivers or lower-level recovery servers) are the leaves of the tree. We partition the leaves in *k* subgroups (reference to Figure 2). If a NACK from a client is received, the repair packet is transmitted to the whole subgroup from where the NACK was sent. It is to be noted that the packet loss in multicast session is correlated in that if a node does not receive a packet, then all the receivers downstream to this node in the multicast tree also lose this packet. Its neighboring nodes also may not receive this packet in a high probability. As our scheme does not maintain per packet loss information on each router, it is likely to save router memory.

However, the subgroups in our scheme are relatively static, i.e., computed during the original multicast tree construction or recomputed only when there are changes in the multicast tree

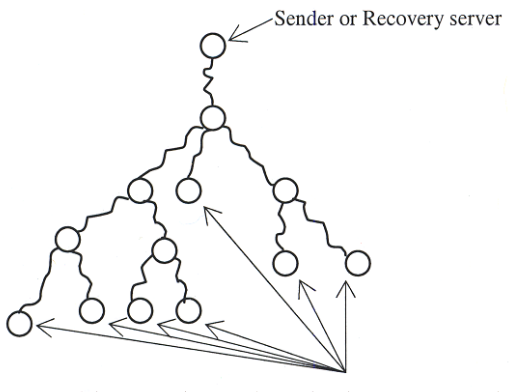

Sender or Recovery server
Clients (receivers or lower-level recovery servers)

Figure 1: A multicast network tree (wavy line denotes consecutive links)

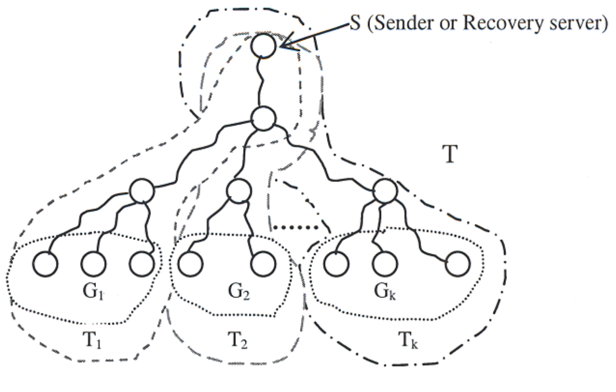


Figure 2: Subgroups and subtrees in a general case

topology. Hence, it may result in some wastage of bandwidth. The performance of our scheme depends on the subgroup computation. In this paper we discuss algorithmic aspects of subgroup computation.

The next subsection derives an objective function to measure the merit of a subgrouping scheme.

3.1.1 Objective Function for Subgrouping Scheme. We start our discussion from a simple case (Figure 3). The sender or recovery server S can send packets to clients X and Y through the intermediate router R. The paths SR, RX and RY consist of a, b and c consecutive links respectively. We also suppose the probability that a packet will be lost in a link is p.

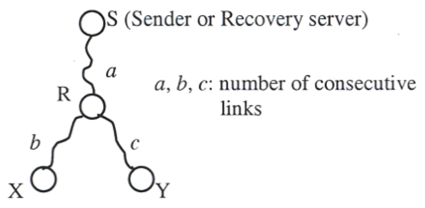

S (Sender or Recovery server)
a
R
a, b, c: number of consecutive links
b
c
X
Y

Figure 3: A simple multicast network

Now let the sender or recovery server S send a packet to X and Y through R. Owing to the link failure in SR, RX or RY, X or Y or both of them may not receive the packet, thus require S to resend it. Here we have two strategies that we can adopt for retransmission.

Strategy 1: S will always retransmit the packet to both X and Y even if one of them lost the packet.

Strategy 2: S will individually retransmit the packet to any recipient that has not received the packet.

Apparently, Strategy 1 will introduce link wastage when only one of them loses the packet. In this case, an extra packet has to traverse the connection RX or RY, thus wasting the bandwidth. On the other hand, for Strategy 2, the bandwidth wastage will occur if neither X nor Y received the packet - the same packet has to pass the connection SR twice.

Let A be the event that X has not received a packet and let B be the event that Y has not received a packet. Thus, the conditional probability that A occurs under the condition that at least one recipient has not received the packet is:

$$P(A \mid A \cup B) = \frac{P(A)}{P(A \cup B)} = \frac{1-(1-p)^{a+b}}{1-(1-p)^{a+b+c}} \qquad (1)$$

While (1) gives an accurate estimate of the conditional probability, it is harder to handle. However, it is known that the per link loss probability for Internet is not very high [11]. According to the statistics of Internet performance in PingER project [11], 95 percent or more hosts in Internet just endure with <u>at most</u> 10 percent or so packet loss. We have shown that the optimal subgrouping may be computed if we ignore p^i (p is the per link loss probability) for $i \geq 2$, without knowing exact per link loss probabilities.

Therefore,

$$P(A \mid A \cup B) \approx \frac{1-[1-(a+b)p]}{1-[1-(a+b+c)p]} = \frac{a+b}{a+b+c} \qquad (2)$$

as for $p^i \to 0$ where $i \geq 2$.

Thus for Strategy 1 (multicast), the cost of recovery is $C_1 = a+b+c$, while for Strategy 2 (unicast), the cost of recovery is $C_2 = (a+b)P(A \mid A \cup B) + (a+c)P(B \mid A \cup B)$ $= \dfrac{(a+b)^2 + (a+c)^2}{a+b+c}$. Obviously, if $C_2 > C_1$, we should use multicast recovery, that is, if $a^2 > 2bc$, X and Y should be put into one subgroup, otherwise not. So far we get the criterion for putting two nodes into one subgroup. The results obtained here may be generalized as follows:

Consider a multicast tree T in Figure 2. The clients (leaf nodes) have been partitioned into k subgroups, $G_1, G_2, \ldots, G_k$. Let T_i be the subtree of T such that the leaves of T_i is G_i for $i = 1, \cdots, k$. Note that any two subtrees may share some internal nodes (including S) and links. Let e_i be the number of links in T_i and e be the number of links in T. Further, assume that a packet consumes one unit of bandwidth to traverse over a link. Let B be the expected bandwidth requirement to recover a loss using the subgrouping scheme.

