Lateral Error Recovery for Application-Level Multicast

K.-F. Simon Wong S.-H. Gary Chan Wan-Ching Wong
Department of Computer Science
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong
{cssmw, gchan, wwilliam}@cs.ust.hk

Qian Zhang Wen-Wu Zhu Ya-Qin Zhang
Microsoft Research Asia
5F, Beijing Sigma Center, No. 49, ZhiChun Road
Haidian District, Beijing 100080, P.R. China

***Abstract*— We consider the delivery of reliable and streaming services using application-level multicast (ALM) by means of UDP, where packet loss has to be recovered via retransmission in a timely manner in order to offer high level of service. Since packets may be lost due to congestion, tree-reconfiguration or node failure, the traditional "vertical" recovery whereby upstream nodes retransmit the lost packet is no longer effective. We therefore propose and investigate lateral error recovery (LER). In LER, hosts are divided into a number of planes, each of which forms an independent ALM tree. Since the correlation of error among the planes is likely to be low, a node can effectively recover its error "laterally" from nearby nodes in other planes. We employ the technique of global network positioning (GNP) to map the hosts into a coordinate space and identify a set of close neighbors for error recovery by constructing a Voronoi diagram for each plane. We present centralized and distributed algorithm on how to construct the Voronoi diagrams.**

Using Internet-like topologies, we show via simulations that our system achieves low overheads in terms of relative delay penalty and physical link stress. For reliable service, lateral recovery greatly reduces the average recovery time as compared with vertical recovery schemes. For streaming applications, LER achieves much lower residual loss rate under a certain deadline constraint.

***Index Terms*— Application-level multicast (ALM), quality of service (QoS), error recovery, simulations, reliable service, streaming application**

This work was supported, in parts, by the Area of Excellence in Information Technology (AoE-IT) from the University Grant Council (AoE/E-01/99) and the Research Grant Council of the Hong Kong Special Administrative Region, China (HKUST6165/03E)

I. INTRODUCTION

Application-level multicast (ALM) has emerged as a promising technique to overcome current limitations in IP multicast for point-to-multipoint applications such as file distribution, video conferencing, movie streaming, etc. 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. We consider two types of services for ALM in this paper:

- Reliable service, where a server is to distribute files to a pool of users so that any packet lost has to be recovered [1]. The recovery time for a packet should be as low as possible.
- Streaming applications, where there is a certain play-out (or recovery) deadline δ within which the lost packets have to be recovered. In this case, the residual loss rate is an important issue.

Traditionally, most ALM research has been focusing on the *connectivity* among the hosts by addressing how messages are routed from one point (the server or origin) to all the other group members. In order for ALM to be successful, how to offer quality of service needs to be addressed [2]. In this work, we consider error recovery mechanism for ALM networks. Our objectives are to offer low recovery delay for reliable service or low residual loss rate for streaming applications, without compromising ALM tree performance (in terms of link stress and relative delay penalty).

Indeed, packets may be lost when they are forwarded from hosts to hosts. Such loss may be due to congestion in the underlying links. Furthermore, some nodes may leave the system or fail, leading to transient tree-reconfiguration and hence packet loss.[1] Since the aforementioned error conditions may persist for some time, recovery mechanism is necessary to mitigate the loss.

Using TCP from one host to another appears to be a feasible solution for reliable services. However, it may not achieve high throughput due to TCP backoff mechanism. The hosts at the leaves of the delivery tree may suffer from high delay, as a data segment has to be completely received before being forwarded downstream. Furthermore, it is not obvious to extend TCP in hop-by-hop, packet-by-packet manner for reliable service. It is also not applicable in streaming applications, since reliable connection is not required. We hence will focus on using UDP in this paper.

Most of the traditional ALM protocols tend to cluster or chain hosts close in network distance together to form a tree for data delivery. While this reduces end-to-end delay, it is not good for error recovery. Whenever there is an error upstream, all the downstream hosts will be affected. Since hosts close together are clustered, their losses are correlated, which makes retransmission ineffective. The problem becomes more severe as the number of hosts and hence tree depth increases, because the hosts towards the leaves of the delivery tree would suffer relatively high loss.

A natural way to deal with packet loss is to request the upstream hosts for retransmission, i.e. "vertical" retransmission. However such vertical recovery suffer from the following problems:

- High error correlation: As mentioned above, the errors of all downstream nodes are correlated. Therefore, when a node discovers an error, most likely its parent would also be in error. Given that it does not know where the error originates, the host may have to probe upstream for a number of times before its packets can be successfully retransmited. This incurs a long recovery delay.
- Implosion problem: Vertical recovery may lead to implosion if retransmission requests are not aggregated. If all the downstream nodes perform error recovery upstream, the parent where the error originates may be overwhelmed by retransmission requests.
- Outage due to node/link failure: Upon the failure of a node/link, there would be a temporary outage for all the downstream nodes due to tree-reconfiguration and hence vertival 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, the error correlation among nodes of different planes is greatly reduced, leading to retransmission efficiency.

