Lateral Error Recovery for Media Streaming in Application-Level Multicast

W.-P. Ken Yiu, *Student Member, IEEE*, K.-F. Simon Wong, S.-H. Gary Chan, *Senior Member, IEEE*, Wan-Ching Wong, Qian Zhang, *Senior Member, IEEE*, Wen-Wu Zhu, *Senior Member, IEEE*, and Ya-Qin Zhang, *Fellow, IEEE*

***Abstract*—We consider media streaming using application-level multicast (ALM) where packet loss has to be recovered via retransmission in a timely manner. Since packets may be lost due to congestion, node failures, and join and leave dynamics, traditional "vertical" recovery approach where upstream nodes retransmit the lost packets is no longer effective. We therefore propose lateral error recovery (LER). In LER, hosts are divided into a number of planes, each of which forms an independent ALM tree. Since error correlation across planes is low, a node effectively recovers its error by "laterally" requesting retransmission from nearby nodes in other planes.**

We present analysis on the complexity and recovery delay on LER. Using Internet-like topologies, we show via simulations that LER is an effective error recovery mechanism. It achieves low overhead in terms of delivery delay (i.e., relative delay penalty) and physical link stress. As compared with traditional recovery schemes, LER attains much lower residual loss rate (i.e., loss rate after retransmission) under a certain deadline constraint. The performance can be substantially improved in the presence of some reliable proxies.

***Index Terms*—Application-level multicast (ALM), error recovery, retransmission deadline, streaming applications.**

I. Introduction

IP MULTICAST-CAPABLE routers are still not commonly deployed to achieve global multicast at the network layer. Furthermore, IP multicast is generally based on UDP, which leads to flow control and reliable difficulties. To overcome these limitations in IP-multicast, application-level multicast (ALM) has been proposed for point-to-multipoint streaming applications such as video conferencing, movie streaming, etc [1]–[3]. In ALM, multicast functionalities are shifted from the network layer to end-hosts, whereby hosts forward messages from one to another via piece-wise unicasts. Due to the flexibility and freedom in tree configuration at the application layer, ALM is a much more practical approach than protocol design at the network layer for real-time applications [4]–[7]. It does not require any support of multicast-capable infrastructure and flow-control and error recovery can be more readily introduced due to the piece-wise end-to-end connections.

Traditionally, most ALM research has been focusing on the *connectivity* among end-hosts by addressing how messages are routed from one point (the server or origin) to all the other group members. In order to successfully deploy streaming service with ALM, we need to further address its quality-of-service (QoS) issues [8]. In this work, we consider error recovery issue in ALM networks. For streaming service, there is usually a certain playback (or recovery) deadline within which lost packets have to be recovered. The goal is to achieve low residual loss rate (i.e., the loss rate after error recovery within the deadline) without compromising ALM tree performance (in terms of link stress and relative delay penalty).

Given that TCP is not so applicable for streaming applications (where timely recovery rather than 100% reliability is more important), we focus on using UDP in this paper. In UDP, packet loss has to be recovered by the applications. Besides congestion in underlay links, loss may arise from nodal join and leave activities or failures, which lead to transient tree reconfiguration.[1] Since loss may persist for some time, a good recovery mechanism is essential.

Traditional ALM protocols tend to cluster or chain nearby hosts together to form a tree for data delivery. While this reduces ALM delay and stress, it is not effective for error recovery. Whenever there is an upstream error, all downstream hosts are affected. Since hosts close together are clustered, their losses are correlated, leading to low retransmission efficiency. The problem becomes even more serious when the number of hosts increases, because loss rate increases with the depth of the tree.

A natural way to deal with packet loss is to request the upstream hosts for retransmission, the so-called "vertical" recovery. However such mechanism suffers from the following problems.

- High error correlation: As mentioned above, the errors of all downstream nodes are correlated upon an upstream error. Therefore, the parent of a failed node is also likely in error. Given a node does not know where the error occurs, the retransmission request may have to be forwarded upstream multiple times before the packet is retransmitted. This incurs long recovery delay.
- Implosion problem: Vertical recovery may lead to implosion if retransmission requests from downstream nodes

Manuscript received January 24, 2005; revised October 18, 2005. This work was supported in part by the Areas of Excellence (AoE) Scheme on Information Technology funded by the University Grant Council in Hong Kong (AoE/E-01/99), and by the Research Grant Council in Hong Kong (HKUST6199/02E & HKUST6156/03E). The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Benjamin W. Wah.

W.-P. K. Yiu, K.-F. S. Wong, S.-H. G. Chan, W.-C. Wong, and Q. Zhang are with the Department of Computer Science, the Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong (e-mail: kenyiu@cs.ust.hk; cssm@cs.ust.hk; gchan@cs.ust.hk; wwilliam@cs.ust.hk; qianzh@cs.ust.hk).

W.-W. Zhu is with the Intel Communication Technology China Lab, Beijing 100080, China (e-mail: wenwu.zhu@intel.com).

Y.-Q. Zhang is with Microsoft Research Asia, Beijing 100080, China.

Digital Object Identifier 10.1109/TMM.2005.864268

[1]We use the terms "host" and "node" interchangeably in this paper.

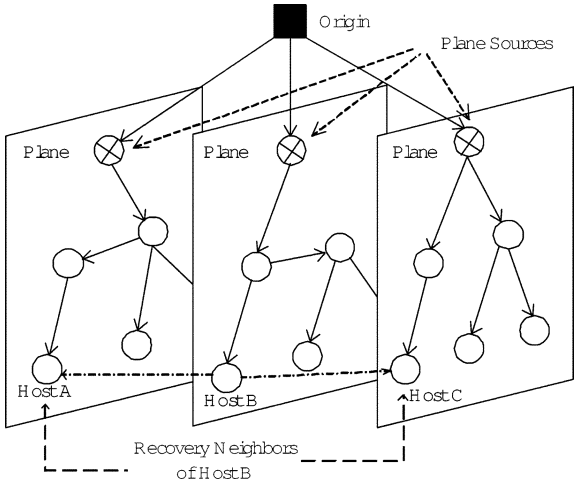


Fig. 1. Lateral error recovery using tree planes. Hosts A and C are the recovery neighbors of B.

are not aggregated (which is often the case for simplicity). Therefore, the parent where the error originates may be overwhelmed by requests.

- Outage due to network dynamics: Due to the failure of a node/link or group dynamics such as join/leave activities, tree reconfiguration would lead to temporary outage for all downstream nodes from their upstream. In this case, many nodes do not even know their parents and hence vertical recovery is simply not possible.

To address these weaknesses, we propose the use of *lateral error recovery* (LER). Hosts are first divided into ω different planes, each of which independently forms an ALM tree. Due to the random nature of dividing hosts into planes, hosts close together likely belong to different planes. Since data are delivered along different trees independently, the error correlation across different planes is greatly reduced. This greatly increases the efficiency of recovering error across planes.

We show the basic idea of LER in Fig. 1, where hosts are distributed into multiple planes with the source indicated as the "origin." The origin first sends data to each of the plane sources, where ALM trees are formed independently among all the hosts in the planes. Consider an arbitrary host B. It first identifies N hosts in each of the other planes as its recovery neighbors ($N = 1$ in the example). Whenever an error occurs, host B performs "lateral" retransmission with its recovery neighbors according to some precomputed retransmission sequence. One may picture host B as temporarily attached to its recovery neighbors upon an error. A strength of this system is that recovery no longer depends on upstream nodes, but adjacent "lateral nodes." This greatly alleviates error correlation and the implosion problem.

We address the following two major issues in LER.

- Identification of plane sources

  The nodes close to the origin are chosen as the plane sources. This reduces end-to-end delay and physical link stress. To achieve this, we first obtain host coordinates by means of a positioning technique such as global network positioning (GNP) or Vivaldi [9], [10]. Based on the coordinates, the closest nodes to the origin can be found with Voronoi diagram. We discuss how to construct the Voronoi diagram by a centralized and a distributed algorithms.
- Identification of one's recovery neighbors and, upon an error, the sequence with which the neighbors are requested for retransmission.

  A good choice of recovery neighbors leads to low recovery delay. We show how to identify the set of close recovery neighbors from the constructed Voronoi diagram and how to sequence them for retransmission requests.

We also propose a simpler LER which does not require network coordinates to identify plane sources and recovery neighbors. Since recovery neighbors are identified from a set of randomly chosen nodes, the scheme is called random neighbor selection (RNS). The recovery mechanism of RNS is the same as the original one based on network coordinates, i.e., retransmissions are conducted across planes. The scheme is simple to implement, but trades off some recovery efficiency.

We compare LER with two vertical recovery schemes, namely, *Source Recovery* (recovery from the origin) and *Parent Recovery* (recovery from one's parent). We have also simulated a recently proposed recovery scheme, *Probabilistic Resilient Multicast* (PRM) [11]. By using Internet-like topologies, we show that LER achieves low overhead in terms of streaming delay (i.e., relative delay penalty) and physical link stress. As compared with vertical recovery schemes and PRM, LER achieves substantially lower residual loss rate.

We briefly discuss previous work here. Many ALM protocols have been proposed in literature, such as ALMI, CoopNet, NICE, YOID, Narada, DT, etc. [12]–[24]. All these schemes do not consider the error recovery issue and are based on a single tree. Our approach differs by considering error recovery issues by using multiple plane-trees. Our work is complementary to this body of work in the sense that in each plane, we may construct the delivery tree by any of the schemes.

An error recovery scheme named PRM has been recently proposed [11]. It adds some extra paths or "links" into the ALM tree. Packets are replicated along these links with some probability. Though this approach reduces the error rate, it introduces redundant packets and hence high bandwidth overhead. Our LER scheme is shown to offer a lower error rate and bandwidth overhead. As different from [25], we propose in this paper a simpler scheme (RNS) which does not depend on network coordinates and present a simpler alternative to identify plane sources and recovery neighbors. We also consider how reliable proxies can be integrated with LER to provide better recovery capability, and how to recover packet loss in the presence of membership changes. Note that forward error correction (FEC) can be integrated with LER to further reduce the error rate (for FEC as used in streaming applications, readers may refer to [26]–[32]). We may first perform FEC to repair the lost packets. After such repair, LER can then be applied.