Lemma 1. $B \approx \sum_{i=1}^{k} \frac{e_i^2}{e}$ as for $p^i \to 0$ where p is per-link loss probability and $i \geq 2$.

Proof. Let A_i be the event that at least one member of G_i has not received a packet. Let $A = A_1 \cup \cdots \cup A_k$, then $B = \sum_{i=1}^{k} e_i P(A_i / A)$, and $P(A_i / A) = \dfrac{1-(1-p)^{e_i}}{1-(1-p)^e} \approx \dfrac{e_i}{e}$ as for $p^i \to 0$ where I ≥ 2. Hence the proof.

From Lemma 1, we know that the optimal subgrouping may be obtained by partitioning the leaves of the multicast tree into $G_1, G_2, \ldots, G_k$, such that $\sum_{i=1}^{k} e_i^2$ is minimized.

We shall prove that the optimal subgroup construction problem is NP-hard even for degree bounded trees. We shall prove it using trees shown in Figures 4 and 13 (refer to Appendix).

Note that the number of partitions in the optimal subgrouping scheme is not fixed. Before we prove the NP-hardness theorem, we prove the following monotonicity lemma for the tree in Figure 4. We call the tree in Figure 4 a generalized star tree.

3.1.2 Monotonicity Property. In Figure 4 S is

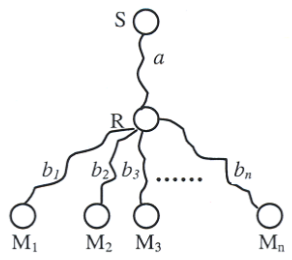


Figure 4: A generalized star tree

source/recovery server, R is an intermediate router and $M_1, \ldots, M_n$ are clients. The path from S to R consists of a links and disjoint paths from R to M_i consist of b_i links, i=1, ..., n.

Lemma 2 (Monotonicity property). Consider two generalized star trees T_1 and T_2. The S to R path in T_1 and T_2 contains a and $a+\varepsilon(\varepsilon \in Z^+)$ links respectively. T_1 and T_2 are identical in all other respects. If the optimal grouping in T_1 and T_2 has k_1 and k_2 partitions respectively, then $k_1 \geq k_2$.

Proof. Refer to the Appendix.

3.1.3 Bounded Number of Subgroups.

Lemma 3. For $a = 0$, $k_{opt} = n$; and for $a = \sum_{i=1}^{n} b_i$, $k_{opt} = 1$.

Proof. Refer to the Appendix.

3.1.4 NP-Hardness of Optimal Subgrouping Problem

Optimal Subgrouping Problem. Given finite set $G = \{b_1, {}_2, \cdots, b_n\}$, positive integer a. Partition G into k disjoint sets (or subgroups) $G_i (i = 1, 2, \cdots, k)$ such that $ka^2 + \sum_{i=1}^{k} \left(\sum_{b_j \in G_i} b_j \right)^2$ is minimized, where $k, n, a, b_j \in Z^+$ for $1 \leq j \leq n$ and $1 \leq k \leq n$. We shall prove the *OSG (Optimal SubGrouping)* problem is NP-hard by reduction from *MSS (Minimum Sum of Squares)* problem. As shown in [3], *MSS* is NP-hard in a strong sense.

Theorem 1. *OSG* problem is NP-hard in ordinary sense.

Proof. Refer to the Appendix.

OSG problem remains NP-hard even if the multicast tree is a degree bounded tree. We prove the result for binary trees.

Theorem 2. *OSG* problem is NP-hard for a degree bounded multicast tree.

Proof. Refer to the Appendix.

3.2 Sub-optimal Subgrouping Schemes

Since OSG (Optimal Sub-Grouping) problem is NP-hard, sub-optimal subgrouping algorithms need to be carefully designed such that the average bandwidth used for recovering a lost packet is low and the network resource involved in recovering lost packets is not much. In the next section, two sub-optimal subgrouping algorithms are presented under the assumptions that per-link loss probability in a network is low and the multicast group does not change frequently.

4 Heuristics For Constructing Subgroups

According to monotonicity property, if the length of common path is increased, the number of subgroups for optimal subgroup-based scheme will not be decreased. However, this does not prevent the subgroups from reorganizing to obtain the minimal cost of recovery. For example, refer to two trees T_1 and T_2 in Figures 5 and 6, respectively as follows,

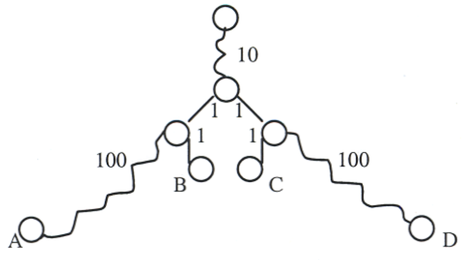


Figure 5: A special multicast tree T_1

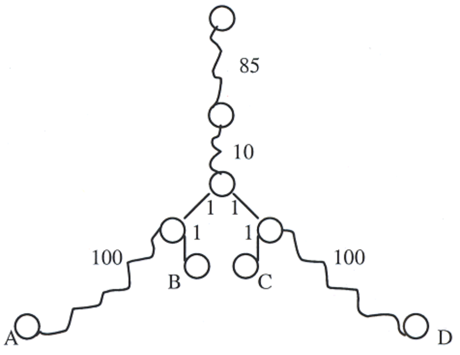


Figure 6: A special multicast tree T_2

For tree T_1, the optimal subgrouping scheme is $G_1 = \{A\}$, $G_2 = \{B,C\}$, $G_3 = \{D\}$. For tree T_2, the length of common path is increased by 85 and the optimal subgrouping scheme is $G_1 = \{A,B\}$, $G_2 = \{C,D\}$. B and C are in the same subgroup in T_1, but are reorganized into different subgroups in T_2 to achieve the minimal cost of recovery. This fact complicates the computation of optimal subgroups and may be one reason that *OSG* problem is NP-hard. However, to simplify computing subgroups, we assume that once two recipient nodes are put into one subgroup, they will never be separated later. Based on this assumption, our heuristic adopts a simple merging scheme called *First-fit heuristic*.