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

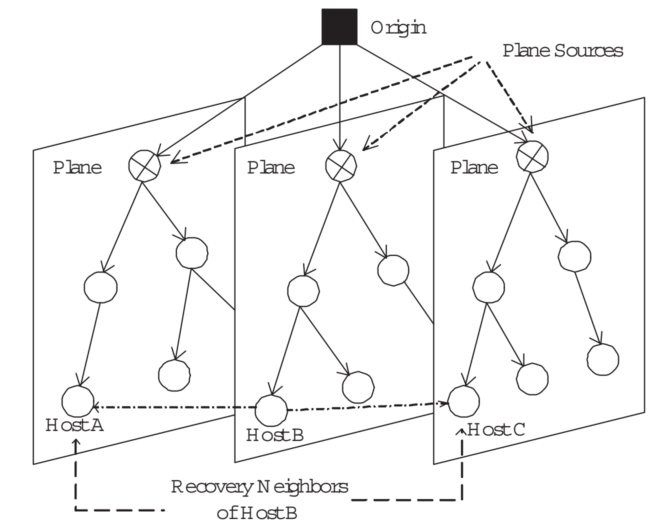


Fig. 1. Lateral error recovery using tree planes.

We show the architecture in Figure 1, where hosts are distributed into different planes with the source indicated as the "origin." The origin first sends the data to the plane sources, where ALM trees are formed among all the hosts in the planes. Consider an arbitrary host B, it first identifies some hosts in other planes (hosts A and C in the figure) as its recovery neighbors. Whenever an error occurs, host B selects some of its recovery neighbors for "lateral" retransmission. We 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 failure nodes, but adjacent "lateral nodes." This effectively reduces the error correlation and implosion problem.

We address the following two issues in the lateral recovery:

- Selection of plane sources
  Given the plane nodes, we consider choosing the nodes close to the origin as the plane sources. This approach reduces the end-to-end delay and physical link stress. To achieve this, we employ the technique called global network positioning (GNP) [3] to obtain host coordinates. Based on the GNP coordinates, the closest neighbor for each plane can be obtained by constructing Voronoi diagram. We discuss how to construct the Voronoi diagram by a centralized algorithm and a distributed algorithm based on Delaunay Triangulation (DT).
- Identification of one's recovery neighbors and, upon an error, the sequence with which the neighbors are

requested for retransmission.

A good choice in recovery neighbors leads to low recovery delay. We show how to identify the set of close recovery neighbors from the constructed Voronoi diagram. To recover loss, a node orders its neighbors according to the turn-around times in its retransmission attempt.

We compare our lateral recovery scheme with two variants of vertical recovery schemes, namely source recovery (recovery from the origin) and parent recovery (recovery from one's parent). For comparison, we have also simulated a recently proposed recovery scheme, the Probabilistic Resilient Multicast (PRM) [4]. By using Internet-like topologies, we show that LER achieves low overheads in terms of RDP and physical link stress. As compare with vertical recovery and PRM, LER reduces substantially the recovery delay for reliable services and residual loss rate for streaming services.

This paper is organized as follows. In Section II, we briefly discuss the related work. The operation of lateral error recovery is then discussed in Section III, followed by our simulation results in Section IV. We conclude in Section V.

II. Related Work

Many ALM protocols have been proposed in literature, such as NICE [5], ALMI [6], YOID [7], DT [8], Narada [9], and Scribe [10]. All these schemes consider the connectivity rather than error recovery issue and are based on a single plane. No error issue is discussed. Our approach is different from them by dividing the hosts into a number of planes, so that nearby hosts may effectively recover their error. Note that, we are not proposing any new ALM protocol and hence our work is complementary to them. An error recovery scheme named PRM (Probabilistic Resilient Multicast) has been recently proposed [4]. It adds some extra links in the ALM tree and packets are forwarded along these links with some probability. Though this approach reduces the error rate for streaming applications, however it cannot support error-free reliable service. Our LER scheme can offer much lower error rate and is suitable for reliable applications. The work on LER has been reported in [11]. However, the result PRM and simulation comparsion of different schemes have not been extensively explored.

There has been much work on providing reliable services over IP multicast networks. Our work differs from them in many ways. For example, much of the previous work focuses on the design of scalable feedback mechanisms to address the implosion problem at the origin [12], [13], [14], [15]. Recovery from the source is generally expensive in terms of delay and bandwidth. In ALM, since some other nodes may buffer the lost packets, the feedback does not need to go all the way to the source. This changes fundamentally how recovery mechanism can be designed as discussed in this paper. Recovering loss from local groups of users has also been discussed in [16], [17]. These groups are usually fixed or given without much control by the receivers. In ALM, the particular nodes to recover one's losses can be chosen to improve one's recovery probability. We show how this can be done in this paper. Using proxies to recover losses has also been investigated in [18], [19], [20], [21]. It generally involves intelligent placement of proxies so that recovery can be done by traversing upstream of the proxy tree. Our work differs by the absence of such recovery proxies and the recovery can be done *laterally* rather than vertically along the tree. All the aforementioned previous work on reliable multicast has treated the recovery tree as given and fixed, which is certainly the case in IP multicast. In ALM, however, the recovery mechanism can in fact be *engineered* for better loss recovery, as we show in this paper.