There has been much work on providing reliable services over IP multicast networks. Our work differs from them in addressing streaming service with a certain recovery deadline. Furthermore, much of the previous work focuses on the design of scalable feedback mechanisms for the source-implosion problem [33]–[37]. In ALM, recovery from the source is costly

in terms of delay and bandwidth. Because some ALM nodes may buffer the lost packets, retransmission requests do not need to go all the way to the source. This fundamentally changes how recovery mechanism should be designed. Recovering loss from local groups of users in IP multicast has also been discussed in [38], [39]. In these schemes, group members in proximity with each other collaborate to recover each other's errors. These groups are usually fixed or determined administratively without much flexibility in self-configuration. In ALM, the particular nodes to recover losses can be *adaptively chosen* to improve the recovery probability. We show how this is done in this paper. Using proxies to recover loss in IP multicast has been investigated in [40]–[43]. It generally involves intelligent placement of proxies either in the IP multicast tree or as a separate tree so that recovery can be done by traversing upstream of the proxy tree. LER does not require the proxies to be preconfigured in any special manner as above, and hence is more flexible and self-adaptive.

The remainder of this paper is organized as follows. The operation and the complexity analysis of lateral error recovery is presented in Section II. We address the use of proxies and join/leave activities in Section III. Illustrative simulation results are shown in Section IV, followed by the conclusion in Section V. We also present an analysis on recovery delay of LER in Appendix.

II. Lateral Error Recovery

In this section, we first give an overview of LER operation in Section II-A. Then, we discuss in detail each step in the operation, namely, selection of plane sources (Section II-B) and selection of recovery neighbors and retransmission schedule (Section II-C). We also describe a distributed version of LER in Section II-D. Lastly, we present the computational complexity of LER in Section II-E.

A. LER Overview

In this section, we present the overview of LER. There are two variants of LER: one based on network coordinates, and another based on random neighbor selection (RNS). For concreteness in our exposition and simulation, we will use GNP to identify nodal coordinates, though other technique such as Vivaldi can also be used [9], [10].

In LER, there are ω planes, where $\omega \geq 1$ (the system degenerates to the traditional single-tree approach when $\omega = 1$). A joining host is randomly assigned to any one of the ω planes. Due to this random assignment, hosts close together are likely to be distributed into different planes. This simple mechanism also roughly balances the number of nodes in each plane. (Note that LER operation does not depend on the exact balanced distribution of nodes among the planes.) By adjusting the parameter ω, we achieve different levels of tradeoff between network performance (in terms of delay and link stress) and recovery efficiency.

Once nodes are assigned to planes, the plane sources and the recovery neighbors for each node are identified. The issues are to identify the nodes that are close to the origin as plane sources and that are close to a node/host as its recovery neighbors. Given that it is costly to ping every pair of nodes to find their mutual distances, each host first estimates its network coordinates and identify the closest nodes by Voronoi diagrams. We denote this approach as LER(GNP). (We discuss another version based on random neighbor selection denoted as "LER(RNS)" in Section II-D.)

An ALM delivery tree is constructed in each plane using any traditional ALM tree/mesh building scheme (e.g., ALMI, CoopNet, Narada, DT, etc). In our study, we have arbitrarily chosen an existing distributed protocol called Delaunay Triangulation (DT) to build the tree [17].

The origin marks each packet in the media stream with increasing sequence number. Transient packet loss and node failures can be detected by gaps or timeout in the sequence number. LER may be integrated with other more sophisticated mechanism to discover errors [40]. Upon detecting an error, the host performs lateral error recovery with its recovery neighbors. In this paper, we are interested in such recovery time *after* an error is detected. The steps of LER can be summarized as follows.

1) Assignment of hosts into planes.
2) Estimation of network coordinates for each host using any coordinate location technique.
3) Construction of the Voronoi diagram for each plane. (Steps 2 and 3 are not necessary for RNS.)
4) Plane source selection for each plane.
5) ALM tree construction for each plane.
6) Identification of recovery neighbors and the sequence of retransmission request.
7) Data delivery.
8) Upon detecting an error, a node sequentially requests retransmission from its recovery neighbors according to the precomputed schedule as given in Step 6.

For LER(GNP), we discuss Steps 3 and 4 in Section II-B and Step 6 in Section II-C. We present LER(RNS) in Section II-D.

B. Selection of Plane Sources

Network coordinates can be used to efficiently estimate the distance between two hosts. LER(GNP) makes use of network coordinates to construct the Voronoi diagram, which is then used to find the closest neighbors.

With the network coordinates of all the hosts, LER constructs for each plane a Voronoi diagram, using which LER finds the closest node in each plane to the origin. The Voronoi diagram can be constructed in a centralized or distributed manner, discussed below and illustrated in Fig. 2.

- Centralized approach

  Each node sends its network coordinates to a specific central server. The server then constructs the Voronoi diagram for each plane (excluding the origin) by, for example, Fortune's algorithm [47]. The algorithm sectors the nodes into different regions. The server then informs the origin the closest neighbor of each plane as obtained from the sector with the origin in the Voronoi diagram. We show an example of Voronoi diagram in Fig. 2(a). The square indicates the origin and the circles indicate the coordinates of the nodes in plane 1. From the Voronoi diagram, since node k shares the same sector with the origin, it is the closest to the origin.

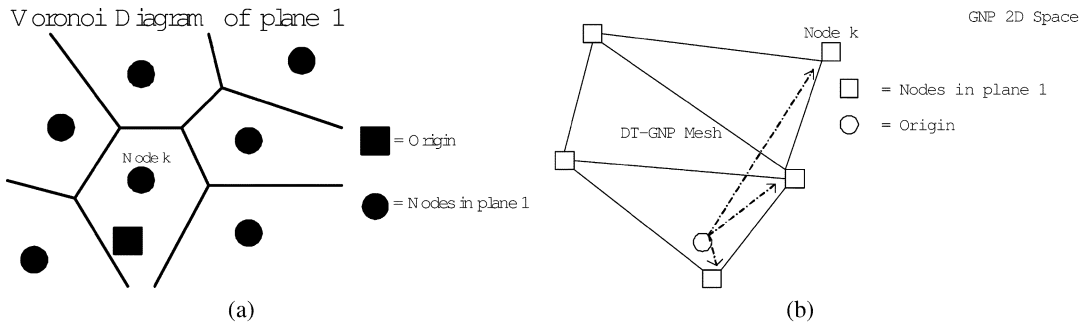


Fig. 2. Finding the closest neighbor in LER(GNP). (a) With a Voronoi Diagram (centralized approach). (b) By successive probing (distributed approach).

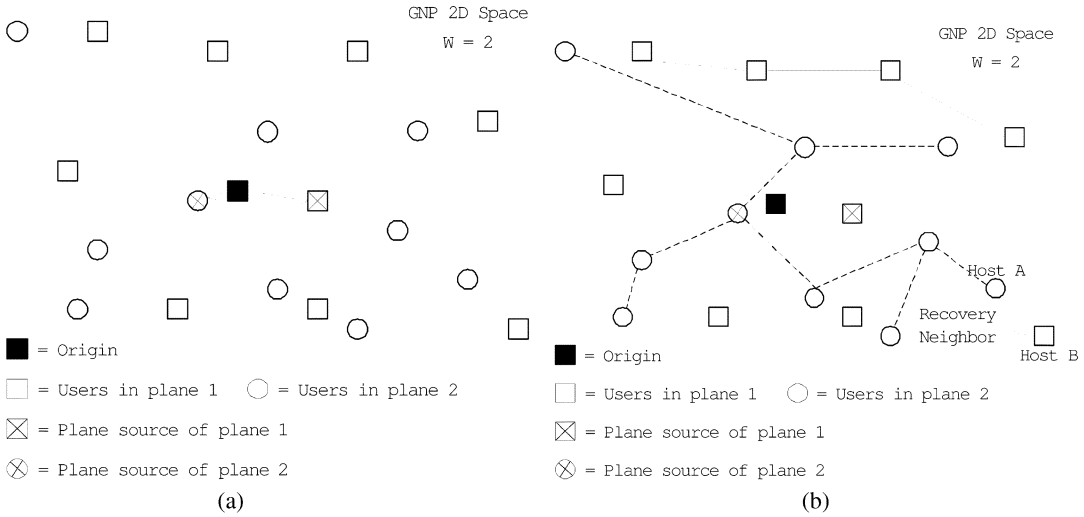


Fig. 3. Tree formation of LER with $\omega = 2$. (a) Finding the closest nodes as the plane sources. (b) Resultant delivery trees.

- Distributed approach

  We construct Delaunay Triangulation (DT) meshes among the hosts by the incremental construction algorithm as discussed in [17]. DT has a property that closer nodes (in the coordinate space) are connected by edges. Note that the DT mesh constructed has already embedded the Voronoi diagram and hence the closest node can be identified easily by successive probing as follows. The origin probes the node one by one. In each step, a closer node is found until the probing terminates at the closest node.

  We illustrate the idea in Fig. 2(b), where the origin (indicated by the circle) is to identify the closest node in a plane, say plane 1 (indicated by squares). The nodes are placed according to their coordinates. The origin first finds an arbitrary node, say node k, in plane 1. Node k checks its connected nodes in the DT mesh and replies to the origin with another node closer to the origin. By repeating this process, the origin eventually finds the closest node in the plane.

Fig. 3 shows an example of LER tree formation with $\omega = 2$ in 2-D coordinate space. Users are divided into two planes indicated by squares and circles [Fig. 3(a)]. The origin finds the closest node in each plane to be the plane source, indicated in the figure by crosses. After identifying the plane sources, ALM trees are built independently to span all the nodes in the plane [Fig. 3(b)].

C. Selection of Recovery Neighbors and Retransmission Schedule

In this section, we discuss how a node finds its recovery neighbors and computes its retransmission schedule (i.e., the node sequence for retransmission requests).

In N-LER, a node finds N nodes in each plane close to itself as recovery neighbors for lateral recovery. For $N = 1$, the neighbors can be found from the constructed Voronoi diagrams for each plane or in a distributed manner as discussed in Fig. 2 in the previous section. For example, in Fig. 3(b), hosts A and B are recovery neighbors of each other, since they belong to different planes and are close to each other. For $N \geq 2$, we may find the set of neighbors by DT. Suppose a node finds its closest neighbor, say node j, using DT. The other recovery neighbors can be the connected hosts of node j in the mesh by comparing their distances with node i.

In addition to the origin, a node puts its $(\omega - 1)N$ recovery neighbors into a list called "Potential Recovery-Neighbors List" (PRL). (Therefore, including the origin, the number of nodes in the PRL is at most $N(\omega - 1) + 1$.) The node then prioritizes the list according to their turnaround time for retransmission purpose. This is done as follows.