4.1 First-Fit Heuristic

The basic idea of *First-fit* is for each recipient; sequentially check whether any of the other recipients can be combined with this recipient into one subgroup. When all the recipients have been traversed, repeat this procedure on remaining ungrouped recipients.

The criteria used for subgrouping has been provided and proven in Subsection 2.1, that is, if $a^2 > 2bc$, then put X and Y into the same subgroup, otherwise not. The pseudo-code for this algorithm is as follows: (Our discussion is based on Figure 4. Note that a, b_i, b_j are variables and their values need to be adjusted accordingly in line 8 of the following algorithm.),

```
Algorithm 1: First-fit
1.  for $i$=1 to $n-1$ do
2.  if      $M_i$ not yet be put into a subgroup
3.  then    $G_i \leftarrow \{M_i\}$  // $M_i$ is put into a subgroup $G_i$
4.  $cost\_G_i \leftarrow b_i$
5.           if $a^2 > 2\times cost\_G_i$
6.           then for $j=i+1$ to $n$ do
7.                 if   $M_j$ also not yet be put into a group
8.                 then  if  $a^2 > 2\times cost\_G_i \times b_j$
9.                             then  $G_i \leftarrow \{M_j\}\cup G_i$
10.                                 $cost\_G_i \leftarrow cost\_G_i + b_j$
11.                      endif
12.                            if $a^2 \le 2\times cost\_G_i$ then break
13.                            endif
14.                endif
15.          enddo
16.      endif
17.    endif
18. enddo
19. if      $M_n$ not yet be put into a group
20. then    $G_n \leftarrow \{M_n\}$
21. endif
```

The above *First-fit* algorithm may be embedded into the real-world multicast protocol so that the subgroups can be constructed at the same time when the original multicast tree or local groups are formed. This saves overhead for network transmission and avoids intolerable cost of constructing subgroups in the case of retransmission.

4.2 All-or-One Heuristic

It is to be noted that if we use first-fit, some intermediate routers will have to participate in multiple subgroups. Suppose $l_1,\ldots,l_r$ are outgoing links from an intermediate router R. If the proxy sends the recovery packet to subgroup M_i, the router R may have to send the recovery packet along some subset of $\{l_1,\ldots,l_r\}$, whereas, for subgroup M_j, R may need to forward the packet to some other subset of $\{l_1,\ldots,l_r\}$. Therefore, in the worst case, most intermediate routers will have to remember most subgroups' forwarding information. Given that a router may have to support many multicast and unicast connections, these additional costs may add-up quickly. In fact, it is well recognized that the router memory consumed to store flow-state is a scarce resource [19]. Here we propose an alternative grouping scheme to solve the problem.

In the new grouping scheme (called all-or-one), all subtrees rooted at an intermediate node must belong to the same subgroup or no two of the subtrees may belong to the same subgroup. Therefore, an intermediate router either forwards the repair packet to all outgoing links in the original multicast tree or it forwards it to exactly one outgoing link. Figure 7 illustrates this method.

In Figure 7, all intermediate routers (R_1, R_2, R_3, R_4, R_5) store only the original multicast group address and forwarding information. Suppose S wants to send a repair packet to subgroup M_2. The proxy S will use a feature of IP routing called loose source routing [12] to achieve this. The loose source routing

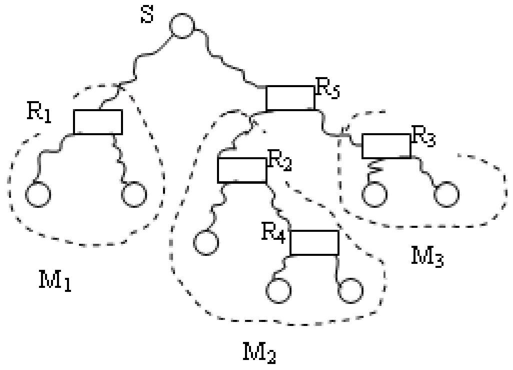


Figure 7: An example for all-or-one subgrouping

allows a sender to route a packet through a set of routers. The proxy will set the destination address of the repair packet to the original multicast address and then route it through R_2 using loose source routing. In this case, the repair packet will arrive at R_2 first. Then R_2 (and other downstream routers) will treat this repair packet as ordinary multicast packet. Note that the subgroups are subtrees rooted at R_1, R_2 and R_3. The routers R_1, R_2 and R_3 are called *subgroup anchors*. The implementation detail is discussed later. In the complexity analysis we will see that it is possible for the proxy to avoid storing subgroup and receiver information.

4.2.1 Distributed Algorithm for All-or-One Scheme. The all-or-one grouping process starts immediately after the multicast tree is constructed and the proxies are in place. There are numerous proposals about proxy placement in literature. Here we concentrate on further grouping under each proxy for efficient local recovery.

The proxy initiates the all-or-one subgoruping algorithm by multicasting an *init* packet with *count* field set to 0. The intermediate routers receive *init* packet, then add *count* by 1 and save this *count* value in its caching memory and multicast this modified *init* packet. When each receiver receives *init* packet, it creates a *reply* packet with *link-count* field set to 0, *group-flag* field set to 1 and *group-root* field set to its own IP address. Then this receiver sends back the *reply* packet to the proxy along the multicast tree. Suppose intermediate routers receive *reply* packets from r outgoing links with link counts $l_1,\dots,l_r$, group flags $g_1,\dots,g_r$ and group roots $I_1,\dots,I_r$. It works as follows on intermediate routers.

```
If ((all $g_i$'s are 1) and
  $(count + r + l_1 + \cdots + l_r)^2 < \sum_{i=1}^{r}(count + 1 + l_i)^2$ )
  then    send reply packet with link-count=$r+l_1+\dots+l_r$,
          group-flag=1 and group-roots=own IP
                      address;
  else    send reply packets with link-count=$r+l_1+\dots+l_r$,
  group-flag=0 and group-roots=union($I_1,\dots,I_r$);
EndIf
```

Once the proxy gets reply packets from all outgoing links, it merges the *group-root* lists to form groups and sets its own *group-flag* to 0 or 1 depending on the grouping results. If the *group-flag* is 0, the proxy sends the recovery packet to only one outgoing link at a time. Otherwise the proxy transmits the repair packet to all the outgoing links simultaneously.