III. Lateral Error Recovery

In this section, we present the detailed operation of lateral error recovery (LER). First, hosts are assigned to anyone of the ω planes randomly when they join the multicast group. Due to this random assignment, hosts close together are likely distributed into different planes. This simple mechanism roughly balances the number of close nodes in each plane (note that LER does not require exact balanced distribution of nodes into the planes).

Next an ALM delivery tree for each plane is constructed. Most of the traditional ALM schemes can be applied here. (In our simulation, we have used an existing proposal called Delaunay Triangulation (DT) [8] to build the tree.) Then the plane sources[2] and the recovery neighbors are identified. The issue is to find the nodes that are close to the origin as plane source and close to a node as its recovery neighbors. Obviously, it is costly to measure the distance between every pair of nodes, because this causes massive pinging among the hosts. Instead, we estimate the host location by GNP coordinates and identify the close nodes by constructing Voronoi diagrams. In the next sections, we discuss how GNP is performed and on top of the GNP coordinates, we mention selection of plane sources and recovery neighbors.

We assume the origin marks each packet with increasing sequence number. Error packet is discovered by the

[2]We may need to identify the plane sources first before constructing the delivery tree for some source specific tree-building schemes,

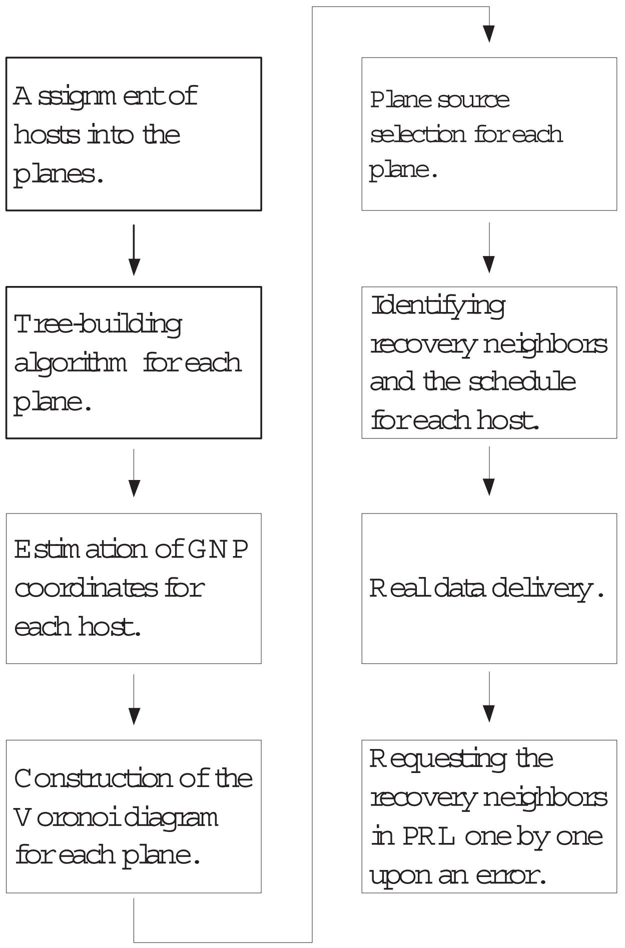

Assignment of hosts into the planes.
Tree-building algorithm for each plane.
Estimation of GNP coordinates for each host.
Construction of the Voronoi diagram for each plane.
Plane source selection for each plane.
Identifying recovery neighbors and the schedule for each host.
Real data delivery.
Requesting the recovery neighbors in PRL one by one upon an error.

Fig. 2. LER system operations.

gaps in the sequence number. Upon an error, the errored host performs lateral error recovery with the recovery neighbors across the planes. We summarize the steps of LER in the Figure 2.

A. Estimation of Hosts Coordinates using GNP

GNP has been proposed in [3] to estimate the relative location of a host in the Internet based on measured network delays, such that the difference between the GNP locations of two hosts correlates well with the actual round trip time between them in the Internet. LER makes use of GNP to estimate host coordinates to contruct Voronoi diagram and hence it will be used to find the closest neighbor.