Let t_i and t_j be the total delay from the origin to node i and node j, respectively, where $j \in \mathrm{PRL}$ of i. Let d_{ij} and d_{ji} be the one-way delay from i to j and j to i, respectively. We illustrate the definitions in Fig. 4, which shows the time diagram upon

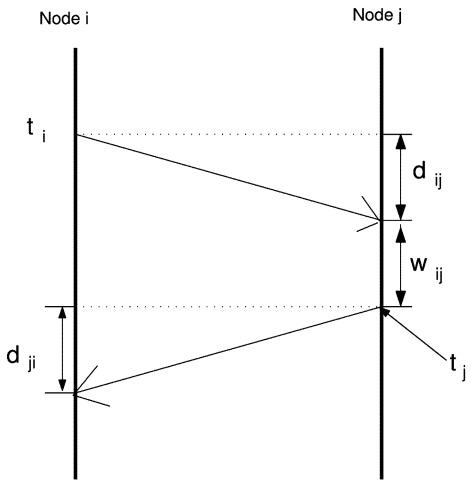


Fig. 4. Time diagram for recovery neighbor selection for the case $t_j > t_i + d_{ij}$.

an error detected at time t_i in node i, given that the packet is transmitted from the origin at time 0. Node i requests a retransmission from node j. Clearly, the retransmission request arrives at node j at time $t_i + d_{ij}$. Let w_{ij} be the waiting time for the requested packet at node j from the arrival of the retransmission request of node i to the arrival of the data packet. (The figure shows the case when $w_{ij} > 0$.) It is not difficult to see that

$$w_{ij} = \max(0, t_j - t_i - d_{ij}). \tag{1}$$

The minimum turnaround time, Γ_{ij}, defined as the delay from the time when node i requests a retransmission from node j until the time the packet arrives at node i, is hence given by

$$\Gamma_{ij} = d_{ij} + w_{ij} + d_{ji}. \tag{2}$$

We see that it is better to request a node with low Γ_{ij} as it leads to faster recovery delay.

We may obtain the time of arrival (i.e., t_i and t_j) from the origin to a node as follows. The origin distributes control messages to each plane source, which in turn multicasts it to its plane nodes using ALM. There is a hop-count field in the control message. Every time a node receives a control message, before forwarding downstream, it increments the hop counter by 1 and appends its IP address and its delay from the origin along the tree. From the control message, a node hence knows all its upstream nodes along the branch from itself to the origin and its corresponding delay from the origin.

In media steaming, there is usually a certain recovery deadline δ (in seconds) after detecting an error. If a lost packet cannot be recovered within δ, it is as good as lost. Clearly, in order to meet the recovery deadline, node i needs to keep only those nodes, say node j, in the PRL which satisfy (see Fig. 4):

$$t_j + d_{ji} \le t_i + \delta \tag{3}$$

i.e., the packet arrival time at node j and its transmission to node i (LHS) has to meet the playback deadline of node i (RHS). Furthermore, we also need

$$d_{ij} + d_{ji} \le \delta, \tag{4}$$

i.e., the round-trip time has to be less than playback deadline.

Failure to meet (3) and (4) means that it is not possible for the requested node to retransmit the packet in time. After eliminating all nodes violating such conditions from PRL, node i then orders the nodes according to their Γ_{ij} and sequences its retransmission requests according to this order. Note that the identification and ordering of recovery neighbors are performed before data delivery or during tree reconfiguration. Therefore, they do not need to be run continuously. The overhead is hence relatively low.

D. Random Neighbor Selection (RNS)

In random neighbor selection (RNS), nodal coordinates are not required. It is a simpler way to obtain the plane sources and recovery neighbors as compared to LER(GNP).

As in LER(GNP), each node in LER(RNS) finds N recovery neighbors in each plane as follows. Each node is first randomly given M nodes in each of the other planes, where $M \ge N$. The node then pings the M nodes and selects N nodes with the lowest turnaround time as its recovery neighbors. For centralized ALM algorithm (e.g., ALMI and CoopNet), the M random nodes can be easily obtained from the central server. For distributed ALM algorithm (e.g., Narada, DT), a node may perform a random walk on a plane tree to find the random nodes.

Clearly, RNS is a suboptimal solution as compared with the GNP scheme. Indeed, as shown in Fig. 14 later, it has a higher error rate as compared with LER(GNP). However, it can be easily implemented in reality.

E. Computational Complexity

The nodal computational complexity refers to the computing steps each node executes in the network. We divide our discussion into two categories—the centralized LER and distributed LER—as follows.

- Centralized LER

  In centralized LER, the major complexity comes from the central server which keeps the network locations of nodes and constructs a Voronoi diagram for each plane by Fortune's sweep line algorithm [47].

  Let n be the number of nodes in the network. Since the number of nodes in each plane are roughly equal to n/ω, the running time complexity for each plane using the Fortune algorithm is $O((n/\omega)\log(n/\omega))$. Therefore, with ω planes, the total complexity at the server is

  $$O\left(\omega \times \frac{n}{\omega}\log\frac{n}{\omega}\right) = O\left(n\log\frac{n}{\omega}\right).$$

  The closest node to the origin in each plane can be identified during the construction of each diagram (by simple bookkeeping operation in $O(1)$ time). Therefore, this identification step does not increase the complexity of the process.
- Distributed LER

  In distributed LER, a Delaunay triangulation mesh is formed among the nodes. The distributed protocol to form such DT mesh overlay was discussed in [17]. According to the results in [17], the protocol only consumes $O(1)$ bandwidth and computation at each node.

We next discuss both the time and space complexity for recovery retransmission. Note that a node has a total of $O((\omega - 1)N) = O(\omega N)$ recovery neighbors, and hence sorting them takes $O(\omega N \log(\omega N))$ time. A node needs to request for retransmission of its lost packet from its recovery neighbors one by one. This can be done in $O(1)$ time using the sorted sequence. In LER, all a node needs to keep is only a PRL of $(\omega - 1)N$ recovery nodes, therefore the state maintenance cost at each node is $O(\omega N)$.

III. Proxies and Join/Leave Issues

In this section, we discuss other issues of LER. We present how to use reliable proxies to improve the loss recovery of the system (Section III-A) and how LER recovers packet loss arisen from dynamic membership changes (Section III-B).

A. Use of Proxies

In order to further improve the error recovery capability, we may put some permanent (or long-lived) proxies in the network. These proxies are unlikely to fail in the system, though they may still experience transient packet drops in the network as in other hosts.

In this section, we consider how to integrate these proxies into the framework of LER by taking advantage of their reliability to strength the overall error recovery. We know which hosts are proxies in the system. Though these proxies are not essential in LER, we show that only a small fraction of them (say, 10–20%) would greatly improve the system performance.

To integrate the proxies into LER, we put them into a separate plane to form a "reliable plane." Every node finds the closest proxy on this plane as its recovery neighbor and put the proxy in its PRL (the number of nodes in the PRL hence becomes $N(\omega-1)+2$, where, as before, ω is the number of "unreliable" planes in the network and the "2" is due to a reliable proxy and the origin.) The PRL is then ordered as discussed before.

Fig. 5 shows an example of reliable plane with $\omega = 2$. Users are divided into two planes indicated by squares and circles. The reliable proxies, indicated by triangles, form a reliable plane themselves. Hosts A and B identify proxy R as one of their recovery neighbors.

Note that, in general, the number of proxies is much lower than the group size. Therefore, a node is more likely to request retransmission from other nodes before requesting a proxy. This greatly relieves the implosion at the proxies. Since the proxies have low packet loss and likely closer to the hosts than the origin, they provide a good alternative for retransmission.

B. Packet Loss Due to Membership Changes

In this section, we present how LER recovers packet loss arisen from dynamic membership changes. After a member joins or leaves, it often takes some time for a new tree to form, during which packets may be lost. LER can be used to recover packet loss during tree reconfiguration. Note that our approach is independent on how the underlying ALM protocol reconfigures its tree.

In the following, we discuss the join and leave/failure processes for LER:

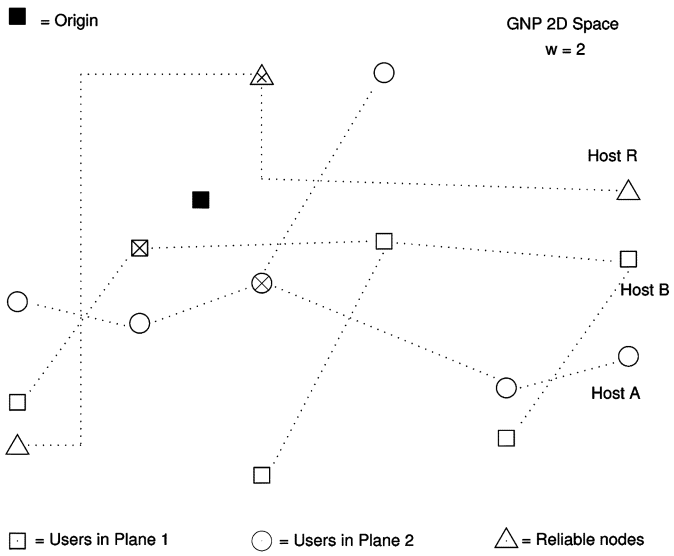


Fig. 5. Plane formed by reliable proxies (indicated by triangles) with $\omega = 2$.

- Member joins

  For LER(RNS), the PRL of the joining node can be obtained by random ping as mentioned before. For LER(GNP), a new member first computes its network coordinate, and is randomly assigned to one of the planes. The coordinate is then either forwarded to a central server (the centralized algorithm) or used in forming the DT mesh (the distributed algorithm).

  Streaming to the new node starts when the new tree is formed. Once the new node knows its parent and children, it updates its PRL with the aforementioned recovery identification processes. Whenever the tree is reconfigured, the affected node reconstructs its PRL.
- Member leaves/failures

  There are two types of leaving: graceful leaving and failure (treated as sudden leaving without notification). A gracefully leaving node notifies its children of its leaving, while a failure node can be detected by the absence of several successive packets in the stream. In any case, packet loss occurs when a node discovers that its parent has left the system. Under this circumstance, it *temporarily* attaches to its recovery neighbor to enter a transient state before a new plane tree is formed.

  Fig. 6 illustrates the idea for two planes. The parent of host A in plane 2 leaves the system [Fig. 6(a)]. Host A then temporarily attaches to its recovery neighbor, host B, to get its packets and to recover any packet lost during the reconfiguration of the plane tree [Fig. 6(b)]. One may picture that Host B temporarily adopts the subtree rooted at Host A. Once a new tree is formed, the node detaches itself from its recovery neighbor in the other plane.