4.2.2 All-or-One Requires No Additional Router Memory. As discussed earlier, existing proxy-based recovery schemes require every intermediate router from the requesting receiver to the proxy to store per packet loss information in its cache for recovery purpose. One drawback is that this occupies a large amount of router memory. Another is when the cache is flushed, every intermediate router has no choice but to transmit the packet to all the outgoing links, which wastes quite an amount of bandwidth. Our simulation results in the next section show that all-or-one scheme can save significant amounts of bandwidth for recovery than simply multicast or unicast recovery. In addition, in all-or-one, every intermediate router does not need to remember per packet loss information to retransmit the packet. The proxy uses loose source routing to retransmit the repair packets to each subgroup. For example, in Figure 5, the proxy S can send the repair packet to all the receivers in subgroup M_3 by sending the repair packet to the original multicast group address through M_3's subgroup root R_3.

This can be further explained as follows.

Figure 8 shows part of the Internet datagram header that corresponds to the loose source and record route (LSRR) option [12]. LSRR provides a means for the source to supply routing information to be used in forwarding the datagram to the destination. That is, if the packet reaches the destination address and the pointer is greater than the length, the routing is based on the destination address field. If the address in destination address field has been reached and the pointer is not greater than the length, the next address in the source route (route data) replaces the address in the destination address field and the pointer is increased by four.

Therefore, in Figure 7, in order for S to send repair packets

```
+--------+--------+--------+---------//--------+
|10000011| length | pointer|     route data    |
+--------+--------+--------+---------//--------+
 Type=131
```

Figure 8: LSRR (loose source route record) header

to subgroup M_3, S creates the datagram with the *destination address* set to be *group-root* R_3's address, the first address in the *source route* (*route data*) set to be the original multicast group address and the value of *pointer* less than the value of *length* and greater than the value of *length*−4. In this way, when this datagram reaches R_3, since the *pointer* is less than the *length*, the original multicast group address replaces the *destination address* and the *pointer* is increased by four (thus the *pointer* is greater than the *length*, the routing is to be based on the destination field). This repair packet is therefore transmitted to all the receivers underneath R_3 as a regular multicast data packet. All-or-one implements retransmission without costing intermediate routers caching memory.

4.3 Time and Space Complexity Analysis

For a n-client multicast tree, assume the height of this multicast tree is h, then the running time of all-or-one in a single machine is $O(n)$. If we embed our algorithm into a reliable multicast protocol in such a way that the nodes in this multicast tree calculate the subgroups in parallel, then this distributed version runs in $O(h)$ time since all the clients calculate and report data to their upper-level nodes in parallel.

The intermediate routers do not need to store any extra information. The proxy may or may not have to store any information about the receivers/groups under it depending on the implementation. If the receivers store the subgroup anchor information, and forwards that information to the proxy in the NACK packets, then the proxy does not need to store any subgroup and receiver information. However, this will need one packet/link cost during the subgroup computation so that every receiver knows its subgroup anchor.

4.4 Handling Member Join/Leave

Under all-or-one scheme, when a member leaves the multicast session or a new member joins the multicast group, we need one packet/link from proxy to the leaves to collect information of current members and one packet/link from leaves to the proxy to recalculate groups. Therefore, handling join/leave amounts to multicasting two extra packets per join/leave. This would not increase much computational overhead for handling dynamic memberships.

5 Simulation and Results

The primary objective of our simulation is to verify that our First-fit and All-or-one schemes can achieve less bandwidth usage than simply multicast or unicast recovery packets. To strengthen the robustness and reliability of our simulation results, we ran our simulation on Internet-like topologies, varying end-to-end loss probabilities, number of end receivers, size of network, etc. As indicated in Section 2, unicast and conventional multicast are widely adopted as local recovery strategies in many well-known reliable multicast schemes [1, 6, 7, 9, 13]. Our simulation results show that first-fit and all-or-one schemes perform significantly better than multicast and unicast recovery. Given the cost of implementation (in terms of router memory), we recommend using all-or-one for recovery by a proxy.

5.1 Simulation Details

5.1.1 Internet-Like Topologies Generated by Brite. As described earlier, our network topologies were generated with the top-down hierarchical approach in Brite [10]. Like Internet it consists of many autonomous systems (AS). The routers involved in inter-AS connections are AS-level nodes, while other routers that are primarily connected by intra-AS connections are called router-level or RT-level nodes. In our network topology configuration, AS-level nodes are randomly placed and connected with *ASWaxman* model [10]. RT-level nodes apply heavy-tailed placement and are connected using *RouterBarabasiAlbert* model [10] that reflects a power-law in the frequency of outdegrees in network topologies, caused by *incremental growth* and *preferential connectivity*, two properties that are observed in real Internet. Other parameters are set by default in Brite.

On each topology, a number of receivers are placed. Each receiver is attached to an unrepeatable RT-level router that is uniformly distributed. One receiver is randomly chosen to be the source. The multicast tree is a spanning subtree rooted at the source and involving all the receivers.

The hop count for each link is one and the loss probability on each link is a random number between 0.8 percent and 6 percent, which leads to the end-to-end loss probability in our network up to 60 percent. According to the statistics of Internet performance in PingER project [11], 95 percent or more hosts in Internet just endure with at most 10 percent or so end-to-end packet loss. Therefore, our simulation network is even more unreliable than the real Internet. The performance metric is the total cost of bandwidth (counted by hops) per packet recovered.