In GNP, a number of infrastructure hosts, termed as *landmarks*, are used as reference points for measurement purpose. The landmarks, after measuring the round-trip time among themselves, forward the measurement results to one of the landmarks, which computes the landmark coordinates in the GNP space by minimizing the folowing objective function:

$$J_{landmark}(L_1, L_2, \ldots, L_M) = \sum_{L_i, L_j \in \{L_1, \ldots, L_M\} | i > j} \left(\|L_i - L_j\| - RTT(L_i, L_j) \right)^2, \quad (1)$$

where L_i and L_j are the 2D coordinates of landmarks in GNP space to be found (i.e., $L_i = (x, y)$) and $RTT(L_i, L_j)$ is the round-trip time between landmarks L_i and L_j. Clearly, $J_{landmark}$ is the sum of the differences between the measured network distances (i.e., round-trip time) and the logical distances in the GNP space among landmarks. Therefore, we seek to find a set of locations for the landmarks such that the sum is minimized.

Given the landmark coordinates, a new host joining the mesh estimates its location by similarly minimizing another objection function given by:

$$J_{host}(u) = \sum_{L_i \in \{L_1, \ldots, L_M\}} \left(\|u - L_i\| - RTT(u, L_i) \right)^2, \quad (2)$$

where u is the desired host coordinates and $RTT(u, L_i)$ is the measured round-trip time between host u and landmark L_i. Using equation 2, the hosts coordinates can be obtained.

B. Selection of Plane Sources

With the GNP coordinates, we construct a Voronoi diagram for each plane to find the close neighbors to the origin. The Voronoi diagram can be constructed by either a centralized algorithm or distributed algorithm discussed as follows:.

- Centralized algorithm
  Each node sends its GNP coordinates to a specific central server. The central server then constructs the Voronoi diagram for each plane by, for example, Fortune's algorithm [22]. It then informs the origin its closest neighbor in each plane. We show an example in Figure 3. The square indicates the origin and the cirles indicate the set of nodes in the plane 1. From the Voronoi diagram for plane 1, since node k share the same sector with the origin, it is the closest to the origin for plane 1.
- Distributed algorithm
  We construct DT meshs among the hosts. This can be done using Delaunay Triangulation (DT) by incremental construction algorithm discussed in [8]. The DT has a property that closer nodes (in the GNP space) are connected with an edge. The DT mesh constructed has already embedded Voronoi diagram and hence closest node can be identified easily. This can be done by successive probing. The origin probes the node one by one. In each step, a closer node is found until the probing terminates

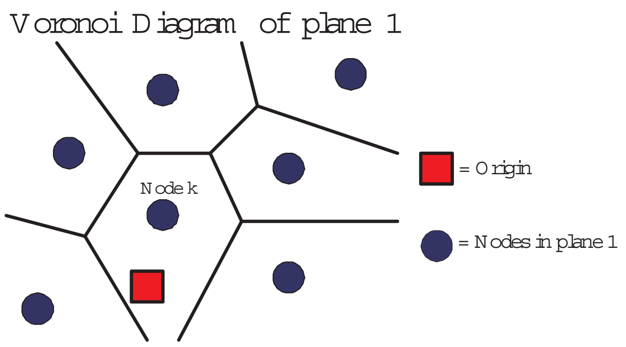


Fig. 3. Finding closest neighbor in Voronoi Diagram.

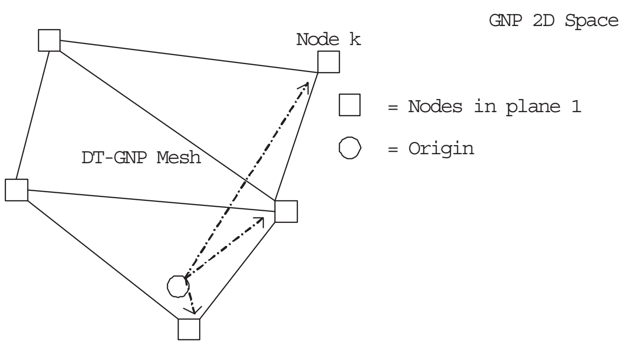


Fig. 4. Finding the closest neighbor by successive probing.

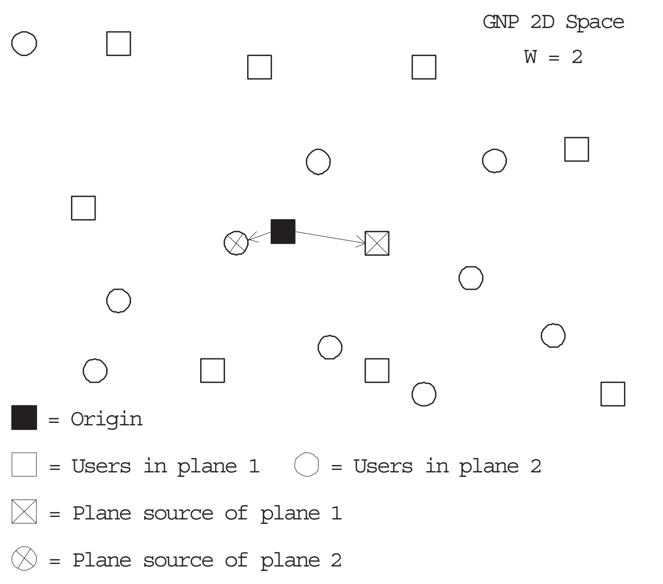
at the closest node. We illustrate the idea in Figure 4, where the origin (indicated by the circle) want to identify the closest node in a plane, say plane 1 (indicated by squares). The nodes are drawn according to their GNP locations. The origin first finds an arbitrary node, say node k, in plane 1. Then node k checks the connected nodes in the DT mesh and reply to the origin another node closer to it. By repeating this successive probing, the origin can then find the closest node in the plane.