  Note that only the children (not all its descendants) of a failure node need to be temporarily adopted by their recovery neighbors. Once the tree is reconfigured in the plane and the nodes detach from their recovery neighbors, the PRL in the affected nodes is then reconstructed.

  Note that the adoption of children from other planes would increase the forwarding load of a node, particularly

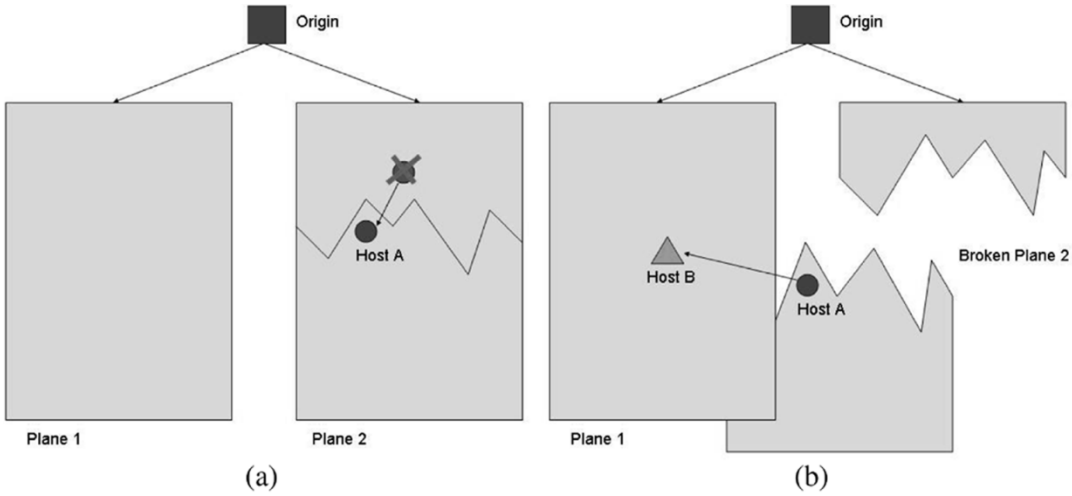


Fig. 6. Node leaving/failure in LER. (a) A leaving/failure node leads to its downstream node, Host A, high loss rate. (b) Host A temporarily attaches to its recovery neighbor, B, while tree is being reconfigured.

when the number of adoptions is large. To relieve its load, a node may have a certain (branch-out) degree constraint and refuses a temporary adoption when it is beyond the constraint. Furthermore, a node may detach an adoption if it temporarily experiences high load. For that node, it can then simply request another node from its PRL for itself to be adopted.

IV. Illustrative Simulation Results

In this section, we present illustrative simulation results for LER using Internet-like topologies. We run many experiments on the system, and the results presented here represent the general qualitative conclusions of the study. We first discuss the experimental setup in Section IV-A. Then we present the performance characteristics of LER in Section IV-B. In Section IV-C, we compare LER with other schemes and show its performance in the presence of reliable proxies.

A. Simulation Setup

We generate a number of *Transit Stub* topologies with the Georgia Tech random graph generator (GT-ITM) [48]. The generated topologies are a two-layer hierarchy of transit networks (with four transit domains, each with 16 randomly-distributed routers on 1024×1024 grid) and stub networks (with 64 domains, each with 15 randomly-distributed routers on a 32×32 grid). A host is connected to a stub router via a LAN (of a 4×4 grid), with a 1 ms delay to the router attached. For ALM tree, we use DT as our mesh construction and compass routing to multicast data as given in [17].

Packets are multicast using ALM and are randomly dropped in physical links of the network in accordance with the parameters based on real measurements [49]: with probability 0.95 the loss rate of a link is uniformly distributed between 0 and $\alpha\%$, and with probability 0.05 the loss rate is uniformly distributed between $5\alpha\%$ to $10\alpha\%$, where α is a scaling constant affecting the link loss in our simulation (real network measurements in [49] indicate $\alpha = 1$). In terms of failure model, we consider that a node is in failure state with a certain probability independent of each other. There are two kinds of packets, data packets (of size 1024 bytes) and control packets for retransmission request (of size 64 bytes).

We are interested in the following performance measures.

- Physical link stress (PLS), defined as the number of identical copies of a packet traversing a physical link. Besides its average, we are also interested in its distribution in terms of 95th and 90th percentiles.
- Relative delay penalty (RDP), defined as the ratio of the delay in the overlay with the shortest-path delay in unicast.
- Residual loss rate, defined as the overall loss rate for all packets within a certain deadline δ s after error is detected.
- Retransmission overhead, defined as the total traffic generated, normalized by the size of the data packet, for each lost packet. This includes control packets, retransmitted but lost packets, and the final correctly-received packet. The overhead is always greater than one. For PRM (discussed later), this includes the duplicated packets.

In our simulation, we use GNP to estimate nodal coordinates (A distributed alternative of GNP which performs similarly is Vivaldi). Unless otherwise stated, we use the baseline or default parameters as follows: LER(GNP), the number of hosts is 128, $\omega = 4, N = 2, \alpha = 1$, failure rate of 5%, and $\delta = 2$ s. In case of LER(RNS), we default $M = 10$. With these parameters, the average node-to-node delay is 1.24 s, and node-to-origin delay is 1.23 s. In the absence of node failures, the network has end-to-end loss rate without error recovery of 6.93%. We will see that LER(GNP) is able to cut the loss rate to less than 1%.

B. LER Performance

We show LER(GNP) performance in terms of RDP and PLS in Fig. 7, given ω. As shown in Fig. 7(a), RDP increases with the number of hosts due to deeper ALM trees. It decreases with the number of planes due to shorter trees. Fig. 7(b) shows that PLS increases with the number of hosts, because more packets are duplicated in the network and they may traverse the same link. PLS increase with the number of planes, because multiple parallel trees in general are less efficient than a single one. The

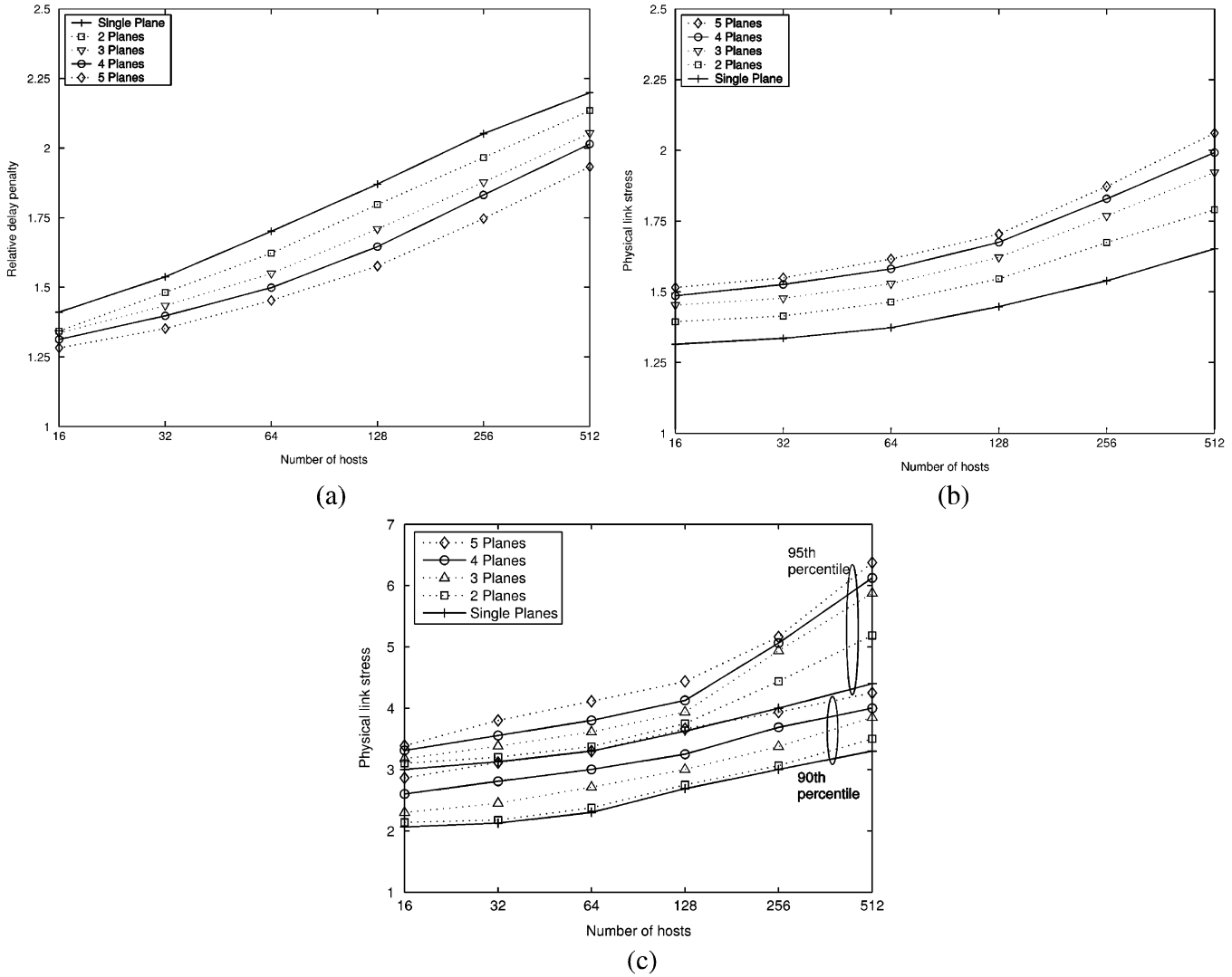


Fig. 7. LER performance versus the number of hosts. (a) Average RDP. (b) Average PLS. (c) 90th and 95th percentiles of PLS.

increase in PLS, however, is not so high; in general, LER introduces slightly higher (5–20%) higher stress as compared with the single tree approach. Fig. 7(c) shows the PLS distribution in terms of its 90th and 95th percentiles. The stress is not much higher than the average, showing that most of the link stress is concentrated around the average. For a large number of planes, some links may have relatively high stress, because packets from different planes may cross each other in underlay. In summary, Fig. 7 shows that LER achieves better RDP with some cost in stress as compared with single-tree approach.

We show in Fig. 8 residual loss rate versus N. Clearly, there is an optimal N to minimize the residual loss rate. This is because when N is too small, there are too few recovery neighbors and the recovery is not effective. On the other hand, when N is too high, the loss rate among recovery neighbors is likely to be correlated and a node may need to try many times in order to recover its loss, leading to ineffective recovery. Note that the loss rate drops sharply when N increases from 1 to 2 and stays relatively flat beyond that. In order to balance complexity and performance, N should be low but greater than one. We see that $N = 2$ is a good value to use.