5.1.2 Simulation Plan. We used a discrete event packet level simulator and designed four simulations to test the performance of the recovery schemes under different end-to-end loss probabilities, number of receivers, number of routers and hierarchical structures, respectively (Figures 9-12). For each case, we run it on 100 different multicast sessions and take the average value as the final result. Also, in each multicast session we simulate sending 1000 packets and count the average total cost of bandwidth for each packet recovery as the result for this session.

5.1.3 Simulation Results and Analyses. (1) Recovery cost for different end-to-end loss probabilities (fixed number of receivers): We used the topology where there are 1000 routers (20 AS and 50 RT-level nodes in each AS) and fix the number of end-receivers at 200. We adjust the per-link loss probability to be a uniformly distributed number between 0.8 percent and 6 percent such that the average end-to-end loss probability varies from 5 percent to 60 percent. From Figure 9, we can see that First-fit and All-or-one schemes consistently require less recovery bandwidth than unicast and multicast when the end-to-end loss probability is between 5 percent and 60 percent. This range includes the range of end-to-end loss probabilities in Internet [11]. Therefore First-fit and All-or-one are supposed to perform well in Internet. First-fit scheme performs slightly better than All-or-one while All-or-one has the advantage of requiring no additional router memory as explained earlier. Also note that multicast strategy spends almost the same bandwidth for recovery regardless of the reliability of the network only if the size of the topology and the number of receivers are fixed. While for unicast, the more unreliable the network is, the worse unicast strategy performs.

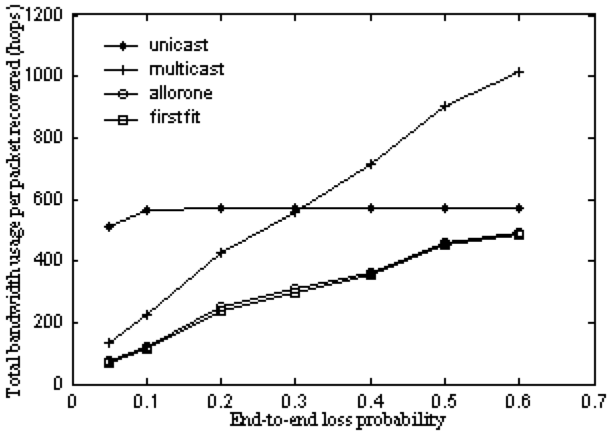


Figure 9: Total bandwidth usage per packet recovered for different end-to-end loss probabilities

(2) Recovery cost for varying number of receivers for fixed end-to-end loss probability: We fix the end-to-end loss probability at 30 percent (by choosing appropriate per link loss probability ranges), the topology size at 1000 (20 AS and 50 RT-level nodes for each AS) and generate 50, 100, 150, 200, 250, 300, 350, and 400 receivers respectively such that the number of receivers varies between 5 percent and 40 percent of number of routers. This simulation is to test the scalability of the recovery schemes. In Figure 10, we can see that the cost of bandwidth for recovery for all four schemes increases with the number of receivers. However, not only in each single case First-fit and All-or-one schemes perform better than multicast and unicast, but the cost for First-fit and All-or-one increases also in a lower degree of order than that of multicast and unicast. In addition, Multicast recovery and unicast recovery has the same scalability and cost when the end-to-end loss probability is 30 percent. The performance of First-fit is marginally better than All-or-one.

(3) Effect of topology size: We fix the end-to-end loss probability at 30 percent and the number of receivers at 100. We run our simulation on different topologies where there are 500, 800, 1000, 1200, 1400, 1600, 1800, and 2000 routers, respectively. The purpose of this simulation is to check the influence of the network size on the performance of the recovery schemes. According to Figure 11, the cost of the four

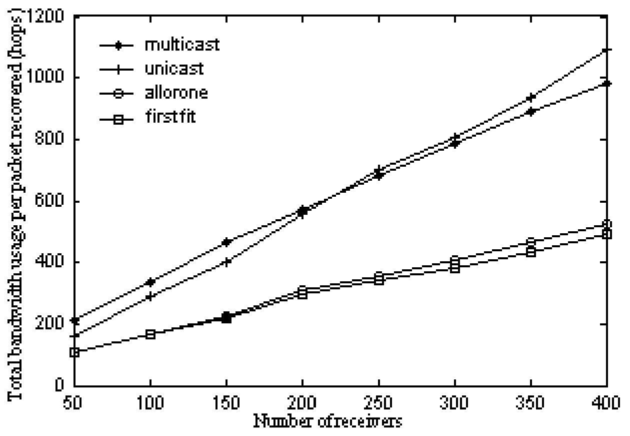


Figure 10: Total bandwidth usage per packet recovered for different number of receivers

schemes increases slightly with the size of the network growing. This means the size of the network has little influence on the cost of recovery in terms of bandwidth usage. Also note that First-fit and All-or-one perform much better than multicast and unicast, and First-fit somewhat outperforms All-or-one.

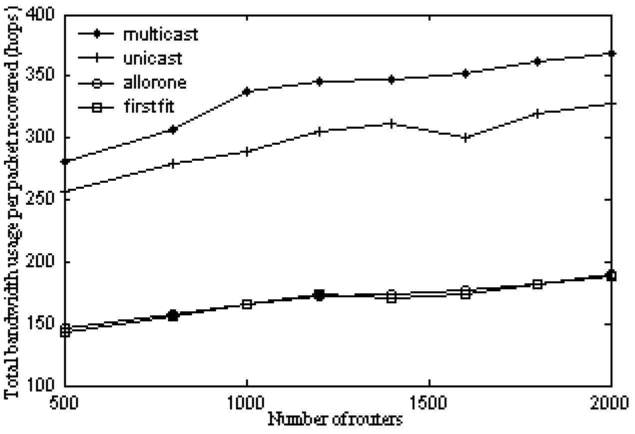


Figure 11: Total bandwidth usage per packet recovered for different number of routers

(4) Effect of hierarchical structures: We fix the end-to-end loss probability at 30 percent, the number of receivers at 200 and topology size at 1000, but change the hierarchical structures by varying number of autonomous systems (AS) from 5, 10, 20 to 25 and keeping 200, 100, 50, 40 routers in each AS, respectively. The purpose of this simulation is to test the influence of different hierarchical structures on performance of the four schemes. The results in Figure 12 show that the hierarchical structure has little impact on the performance of these four schemes and First-fit and All-or-one outperform multicast and unicast by a large degree. First-fit costs slightly less bandwidth for recovery than All-or-one.