Figure 5(a) shows an example with $\omega = 2$ in a GNP 2D space. Users are divided into two planes as indicated by squares and circles. The origin finds the closest node in each plane to be the plane sources which are indicated in the figure by crosses.

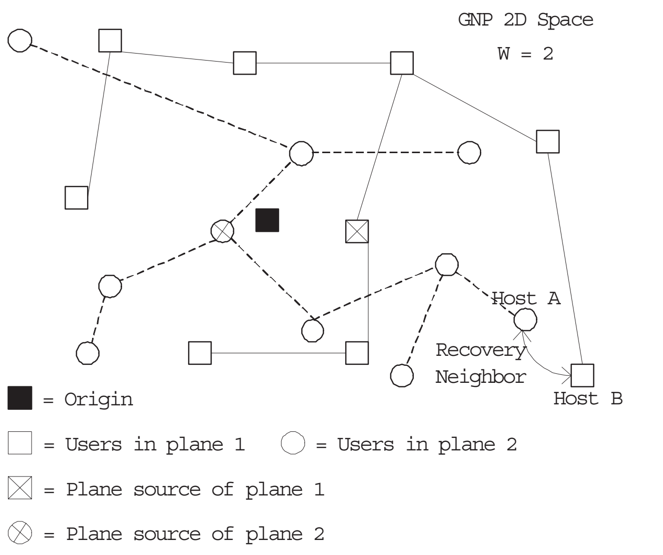
C. Selection of Recovery Neighbors and Retransmission Schedule

In this section, we discuss how to select a node's recovery neighbors and the retransmission schedule (the node sequence for retransmission requests) upon error.

For each node, we need to find the closest neighbor in each of the other planes as the recovery neighbors for lateral recovery. This can be found by constructing the Voronoi diagrams for each plane, as discussed in the previous section. Refer to Figure 5(b) ,hosts A and B are recovery neighbors of each other, since they are belonging to different planes and close to each other.



(a) Example of finding the closest nodes as the plane sources.



(b) Resultant delivery trees.

Fig. 5. Example for LER, assuming number of planes = 2.

A node puts the origin and the recovery neighbors obtained into a list termed the "Potential Recovery-Neighbors List" (PRL). Therefore, the number of nodes in the PRL for node i is at most ω including the origin.

In the following, let's consider an arbitrary node i. Let t_i be the total delay from the origin to node i, and t_j be the total delay from the origin to node j, where $j \in$ PRL of i. Let d_{ij} be the one-way delay between node i and j. We show in Figure 6 the time diagram when there is an error in node i. A packet is transmitted from the origin at time 0. At time t_i, it has not arrived at node i and hence node i detects an error and requests for a retransmission

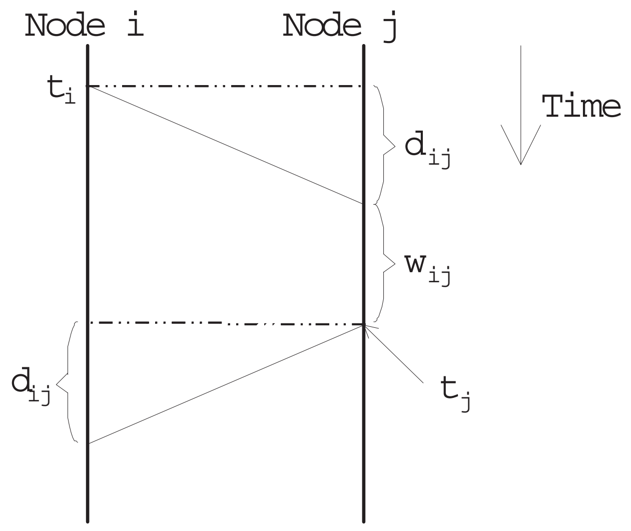
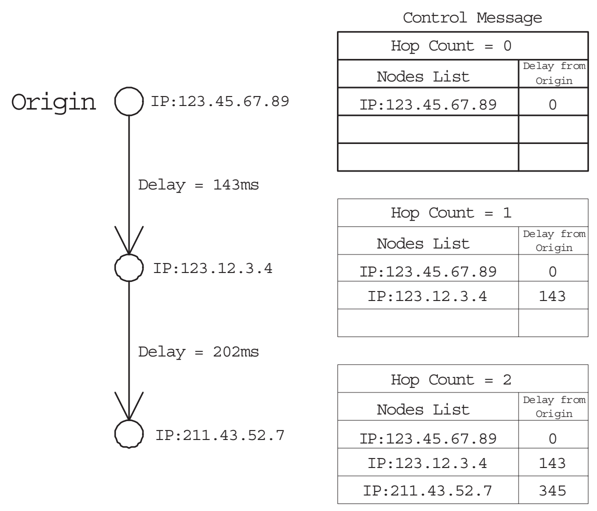


Fig. 6. Ordering of recovery neighbors.