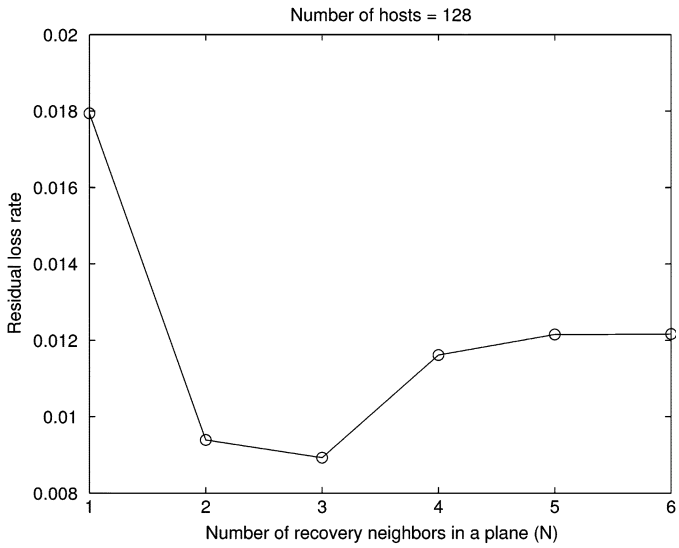


Fig. 8. Residual loss rate versus N for LER(GNP).

We show in Fig. 9 the residual loss rate of LER(RNS) versus M (the set of random neighbors). The loss rate decreases mono-

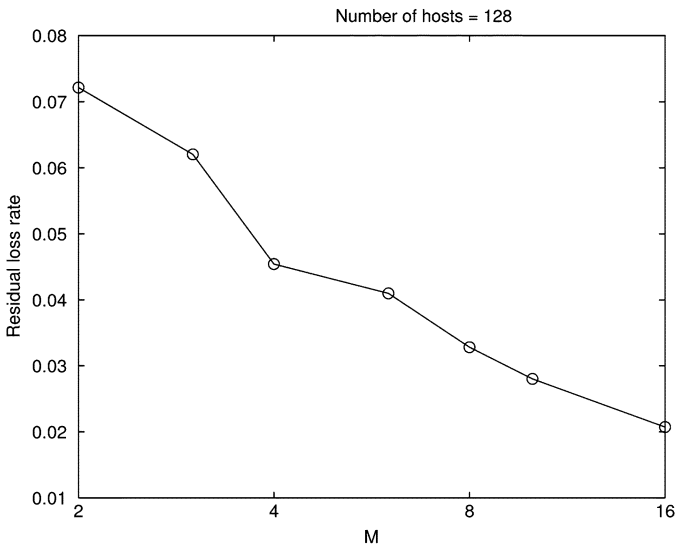


Fig. 9. Residual loss rate for LER(RNS) versus M.

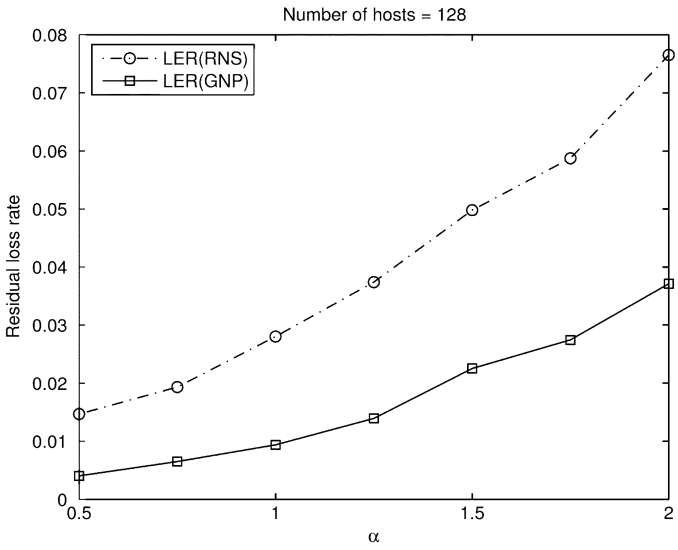


Fig. 11. Residual loss rate versus α.

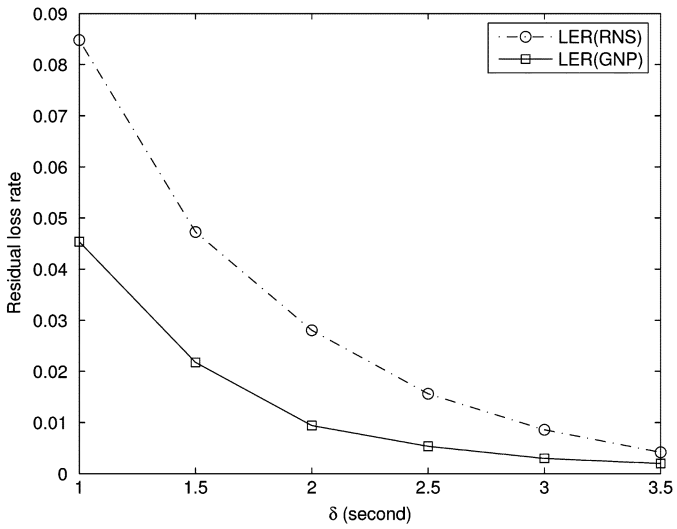


Fig. 10. Residual loss rate versus δ.

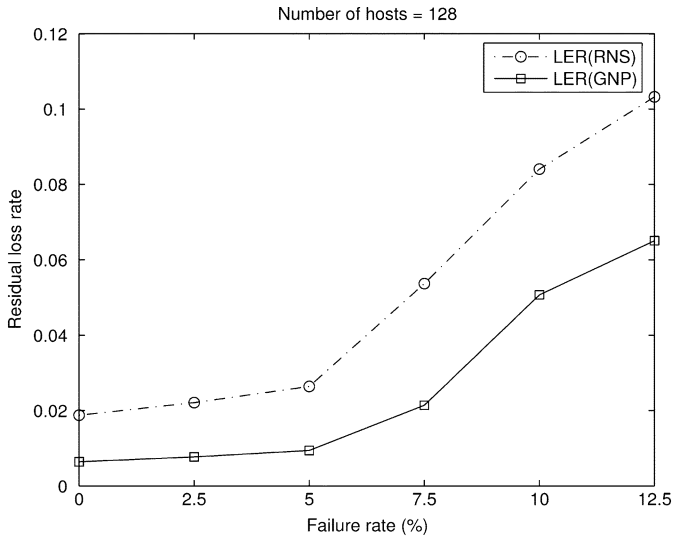


Fig. 12. Residual loss rate versus failure rate.

tonically with M, because the system has a higher probability to find close recovery neighbors. However, a large M implies more ping overheads to find recovery neighbors and hence higher complexity. Note that from the figure M does not need to be high to achieve low loss. In fact, if a node is interested in the closest 10% of the nodes, on average it only needs to randomly ping $M = 1/0.1 = 10$ hosts.

We next examine the sensitivity of parameters in LER. In Fig. 10, we show the residual loss rate versus retransmission deadline δ for LER(RNS) and LER(GNP). Clearly, the higher δ is, the lower is the loss rate. The figure shows that LER is very effective: for a delay of merely a few seconds, the loss rate is negligible. LER(GNP) performs better than LER(RNS) because it can find closer neighbors (with some cost of complexity).

We show in Fig. 11 the residual loss rate versus α, the scaling factor of link loss. The residual loss rate increases with α, as the link loss increases with α. Note that if link loss is doubled (by doubling α), the residual loss rate more than doubles. This is mainly because high link loss increases packet loss, which in turn reduces the efficiency of error recovery. As α decreases, there is a loss "floor" due to node failures.

We examine the influence of nodal failures on residual loss in Fig. 12. As nodal failure increases, residual loss rate first remains rather flat and then increases quickly. This is because a high failure reduces the probability of, and hence efficiency of, recovering from one's neighbors. When failure rate is low, LER is quite effective to achieve low residual loss.

We finally show in Fig. 13 the effect of the number of planes ω on the performance of LER. The residual loss rate decreases to a low value as ω increases. This is because when ω is low, tree tends to be deep and hence nodes generally suffer higher loss. Furthermore, for low ω, a node has few recovery neighbors and hence the origin often has to retransmit the lost packets, making the recovery expensive and not efficient. As ω increases, loss rate reduces due to more recovery neighbors and more effective recovery. For large ω, LER(RNS) performs as well as

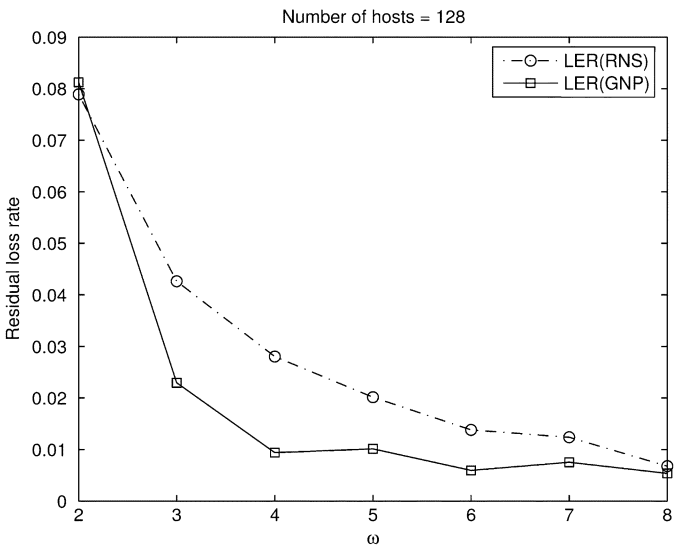


Fig. 13. Residual loss rate versus the number of planes.