5.2 Simulation Conclusions

- For the end-to-end loss probability between 5 percent and 60 percent or larger range, First-fit and All-or-one

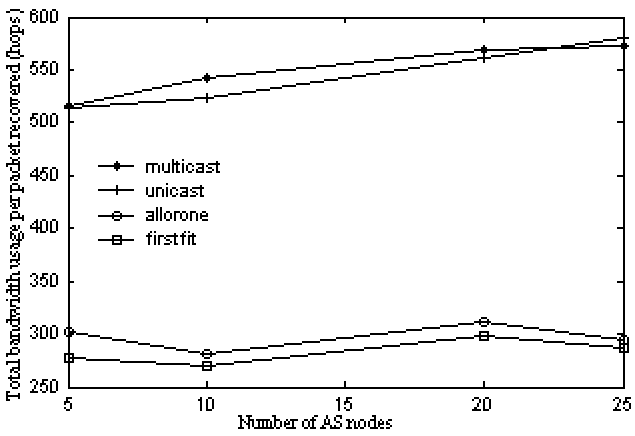


Figure 12: Total bandwidth usage per packet recovered for different number of AS nodes

schemes consistently achieves less bandwidth usage for recovery than multicast strategy and unicast strategy.

- First-fit and All-or-one schemes are more scalable than multicast strategy and unicast strategy if the end-to-end loss probability is a fixed value.
- The topology size and the hierarchical structure of the network have little influence on the performance of the four schemes.
- The performance of First-fit is slightly better than that of All-or-one. However, All-or-one is much easier to implement and costs no additional router memory.

6 Conclusion and Future Work

In this paper, we investigate a subgrouping scheme for reliable multicasting. We prove this subgrouping problem is NP-hard. And according to our simulation analysis, our *First-fit* and *All-or-one* heuristics' algorithms can greatly reduce the bandwidth usage for retransmission. Besides, our *All-or-one* heuristic can minimize the usage of intermediate routers' memory for recovery purpose by using LSRR service provided by IP.

We are currently working on a provably bounded heuristic for computing the optimal subgrouping scheme. In addition, we believe that pseudo-polynomial time algorithm may exist for our subgrouping problem when the height of the tree is bounded.

References

[1] Y. Chawathe, S. McCanne, and E. Brewer, "RMX: Reliable Multicast in Heterogeneous Networks," *INFOCOM '00*, pp. 795-804.

[2] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang, "A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing," *IEEE/ACM Transactions on Networking*, 5(6):784–803, Dec. 1997.

[3] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness,* W. H. Freeman & Company, 01/1979.

[4] S. Guha, A. Markopoulou, and F. Tobagi, "Hierarchical Reliable Multicast: Performance Analysis and Placement of Proxies," *Computer Communications*, 26:2070-2081, Oct. 2003.

[5] S. K. Kasera, G. Hjalmtysson, D. Towsley and J. Kurose, "Scalable Reliable Multicast using Multiple Multicast Channels," *IEEE/ACM Transactions on Networking*, 8(3):294-310, June 2000.

[6] K. Lee, S. Ha, and V. Bharghavan, "IRMA: A Reliable Multicast Architecture for the Internet," IEEE INFOCOM, pp. 1274-1281, 1999.

[7] L. Lehman, S. J. Garland, and D. L. Tennenhouse, "Active Reliable Multicast," INFOCOM '98, pp. 581-589, San Francisco, CA.

[8] B. N. Levine and J. J. Garcia-Luna-Aceves, "Improving Internet Multicast with Routing Labels," IEEE International Conference on Network Protocols (ICNP-97I), pp. 241-250, October 28-31, 1997.

[9] J. C. Lin and S. Paul, "RMTP: A Reliable Multicast Transport Protocol," IEEE INFOCOM '96, pp. 1414-1424, March 1996.

[10] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: An Approach to Universal Topology Generation," Proceedings of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecom-munications System*s*, Cincinnati, OH, August 2001.

[11] PingER Project, Stanford University, http://www-iepm.slac.stanford.edu/pinger/tools/table.html.

[12] RFC 791, Internet Draft, http://www.zvon.org/tmRFC/RFC791/Output/index.html.

[13] I. Rhee, N. Balaguru, and G. N. Rouskas, "MTCP: Scalable TCP-like Congestion Control for Reliable Multicast," IEEE INFOCOM, pp. 1265-1273, 1999.

[14] L. Rizzo and L. Vicisano, "A Reliable Multicast Data Distri-bution Protocol Based on Software FEC Techniques," Proc. HPCS'97, Chalkidiki, Greece, June 23-25, 1997.

[15] D. Rubenstein, J. Kurose, and D. Towsley, "Real-time Reliable Multicast using Proactive forward Error Correction," Proceedings of 8th International Workshop NOSSDAV, 1998.

[16] Tony Speakerman, Dino Farinacci, Steven Lin, and Alex Tweedly, *PGM Reliable Transport Protocol Specif-ication*, Internet Draft, http://www.networksorcery.com/enp/rfc/rfc3208.txt.

[17] K. Sripanidkulchai, A. Myers, and H. Zhang, "A Third-party Value-added Network Service Approach to Reliable Multicast," SIGMETRICS '99*,* pp. 166-177.