Control Message

Hop Count = 0	
Nodes List	Delay from Origin
IP:123.45.67.89	0

Hop Count = 1	
Nodes List	Delay from Origin
IP:123.45.67.89	0
IP:123.12.3.4	143

Hop Count = 2	
Nodes List	Delay from Origin
IP:123.45.67.89	0
IP:123.12.3.4	143
IP:211.43.52.7	345

Fig. 7. Traveral and updating of control message in a tree.

from node j. The packet arrives at node j at t_j. Let w_{ij} be the minimum waiting time at node j before the requested packet arrives locally. Clearly,

$$w_{ij} = \max(0, t_j - t_i - d_{ij}). \tag{3}$$

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

$$\Gamma_{ij} = 2d_{ij} + w_{ij}. \tag{4}$$

Obviously, requesting a node with low Γ_{ij} leads to faster turnaround time.

We may obtain the delay from the origin to a node as follows. The origin distributes control messages to each plane source, which in turn multicast it in its own plane. Each node, upon the reception of a control message, forms a list of upstream nodes. There is a hop counter-field in the control message. Every time a node receives a control message, it increments the counter by 1, and appends its IP address and delay from the origin onto the message. We show an example in Figure 7, where the origin transmits a control message to two other nodes. By examining the control message, a node can then maintain the list of its upstream nodes[3] and the corresponding delay from the origin.

Note that all the identifying and ordering processes of the recovery neighbors are performed before data delivery (or during tree-reconfiguration). Upon an error, the errored host then follows the recovery schedule without any computation. Such recovery is done as follows:

1) Reliable service: The aim of reliable multicast is to achieve 100% reliability with low recovery delay, defined as the time from the discovery of a lost packet to the arrival time of the packet. To achieve this, node i first orders the nodes in the PRL by their values of Γ_{ij} in ascending order. Upon an error, node i requests the recovery neighbor according to the order.

Note that, in general, the closest neighbors of a node are chosen first before requesting the origin. Since most of the losses can be recovered by the lateral nodes, the implosion problem in the origin is relieved.

2) Streaming Applications: For media steaming, there is a certain recovery deadline δ (in seconds). If a lost packet cannot be recovered within δ, it is as good as lost. We hence are concerned on the residual loss rate, the fraction of lost packets after recovery.

Because of the recovery deadline, some of the nodes in PRL may receive packets too late to be useful and hence have to be screened out. Clearly, node j is retained in the PRL of node i if it satisfies the following two conditions:

$$t_j + d_{ij} \leq t_i + \delta, \tag{5}$$

and

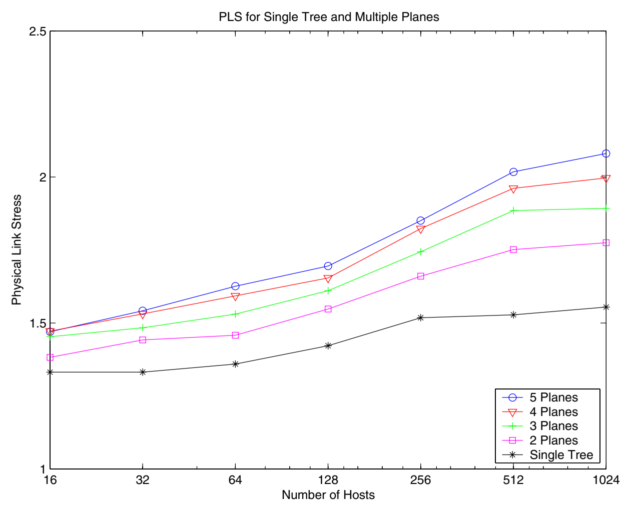
$$d_{ij} \leq \frac{\delta}{2}. \tag{6}$$

Failure to meet these two conditions means that the node is not possible to retransmit the packet in time. The node is then rejected from the PRL. After this step, node i then orders the nodes and makes retransmission according to the order.