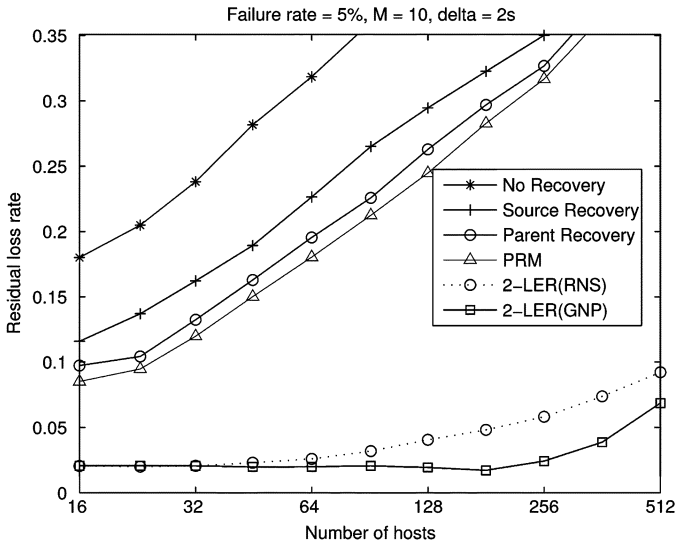


Fig. 14. Comparison of residual loss rate for different recovery schemes.

LER(GNP) because RNS can find many close recovery neighbors as in GNP. Due to the sharp decrease in loss rate, the figure shows that we do not need a high ω (around 4 is enough) to achieve good performance.

C. Scheme Comparison and the Presence of Reliable Proxies

In this section, we compare LER with the following schemes, the first two of which being variants of vertical recovery.

- Source Recovery

  A node always asks the origin for retransmission upon detecting an error. A strength of this scheme is that the origin is certain to have the lost packet. However, as mentioned before, this scheme suffers from source implosion. Furthermore, if the source is far away, the recovery delay can be high.
- Parent Recovery

  A failed node always requests retransmission from its parent. Whenever a lost packet is recovered, the node delivers it immediately downstream. This recovery is simple and effective. However, it cannot deal with node failure: if an upstream node fails or leaves the system, all downstream nodes can no longer recover their errors and would experience service outage.
- Probabilistic Resilient Multicast (PRM) [11]

  PRM employs two mechanisms to reduce the residual loss rate for streaming service, namely *randomized forwarding* and *triggered NAKs*. For randomized forwarding, each node introduces a certain number r of redundant connections with other nodes and with probability β forward a packet in each of these connections. For triggered NAK's, a failed host performs retransmission with the parent (the same idea as parent recovery). We set $r = 3$ and $\beta = 0.01$ because they are reasonably good parameters in the scheme. As compared with parent recovery, PRM addresses the problem of node failure. However, since PRM duplicates packets in its randomized forwarding, it incurs extra retransmission overhead in the network.

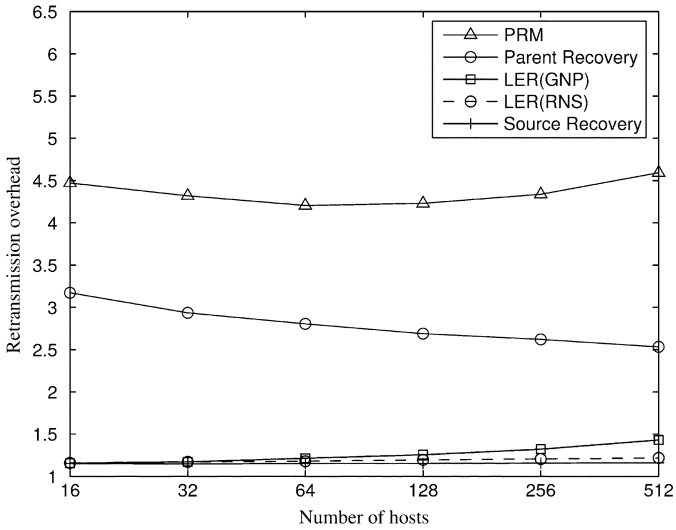


Fig. 15. Average retransmission overhead per lost packet versus number of hosts.

We compare all the schemes in Fig. 14. Also shown is the loss rate without any error recovery (the worst case). Generally the loss rate increases with the number of hosts, because the tree is deeper and packets are more likely to be lost. The performance without loss recovery is clearly unacceptable. Source recovery is better, and parent recovery performs even better than source recovery, due to its shorter distance to recovery nodes. PRM cuts the loss rate further as compared with parent recovery. LER achieves by far the lowest loss rate among all the schemes. LER(GNP) performs better than LER(RNS) due to its better selection on recovery neighbors.

We compare the retransmission overhead of all the schemes in Fig. 15. PRM incurs high overhead due to its constant duplicate packets in randomized forwarding. Source recovery has the lowest overhead, because the source always has the packet requested. LER achieves similar level of overhead as source recovery without implosion problem, but substantially lower loss rate.

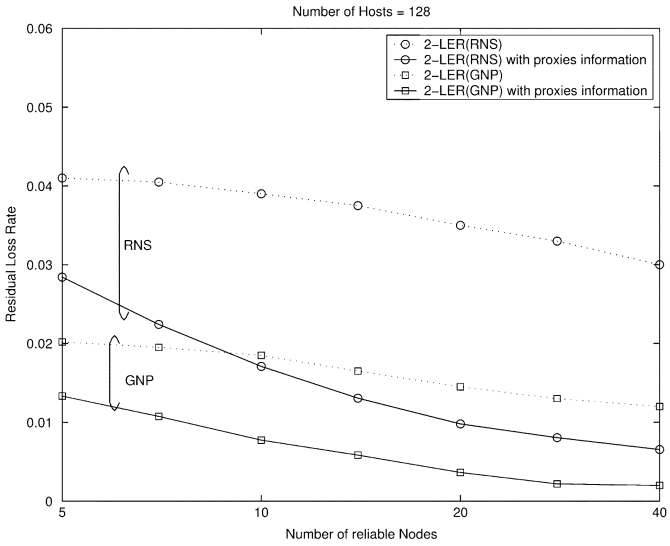


Fig. 16. Residual loss rate versus the number of reliable nodes.

We finally examine the effect of permanent proxies in LER. In Fig. 16, we show the residual loss rate versus the number of reliable proxies. In the figure, we also compare the case where the proxies are treated the same way as hosts and hence they themselves do not form a separate reliable plane (upper dotted line). As the number of proxies increases, the residual loss rate decreases. Clearly, it is beneficial to put all proxies in a separate plane and the presence of proxies substantially reduces the loss rate. Note that only a small fraction of proxies (10–20%) is enough to bring the loss rate to a low value.

V. Conclusion

In this paper, we propose and investigate *lateral error recovery* (LER) to recover packet loss in application-level multicast (ALM). We consider streaming applications where there is a deadline within which lost packets have to be retransmitted. In order to guarantee a certain level of quality, the loss rate after retransmission (i.e., residual loss rate) has to be low.

In LER, nodes are divided into planes, which form independent ALM trees. A node recovers its losses by retransmission from some nearby recovery neighbors of other planes. Since data is sent along the planes independently, the recovery neighbors have low error correlation with the failed node, leading to effective retransmission. We propose two variants of LER, one based on network coordinates and another based on random neighbor selection (RNS). We also describe how to make use of permanent proxies to further improve recovery capability, where the proxies themselves form a separate reliable plane.

LER is simple and effective. As compared with source recovery, parent recovery and probabilistic resilient multicast (PRM), LER achieves substantially the lowest residual loss with good tree performance in terms of RDP and stress. We have also quantified the computational complexity of LER. Our analysis on recovery delay agrees well with simulation. Our result also shows that only a small fraction (10–20%) of proxies in the system is enough to cut the loss rate further to a low value.

Appendix
Analysis of Recovery Delay of LER

In this section, we present an analysis on recovery delay of LER. We derive the average recovery delay defined as the average delay for a node to recover its error. We further define *normalized recovery delay* as the average recovery delay for all nodes divided by the average source recovery delay for all nodes.

LER, despite of its simplicity and ease of implementation, does not readily lend itself to analysis. As a matter of fact, its performance analysis capturing every detail of its operation is quite complicated, mainly due to the fact that packet loss among the nodes is correlated depending upon the network location and tree position of the nodes. In order to keep the analysis tractable, we need to make the following assumptions.

- Symmetric assumption: We assume that the network as seen by any node is the same. Therefore, the performance of an arbitrary node reflects and represents the overall network performance. With this assumption, our analysis can be greatly simplified: we may choose any node in the network and by analyzing its performance, the network performance is obtained. This assumption is often used for symmetric regular networks. Since LER network is tree-based, the assumption may not hold very well. As we show later, our results are pretty good despite of this assumption.
- Identical and independence assumption: We assume that each node has the same retransmission loss p_e as any of its neighbors, independent of each other. The retransmission loss is due to two reasons: the loss due to upstream nodes at the retransmission neighbor, and the network loss between the requester and retransmission neighbor. This assumption clearly is over-simplifying because nodes toward the leaves of the tree are expected to experience higher loss than the ones closer to the root, and nodes farther apart experience higher network loss.
- Far origin/Close neighbor assumption: We assume that the origin is always the farthest among all the recovery neighbors of a node. Recall that a node ranks its $N(\omega-1)$ neighbors in PRL in terms of their turnaround time Γ_{ij} (2). Therefore, the origin may not be the last one in the sequence. However, we expect that the assumption is reasonably good given that the origin is usually far away from a node which always find its recovery neighbor in its vicinity. This assumption also leads to somewhat pessimistic result.

  Associated with this assumption is close neighbor assumption. As we assume that the neighbors are close, we need $M > N$ in LER(RNS).
- Large δ assumption: Because the analysis on the distribution of recovery delay is intractable, we analyze the expected recovery delay given that δ is large (i.e., $\delta \rightarrow \infty$). Our analysis hence shows how long a node takes to recover its error completely.

Using the above assumptions, we analyze the normalized recovery delay as follows. Consider an arbitrary node in the network. Let T be its average recovery delay, and S be the av-

erage recovery delay if it requests retransmission solely from the source (i.e., average source recovery delay). According to our definition and using the symmetric assumption, the normalized recovery delay is given by

$$\text{Normalized recovery delay} = \frac{T}{S}. \tag{5}$$

The node sorts the turnaround time of its recovery neighbors in ascending order. Let

$$K \equiv (\omega - 1)N$$

be the number of recovery neighbors of the node, and $x_1, x_2, \ldots, x_K$ be the sequence of the average turnaround times of these neighbors. Clearly

$$x_1 \le x_2 \le \cdots \le x_K$$

and, from (2), we have

$$x_j = \mathrm{RTT}_j + E[w_j].$$

(Without causing any confusion, we have dropped the index i for brevity as the arbitrary node index i is implicit here.)