[18] D. Towsley, J. Kurose, and S. Pingali, "A Comparison of Sender-initiated and Receiver-Initiated Reliable Multi-cast Protocols," *IEEE Journal on Selected Areas in Communications*, 15(3):398-406, 1997.

[19] T. Wolf and J. Turner, "Design Issues for High-Performance Active Routers," *IEEE Journal on Selected Areas in Communications*, 19:404-409, March 2001.

[20] D. Zhang, S. Ray, R. Kannan and S. S. Iyengar, "A

Recovery Algorithm for Reliable Multicasting in Reliable Networks," ICPP'03, the 32nd International Conference on Parallel Processing, Taiwan, pp. 493-500, Oct. 6-9, 2003.

Appendix: Proof Sketches (Lemma 2, Lemma 3, Theorem 1, and Theorem 2)

Proof of Lemma 2. According to Lemma 1, the factor needed to be minimized for an optimal subgrouping scheme is

$$\sum_{i=1}^{k} {e_i}^2 = \sum_{i=1}^{k}\left(a + \sum_{b_j \in G_i} b_j\right)^2 = ka^2 + 2a\sum_{j=1}^{n} b_j + \sum_{i=1}^{k}\left(\sum_{b_j \in G_i} b_j\right)^2$$

for various subgrouping schemes, we know $2a\sum_{j=1}^{n} b_j$ will be the same, thus the cost of recovery we need to consider here is actually just $C_r = ka^2 + \sum_{i=1}^{k}\left(\sum_{b_j \in G_i} b_j\right)^2$ where $1 \le k \le n$. Without losing generality, assume $C_a = ka^2$ and $C_b = \sum_{i=1}^{k}\left(\sum_{b_j \in G_i} b_j\right)^2$, then $C_r = C_a + C_b$. Notice that even for the same value of *k*, different subgroup partition leads to different cost. Thus for a given b_1, b_2, ..., b_n and *a*, suppose we find $k_{opt} = k_1$ for a minimal C_r for tree T_1.

This means that for any $k_0 > k_1$, for any subgroup partition $G_{k_0}^i$ (where $1 \le i \le C_n^{k_0}$), we have $C_a^{k_0} > C_a^{k_1}$ and $C_b^{k_0} < C_b^{k_1}$. Because $C_r^{k_0} > C_r^{k_1}$, we get $C_a^{k_0} - C_a^{k_1} > C_b^{k_1} - C_b^{k_0}$. If *a* is increased by some amount ε (tree changes from T_1 to T_2), assume for tree T_2, $k_{opt} = k_2$ and $k_2 > k_1$, for any subgroup partition $G_{k_2}^i$ (where $1 \le i \le C_n^{k_2}$), we have $C_{a+\varepsilon}^{k_2} - C_{a+\varepsilon}^{k_2} > C_a^{k_2} - C_a^{k_1} > C_b^{k_1} - C_b^{k_2}$. Therefore $C_r^{k_2} > C_r^{k_1}$, hence $k_1 \ge k_2$.

Proof of Lemma 3. If $a = 0$, then $C_r = \sum_{i=1}^{k}\left(\sum_{b_j \in G_i} b_j\right)^2$. Obviously the smallest value for C_r is to separate $M_1, M_2, \cdots, M_n$ into *n* disjoint subgroups, that is, $G_i = \{M_i\}$ for $1 \le i \le n$. Therefore, $k_{opt} = n$. If $a_1 = \sum_{i=1}^{n} b_i$, then $C_r = ka^2 + \sum_{i=1}^{k}\left(\sum_{b_j \in G_i} b_j\right)^2 = k\left(\sum_{i=1}^{n} b_i\right)^2 + \sum_{i=1}^{k}\left(\sum_{b_j \in G_i} b_j\right)^2$. When $k = 1$, $C_r = 2\left(\sum_{i=1}^{n} b_i\right)^2$. And for any $k > 1$, we have $C_r = k\left(\sum_{i=1}^{n} b_i\right)^2 + \sum_{i=1}^{k}\left(\sum_{b_j \in G_i} b_j\right)^2 > 2\left(\sum_{i=1}^{n} b_i\right)^2$. Therefore, $k_{opt} = 1$.

So if the value of *a* starts at 0 and is increased to $\sum_{i=1}^{n} b_i$, the number of subgroups for optimal subgrouping scheme will be decreased from *n* to 1 monotonously. □

Proof of Theorem 1. We prove that the optimal subgrouping problem is NP-hard by showing that $MSS \le_p OSG$. The *MSS* problem can be described as following according to [3], given finite set *A*, a size $s(a) \in Z^+$ for each $a \in A$, positive integer $K \le |A|$, then whether *A* can be partitioned into *K* disjoint sets A_1, ..., A_K such that $\sum_{i=1}^{K}\left(\sum_{a \in A_i} s(a)\right)^2$ is minimized is NP-hard in the strong sense. The reduction algorithm takes as input an instance of *MSS* problem. Based on Lemma 1 and 2, we can construct many instances for *OSG* problem according to the following steps.

Let $G = A$, $n = |A|$, i.e., $b_i = size(a)$ for each $a \in A$ and $1 \le i \le n$, then we construct

Instance 1: For $K = n$, let $a = 0$, then $k_{opt} = K = n$.

Instance 2: For $K = 1$, let $a = \sum_{i=1}^{n} b_i$, then $k_{opt} = K = 1$.