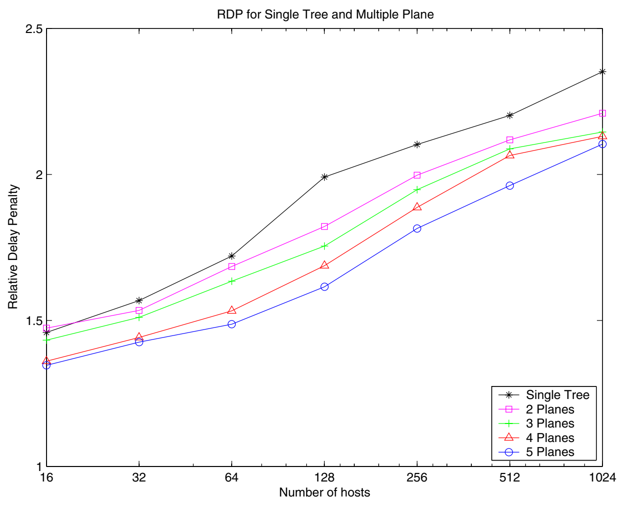
IV. Illustrative simulation results

We present the simulation results of LER based on Internet-like topologies in this section. We generated a

[3]The list of the upstream nodes is reserved for future use.



(a) PLS of LER for different number of planes.



(b) RDP of LER for different number of planes.

Fig. 8. ALM performance of LER.

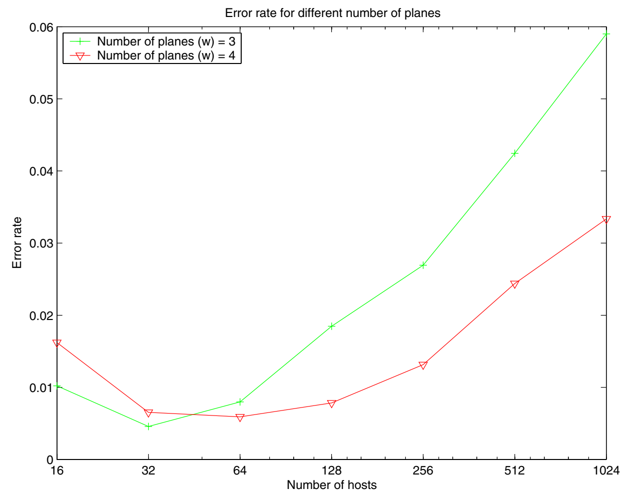


Fig. 9. Residual loss rate for different number of planes.

number of *Transit Stub* topologies with the Geogrgia Tech random graph generator [23]. The hosts are randomly distributed in the network, with a 1 ms delay from itself to the router attached. We use DT as our mesh construction and compass routing to multicast data [8]. Packets are randomly dropped in a link of the network, with probability 0.95 the loss rate is uniformly distributed between 0 and 1%, and with probability 0.05 the loss rate of a link is uniformly distributed between 5% to 10% (as according to a study based on real measurement [24]).

In the simulations, we are interested in the following metrics:

- Physical link stress (PLS), the number of identical copies of a packet that traverse a physical link.
- Relative delay penalty (RDP), the ratio of the delay in the overlay with the delay in the shortest-delay unicast path.
- Recovery delay (for reliable service), the delay from the time of discovering error to the time of recovering the packet.
- Residual loss rate (for streaming applications), the overall loss rate of a host within δ after retransmission.

In Figure 8(a) and 8(b), we show PLS and RDP versus the number of hosts, respecting, given different number of plane trees. From Figure 8(a), we see that as the number of hosts increases, PLS in general increases. Given a certain number of hosts, the more the planes are, the higher the PLS is. This is reasonable as multiple tree is less efficient than single tree. However, the cost is minor. We see that LER only introduces 5 - 20% overheads. For relative delay penalty (RDP) (Figure 8(b)), it increases with the number of hosts as the trees are deeper. Given a certain number of users in the system, higher number of planes reduces the RDP. Since with a larger number of planes, there are more delivery trees. This causes reduction on the tree depth, and thereof RDP, is reduced.

Now we analyze the effect of ω on the performance of LER. We consider the residual loss rate in streaming applications. We set the playout deadline δ to be 2 seconds. So that recovery performed beyond the 2 seconds will be regarded as loss. In Figure 9, we show the residual loss rate versus the number of hosts, respecting, given ω is 3 and 4. Regarding the residual loss rate which first decreases and then increases, because when the number of hosts is small, they tend to be quite far apart. Therefore, the recovery neighbors are also far away, which makes inefficient recovery. On the other hand, when the number of hosts is high, the trees are deeper and hence the error rate over all the hosts generally increases. As more retransmission requests are made and more recovery neighbors have

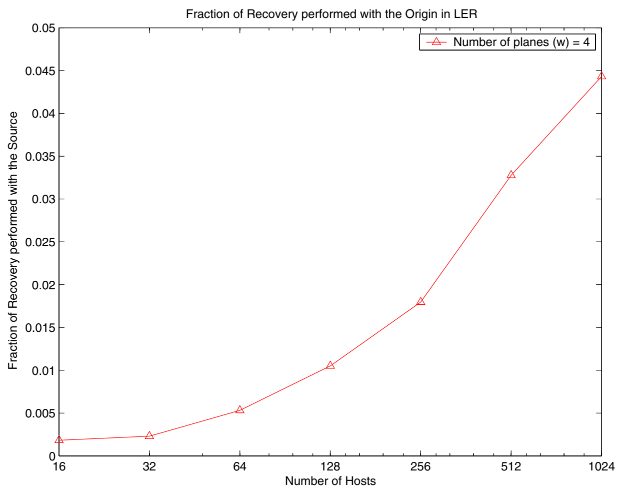


Fig. 10. Fraction of recovery performed with the origin.

to be visited, the time for recovery increases, thus the loss rate increases.