Because the node requests its neighbors according to the sorted sequence, using independence assumption, the average recovery delay of the node is given by

$$\begin{aligned} T = {} & (1-p_e)x_1 \\ & + p_e(1-p_e)(x_1+x_2) \\ & + \cdots \\ & + p_e^{K-1}(1-p_e)(x_1+x_2+\cdots+x_K) \\ & + p_e^K\left(S + \sum_{j=1}^{K} x_j\right). \end{aligned} \tag{6}$$

The above equation may be equivalently written as

$$\begin{aligned} \text{Normalized recovery delay} = {} & \frac{T}{S} \\ = {} & (1-p_e)\sum_{j=1}^{K}\left(p_e^{j-1}\sum_{l=1}^{j}\frac{x_l}{S}\right) \\ & + p_e^K\left(1 + \sum_{j=1}^{K}\frac{x_j}{S}\right). \end{aligned} \tag{7}$$

Due to the analytic complexity of LER(GNP) in obtaining x_j, we analyze only LER(RNS) where neighbors are randomly chosen in each plane. In this case, we assume that RTT_j is equally spaced from 0 to S because they are randomly chosen and this gives an approximate closed-form expression for RTT_j, i.e.,

$$\frac{x_j}{S} = \frac{j}{K+1} + \frac{E[w_j]}{S}.$$

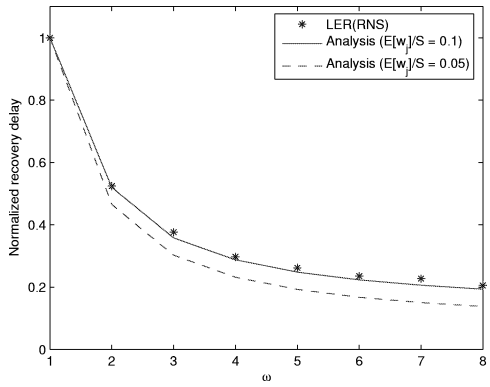


Fig. 17. Normalized recovery delay versus ω for LER(RNS), given $E[w_j]$. ($N = 2, M = 10$, number of hosts $128, p_e = 0.1$, and no node failure.)

$E[w_j]/S$ is expected to be low, because in PRL, those nodes among the lowest w_j are likely to be requested first. However, analytically obtaining $E[w_j]/S$ is difficult, but from our simulation a value of 0.05–0.1 is reasonably good to use. We validate our model using simulation. We show in Fig. 17 a typical plot of normalized recovery delay versus ω. For $\omega = 1$, LER(RNS) degenerates to the source recovery and hence its normalized delay starts from 1. We use parameters $N = 2, M = 10$, number of hosts $128, p_e = 0.1$, no node failure, with $E[w_j]/S = 0.05$ and 0.1. (Details of the network topologies used in the simulation are described in Section IV.A.) The figure shows that our analytic curves based on (7) agree well with simulation.

REFERENCES

[1] V. O. K. Li and Z. Zhang, "Internet multicast routing and transport control protocols," *Proc. IEEE*, vol. 90, no. 3, pp. 60–391, Mar. 2002.

[2] K. Sripanidkulchai, B. Maggs, and H. Zhang, "An analysis of live streaming workloads on the internet," in *Proc. Internet Measurement Conf. (IMC)*, Taormina, Sicily, Italy, Oct. 2004.

[3] K. Sripanidkulchai, A. Ganjam, B. Maggs, and H. Zhang, "The feasibility of supporting large-scale live streaming applications with dynamic application end-points," in *Proc. ACM SIGCOMM*, Portland, OR, Aug. 30–Sep. 1 2004.

[4] R. Keller, S. Choi, D. Decasper, M. Dasen, G. Fankhauser, and B. Plattner, "An active router architecture for multicast video distribution," in *Proc. IEEE INFOCOM 2000*, vol. 3, Israel, Mar. 2000, pp. 1137–1146.

[5] H. M. Radha, M. van der Schaar, and Y. Chen, "The MPEG-4 fine-grained scalable video coding method for multimedia streaming over IP," *IEEE Trans. Multimedia*, vol. 3, no. 1, pp. 53–68, Mar. 2001.

[6] H. Oouchi, K. Takahashi, H. Nagata, and K. Kamasawa, "Multi-rate control method using layered content," in *Proc. IEEE 2005 Symp. Applications and the Internet*, Jan. 31–Feb. 4 2005, pp. 311–317.

[7] C.-M. Huang and P.-C. Liu, "A ubiquitous 1-to-k media streaming architecture using the IPv4/IPv6 multicast transition gateway approach," in *Proc. 18th Int. Conf. Advanced Information Networking and Applications*, vol. 2, 2004, pp. 204–207.

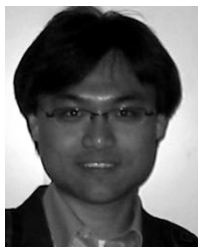
[8] L. Subramanian, I. Stoica, H. Balakrishnan, and R. H. Katz, "OverQoS: Offering QoS using overlays," in *Proc. 1st HotNets Workshop*, Oct. 2002.

[9] T. S. E. Ng and H. Zhang, "Predicting internet network distance with coordinates-based approaches," in *Proc. IEEE INFOCOM 2002*, New York, Jun. 2002.

[10] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A decentralized network coordinate system," in *Proc. ACM SIGCOMM*, Portland, OR, USA, Aug. 30–Sep. 1 2004.

[11] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan, "Resilient multicast using overlays," in *ACM SIGMETRICS 2003, Int. Conf. on Measurement and Modeling of Computer Systems*, San Diego, CA, Jun. 2003.

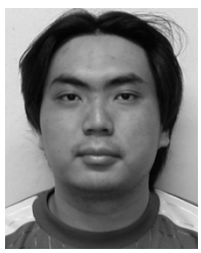
[12] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel, "ALMI: An application level multicast infrastructure," in *Proc. 3rd USENIX Symp. Internet Technology and Systems*, San Francisco, CA, Mar. 2001.
[13] P. A. C. Venkata, N. Padmanabhan, and H. J. Wang, "Supporting heterogeneity and congestion control in peer-to-peer multicast streaming," in *Proc. 3rd Int. Workshop on Peer-to-Peer Systems (IPTPS'04)*, Feb. 2004.
[14] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *Proc. ACM SIGCOMM*, Pittsburgh, PA, Aug. 2002.
[15] P. Francis. (2000, Apr.) Yoid: Your Own Internet Distribution. ACIRI. [Online]. Available: http://www.isi.edu/div7/yoid/.
[16] Y. H. Chu, S. G. Rao, S. Seshanand, and H. Zhang, "A case for end system multicast," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 8, pp. 1456–1471, Oct. 2002.
[17] J. Liebeherr, M. Nahas, and W. Si, "Application-layer multicasting with Delaunay triangulation overlays," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 8, pp. 1472–1488, Oct. 2002.
[18] W.-C. Wong and S.-H. Chan, "Improving delaunay triangulation for application-level multicast," in *Proc. IEEE GLOBECOM 2003*, San Francisco, CA, Dec. 2003, pp. 2835–2839.
[19] M. Castro, P. Druschel, A.-M. Kermarec, and A. I. T. Rowstron, "Scribe: A large-scale and decentralized application-level multicast infrastructure," in *IEEE J. Sel. Areas Commun.*, vol. 20, Oct. 2002, pp. 1489–1499.
[20] D.-K. Kim, K.-I. Kim, K.-D. Kim, I.-S. Whang, and S.-H. Kim, "Scheme for scalable ALM architecture based on topology-awareness," in *IEEE Consumer Communications and Networking Conf. 2004*, Las Vegas, NV, Jan. 2004.
[21] W.-P. K. Yiu, K.-F. S. Wong, and S.-H. G. Chan, "Bridge-node selection and loss recovery in island multicast," in *Proc. IEEE ICC*, Seoul, Korea, May 16–20, 2005.
[22] X. Jin, Y. Wang, and S.-H. G. Chan, "Fast overlay tree based on efficient end-to-end measurements," in *Proc. IEEE ICC*, Seoul, Korea, May 16–20, 2005.
[23] K.-L. Cheng, K.-W. R. Cheuk, and S.-H. G. Chan, "Implementation and performance measurement of an island multicast protocol," in *Proc. IEEE ICC*, Seoul, Korea, May 16–20, 2005.
[24] W.-P. Yiu and S.-H. Chan, "SOT: Secure overlay tree for application-layer multicast," in *Proc. IEEE Int. Conf. Communications (ICC)*, Paris, France, Jun. 2004, pp. 1451–1455.
[25] K.-W. Cheuk, S.-H. Chan, and J. Lee, "Island multicast: The combination of IP-multicast with application-level multicast," in *Proc. IEEE Int. Conf. on Communications (ICC)*, Paris, France, Jun. 2004, pp. 1441–1445.
[26] A. El-Sayed, V. Roca, and L. Mathy, "A survey of proposals for an alternative group communication service," *IEEE Network*, vol. 17, no. 1, pp. 46–51, Jan./Feb. 2003.
[27] M. Castro, M. Jones, A. Kermarrec, A. Rowstron, M. Theimer, H. Wang, and A. Wolman, "An evaluation of scalable application-level multicast built using peer-to-peer overlays," in *Proc. IEEE INFOCOM 2003*, vol. 2, San Francisco, CA, Apr. 2003, pp. 1510–1520.
[28] K.-F. S. Wong, S.-H. G. Chan, W. Wong, Q. Zhang, W.-W. Zhu, and Y.-Q. Zhang, "Lateral error recovery for application-level multicast," in *Proc. IEEE INFOCOM 2004*, Hong Kong, Mar. 2004, pp. 2708–2718.
[29] J. Cai, Q. Zhang, W. Zhu, and C. Chen, "An FEC-based error control scheme for wireless MPEG-4 video transmission," in *Proc. Wireless Communications and Networking Confernce'00*, vol. 3, 2000, pp. 1243–1247.
[30] P. Frossard and O. Verscheure, "Joint source/FEC rate selection for quality-optimal MPEG-2 video delivery," in *IEEE Trans. Image Process.*, vol. 10, Dec. 2001, pp. 1815–1825.
[31] R. Kermode, "Scoped hybrid automatic repeat request with forward error correction (SHARQFEC)," in *Proc. ACM SIGCOMM'98*, Sep. 1998, pp. 278–289.
[32] T.-W. A. Lee, S.-H. G. Chan, Q. Zhang, W.-W. Zhu, and Y.-Q. Zhang, "Allocation of layer bandwidths and FEC's for video multicast over wired and wireless networks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 12, pp. 1059–1070, Dec. 2002.
[33] T. Noguchi, M. Yamamoto, and H. Ikeda, "Reliable multicast protocol applied local FEC," in *Proc. IEEE ICC 2001*, vol. 8, 2001, pp. 2348–2353.
[34] W.-T. Tan and A. Zakhor, "Video multicast using layered FEC and scalable compression," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 3, pp. 373–386, Mar. 2001.
[35] T. Lestayo, M. Fernández, and C. López, "Adaptive approach for FEC reliable multicast," *Electron. Lett.*, vol. 37, no. 22, pp. 1333–1335, Oct. 2001.
[36] M. Grossglauser, "Optimal deterministic timeouts for reliable scalable multicast," *IEEE J. Sel. Areas Commun.*, vol. 15, no. 3, pp. 422–433, Apr. 1997.
[37] D. Towsley, J. Kurose, and S. Pingali, "A comparison of sender-initiated and receiver-initiated reliable multicast protocols," *IEEE J. Sel. Areas Commun.*, vol. 15, no. 3, pp. 398–406, Apr. 1997.
[38] J. Nonnenmacher, E. W. Biersack, and D. Towsley, "Parity-based loss recovery for reliable multicast transmission," *IEEE/ACM Trans. Networking*, vol. 6, no. 4, pp. 349–361, Aug. 1998.
[39] J. Nonnenmacher and E. W. Biersack, "Scalable feedback for large groups," *IEEE/ACM Trans. Networking*, vol. 7, no. 3, pp. 375–386, Jun. 1999.
[40] J. Gemmell, T. Montgomery, T. Speakman, and J. Crowcroft, "The PGM reliable multicast protocol," *IEEE Network*, vol. 17, no. 1, pp. 16–22, Jan./Feb. 2003.
[41] C.-G. Liu, D. Estrin, S. Shenker, and L. Zhang, "Local error recovery in SRM: Comparison of two approaches," *IEEE/ACM Trans. Networking*, vol. 6, no. 6, pp. 686–699, Dec. 1998.
[42] M. S. Lacher, J. Nonnenmacher, and E. W. Biersack, "Performance comparison of centralized versus distributed error recovery for reliable multicast," *IEEE/ACM Trans. Networking*, vol. 8, no. 2, pp. 224–238, Apr. 2000.
[43] M. Calderon, M. Sedano, A. Azcorra, and C. Alonso, "Active network support for multicast applications," *IEEE Network*, pp. 46–52, May/Jun. 1998.
[44] S. K. Kasera, S. Bhattacharyya, M. Keaton, K. Kiwior, S. Zabele, J. Kurose, and D. Towsley, "Scalable fair reliable multicast using active services," *IEEE Network*, pp. 48–57, Jan./Feb. 2000.
[45] B. Whetten and G. Taskale, "An overview of reliable multicast transport protocol II," *IEEE Network*, pp. 37–47, Jan./Feb. 2000.
[46] E. Kim, S. G. Kang, and J. Choe, "A router-assisted session tree configuration mechanism for reliable multicast," *IEEE Commun. Lett.*, vol. 6, no. 9, pp. 464–466, Sep. 2002.
[47] S. Fortune, "A sweepline algorithm for Voronoi diagrams," *Algorithmica*, vol. 2, pp. 153–174, 1987.
[48] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proc. IEEE INFOCOM 1996*, San Francisco, CA, 1996.
[49] V. N. Padmanabhan and L. Qiu, "Network tomography using passive end-to-end measurements," *DIMACS Workshop on Internet and WWW Measurement*, Feb. 2002.