Instance 3: For $1 < K < n$, we can find the value of *a* such that $k_{opt} = K$ using the following binary search algorithm,

Pseudo-code for finding the appropriate value of *a*:

```
1. $a_1 = 0$; $a_2 = \sum_{i=1}^{n} b_i$; $a = \frac{a_1 + a_2}{2}$
2. Calculate $k_{opt}$ according to the value of $a$
3. Repeat If $k_{opt} < K$ Then $a_2 = a$; $a = \frac{a_1 + a_2}{2}$
4.     Else If $k_{opt} = K$ Then break
5.         Else $a_1 = a$; $a = \frac{a_1 + a_2}{2}$
6.     EndIf
7.   EndIf
8.   Calculate $k_{opt}$ according to the value of $a$
9. Until $a_1 = a_2$
```

Note that here we assume we can find an optimal subgrouping scheme to minimize $ka^2 + \sum_{i=1}^{k}\left(\sum_{b_j \in G_i} b_j\right)^2$ within polynomial time and from above we have proven we can find the value of a matching $k_{opt} = K$ in $O(\lg n)$ time (for this binary search algorithm), hence we can find a partition scheme to minimize $\sum_{i=1}^{k_{opt}}\left(\sum_{b_j \in G_i} b_j\right)^2$ (that is, $\sum_{i=1}^{K}\left(\sum_{a \in A_i} s(a)\right)^2$) within polynomial time. This is contrary to the theorem in [3]. Thus, our assumption is incorrect. We cannot find an optimal subgrouping scheme to minimize $ka^2 + \sum_{i=1}^{k}\left(\sum_{b_j \in G_i} b_j\right)^2$ within polynomial time. Hence we have proven that $OSG \in NP-hard$. □

Proof of Theorem 2. Consider Figure 4 and the following Figure 13. We have already proven that to find an optimal subgroup partition is NP-hard. Now we multiply $a, b_1, b_2, \cdots, b_n$ (without losing generality, suppose n is even) by a large constant number r, construct a tree T' by replacing the node R in T with a binary tree on which the cost of each link is one unit. If we investigate this binary tree, we can see that there are totally $2(n-1)$ links in it, where n is the number of leaf nodes. Therefore, the cost of recovery C_r for tree T' satisfies the following condition,

$$C_r < \sum_{i=1}^{k}\left[r\left(a + \sum_{b_j \in G_i} b_j\right) + 2(n-1)\right]^2 \qquad (3)$$

In (3), if we let $r = n^c$, where $c \in Z^+$, and then if we intend to ignore the value introduced by $2(n-1)$, we have to make $r^2 >> rn$ by expanding the above formula, where $>>$ means far more greater than. And $r^2 >> rn \Rightarrow n^{2c} >> n^{c+1} \Rightarrow 2c >> c+1 \Rightarrow c >> 1$. Therefore, we can just let c be a number that is far more greater than 1, then the extra value introduced by $2(n-1)$ in the above formula can be ignored. And it is obvious that we can find an optimal subgrouping scheme for T that minimizes $\sum_{i=1}^{k}\left(a + \sum_{b_j \in G_i} b_j\right)^2$ if and only if we can find an optimal subgrouping scheme for T' that minimizes $\sum_{i=1}^{k}\left[r\left(a + \sum_{b_j \in G_i} b_j\right)\right]^2$, the former problem has been proven to be one NP-hard problem. Therefore, the OSG problem for a degree bounded multicast tree is also NP-hard.

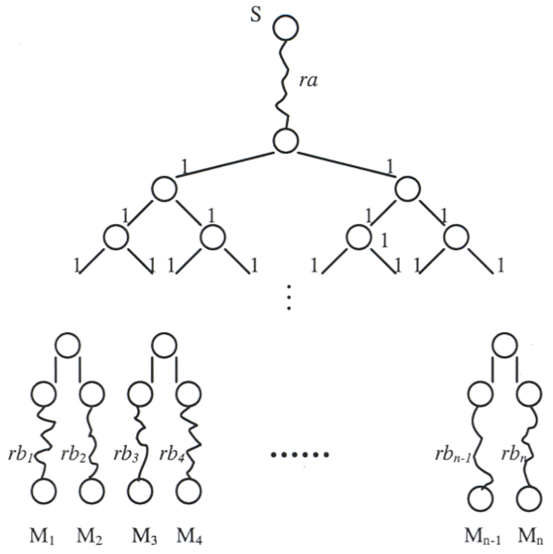
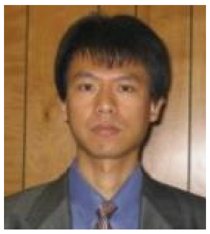


Figure 13: A degree bounded multicast tree T' constructed from T



Danyang Zhang received his Ph.D. degree in Computer Science from the University of Alabama. He is currently a Visiting Assistant Professor in the Department of Computer and Mathematical Sciences in New Mexico Highlands University. He has published several journal papers and a number of conference papers. His research interests include multicasting, network security (cryptography), sensor networks, distributed computing, mobile ad hoc network, graph theory.

Sibabrata Ray received his Ph.D. degree in Computer Science from the University of Nebraska. He is currently an Assistant Professor in the Department of Computer Science at the University of Alabama. He has numerous publications in prestigious conferences and journals. His research interests include Parallel and Distributed Systems, Networking, Sensor Networks, Network Security, Wireless and Ad-Hoc Networking, Distributed Algorithms.

Rajgopal Kannan received his Ph.D. degree in Computer Science from the University of Denver. He is currently an Assistant Professor in the Department of Computer Science in Louisiana State University. He has numerous publications in prestigious conferences and journals. His research interests include Wireless Sensor Networks, Network Security, Wireless and Ad-Hoc Networking, Multistage Interconnection Networks, ATM and Broadband ISDN, Game Theory for Network Control.

S. Sitharama Iyengar is a Fellow of the Association of Computing Machinery (ACM), Fellow of the American Association of Advancement of Science (AAAS), Fellow of the Institute of Electrical and Electronics Engineering (IEEE), and has served on numerous panels for the US-National Science Foundation, National Research Council (Reviewed Proposals), and the Defense Advanced Research Projects Agency, Member of the European Academy of Sciences, etc. He has also published over 300 publications in refereed journals and conference proceedings. Also he has authored/coauthored/edited 13 books, which are used, throughout the nation and the world. He is the Roy Paul Daniels Professor of Computer Science and Chairman in the Department of Computer Science at Louisiana State University.