W.-P. Ken Yiu (S'03) received the B.Eng. and M.Phil. degrees, both in computer science from The Hong Kong University of Science and Technology, Hong Kong, in 2002 and 2004, respectively. He is currently pursuing the Ph.D. degree with the Department of Computer Science at HKUST.

His research interests include computer networks, peer-to-peer systems, multimedia networking, and network security.

Mr. Yiu was awarded the Academic Achievement Medal from HKUST in 2002, and the Sir Edward Youde Memorial Fellowship from Sir Edward Youde Memorial Fund in 2005 and 2006. He is a student member of the IEEE Computer Society.



Simon K. F. Wong received the B.Eng. in computer engineering and M.Phil. degree in computer science from The Hong Kong University of Science and Technology (HKUST). His M.Phil. Thesis, supervised by Dr. Gary Chan, was on the error recovery mechanism in application-level multicast.

From 2002 to 2005, he was a Research Assistant in the Multimedia and Wireless Networking group, HKUST.

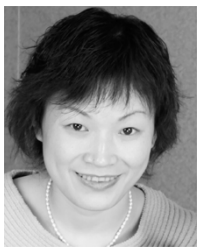
S.-H. Gary Chan (S'89–M'98–SM'03) received the B.S.E. degree (highest honor) in electrical engineering from Princeton University, Princeton, NJ, with certificates in applied and computational mathematics, engineering physics, and engineering and management systems, in 1993. He received the M.S.E. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, in 1994 and 1999, respectively, with a minor in business administration.

He is currently an Associate Professor with the Department of Computer Science, The Hong Kong University of Science and Technology, Hong Kong, and an Adjunct Researcher with the Microsoft Research Asia, Beijing, China. His research interest includes multimedia networking, peer-to-peer networks and wireless communications networks. He was a Visiting Assistant Professor in networking at the Department of Computer Science, University of California at Davis, from September 1998 to June 1999. During 1992–1993, he was a Research Intern at the NEC Research Institute, Princeton. He was a William and Leila Fellow at Stanford University during 1993–1994.

Dr. Chan currently serves as a Vice-Chair of IEEE COMSOC Multimedia Communications Technical Committee, and is a member of Tau Beta Pi, Sigma Xi, and Phi Beta Kappa. At Princeton, he was the recipient of the Charles Ira Young Memorial Tablet and Medal, and the POEM Newport Award of Excellence in 1993.

Wan-Ching Wong received the B.Sc and M.Sc. degrees, both in computer science, from Hong Kong University of Science and Technology (HKUST), Hong Kong, in 2001 and 2003, respectively.

He was an Assistant Researcher at HKUST until August 2001. His research interests are in Internet, peer-to-peer networks and aspect-oriented programming.



Qian Zhang (M'00–SM'04) received the B.S., M.S., and Ph.D. degrees from Wuhan University, China, in 1994, 1996, and 1999, respectively, all in computer science.

She joined Microsoft Research Asia, Beijing, China, in July 1999. She is currently the Research Manager of the Wireless and Networking group. She has published more than 80 refereed papers in international leading journals and key conferences in the arew of wireless/Internet multimedia networking, wireless communications and networking, and overlay networking. She is the inventor about 20 pending patents. Her current research interest includes seamless roaming across different wireless networks, multimedia delivery over wireless, Internet, next-generation wireless networks, and P2P network/ad hoc networks.

Dr. Zhang is a member of the Visual Signal Processing and Communication Technical Committee and the Multimedia System and Application Technical Committee of the IEEE Circuits and Systems Society. She is also a member and chair of QoSIG of the Multimedia Communication Technical Committee of the IEEE Communications Society. She is the Associate Editor for IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGIES. She is also serving as Guest Editor for a special issue on wireless video in *IEEE Wireless Communication Magazine*. She has also served in the technical committee of numerous IEEE and other international conferences. She has recently received TR, 100 (MIT Technology Review) world's top young innovator award.

Wenwu Zhu (S'92–M'97–SM'01) received the B.E. and M.E. degrees from National University of Science and Technology of China (USTC), in 1985 and 1988, respectively, the M.S. degree from Illinois Institute of Technology, Chicago, and the Ph.D. degree from Polytechnic University, Brooklyn, New York, in 1993 and 1996, respectively, all in electrical engineering.

He joined Communication Technology Lab China as Co-Director in September 2004. Prior to his current post, he was with Microsoft Research Asia, Beijing, China, first as a Researcher in the Internet Media Group and later as Research Manager of Wireless and Networking Group. From 1996 to 1999, he was with Bell Labs, Lucent Technologies, NJ, as a Member of Technical Staff during 1996–1999. From 1988 to 1990, he was with the Graduate School, USTC, and the Institute of Electronics, Chinese Academy of Sciences, Beijing. He has published over 180 refereed papers in the areas of wireless/Internet multimedia delivery, and wireless communications and networking. He participated activity in the IETF ROHC WG on robust TCP/IP header compression over wireless links. He is co-inventor of over 20 pending patents. His current research interest is in the area of wireless communication and networking, and wireless/Internet multimedia communication and networking.

Dr. Zhu has been on various editorial boards of IEEE journals such as Guest Editor for the PROCEEDINGS OF THE IEEE, Associate Editor for the IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON MULTIMEDIA, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY (TCSVT). He received the Best Paper Award in IEEE TCSVT in 2001. He currently is also the Chairman of IEEE Circuits and System Society Beijing Chapter and the Secretary of Visual Signal Processing and Communication Technical Committee. He is a member of Eta Kappa Nu, Multimedia System and Application Technical Committee and Life Science Committee in IEEE Circuits and Systems Society, and Multimedia Communication Technical Committee in IEEE Communications Society.

Ya-Qin Zhang (F'97) received the B.S. and M.S. degrees in electrical engineering from the University of Science and Technology of China (USTC) in 1983 and 1985, respectively, and the Ph.D. degree in electrical engineering from George Washington University, Washington, DC, in 1989. He had executive business training from Harvard University.

He is the Corporate Vice President of Microsoft Corporation in Redmond, Washington. He is currently responsible for product development of Microsoft's Mobile and Embedded Division, including WinCE operating system, Smartphone, PocketPC, and other Windows Mobile platform and devices. He was the Managing Director of Microsoft Research Asia, Microsoft's basic research arm in Asia-Pacific region. From 1994 to 1999. he was the Director of Multimedia Technology Laboratory at Sarnoff Corporation, Princeton, NJ (RCA Laboratories). He was with GTE (now Verizon) Corporation, Waltham, MA, from 1989 to 1994. He has published over 300 refereed papers in leading international conferences and journals. He has been granted over 50 US patents in digital video, Internet, multimedia, wireless and satellite communications. Many of the technologies he and his team developed have become the basis for start-up ventures, commercial products, and international standards. He served on the Board of Directors of several IT companies, including Sohu.com (NASQ: SOHU) — a top Chinese portal.

Dr. Zhang served as the Editor-in-Chief for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY. He was the Chairman of Visual Signal Processing and Communications Technical Committee of IEEE Circuits and Systems. He serves on the Editorial boards of seven other professional journals and over a dozen conference committees. He has been a key contributor to the ISO/MPEG and ITU standardization efforts in digital video and multimedia. He received many awards, including several industry technical achievement awards and IEEE academic awards.