Figure 9 shows that when the number of hosts increases, more planes are needed because they give more recovery neighbors for retransmission. This reduces the chance to request the origin, which is in general costly. However, for a given number of users (e.g. 16 and 32), there is diminishing value in increasing the plane number from 3 to 4. The general rule is, we need more planes when the expected number of hosts is large.

As mentioned, origin is costly, however it is necessary to provide 100% reliability. Besides, large fraction of origin recovery causes implosion problem, thus it is important to keep the fraction low. In Figure 10, we show the fraction of recovery performed with the origin versus the number of hosts, given $\omega = 4$. We can observe that the fraction is kept below 5% throughout the experiment. For the baseline system with 256 hosts, the fraction is below 2%. Thus, only small portion of error recovery is performed with the origin and does not cause the implosion problem.

Next, we compare the following different schemes with LER:

- Source Recovery
  All nodes ask the origin for retransmission whenever there is an error. A strength of this scheme is that the origin is certain to have the packet in request. However, a serios problem of this scheme is source implosion: any node losing a packet would request retransmission at the origin. Also the average recovery delay between the origin and plane hosts is relatively high
- Parent Recovery
  Error node requests retransmission from its parent. Whenever the error packet is recovered, the packet is further delivery to the downstream. The idea is simple and effective, however it cannot deal with the problem of node failure. Suppose an interior node leaves the multicast group, all the downstream nodes will temporary outage of service.
- Probabilistic Resilient Multicast (PRM)
  PRM is recently proposed in [4]. The scheme employs two mechanisms to reduce the residual loss rate in streaming applications, namely *randomized forwarding* and *triggered NAKs*. For randomized forwarding, each node introduces a certain number r of redundant link and with probability β to forward a packet in each of the redundant link. For triggered NAKs, error host performs retransmission with the parent (the same idea of parent recovery). PRM gives better solution to the problem of node failure compared with parent recovery, however it does not concern the 100% reliability for reliable services. Since PRM duplicates packets in the random forwarding, it causes extra load to the network.
- Lateral Error Recovery (LER)
  LER is our proposed recovery scheme. In the following, we consider a baseline system with $\omega = 4$ and for distribution of recovery delay we consider the number of hosts to be 256.

TABLE I
AVAILABILITY OF SERVICES FOR DIFFERNT RECOVERY SCHEMES.

	Reliable	Streaming
No Retransmission	$\times$	$\surd$
Source Recovery	$\surd$	$\surd$
Parent Recovery	$\times$	$\surd$
PRM	$\times$	$\surd$
LER	$\surd$	$\surd$

We consider two sorts of services with different recovery schemes. However not both services can be provided for all the recovery schemes. We show the availability in the Table I as a conclusion.

In the following, we consider node failure. Each host has a certain failure probability (set to be 5%). All these error hosts would failure by a certain time, with the failure time uniformly and independently distributed within a period (640 seconds, our simulation time). For simplicity, there is no tree-reconfiguration upon a failure of a node.

We first analyze the performance of providing reliable service. The origin distributes content to the hosts with perfect reliability, the recovery delay is considered. Recovery delay reflects the efficient of the recovery schemes. Since parent recovery and PRM cannot provide reliable service in the condition of node failure, their recovery de-

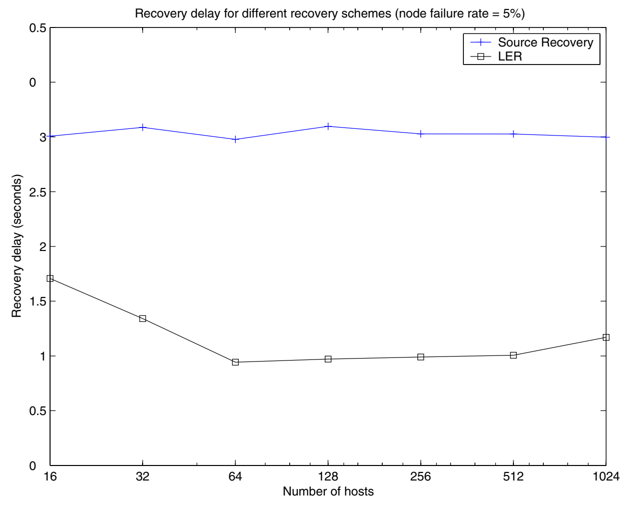
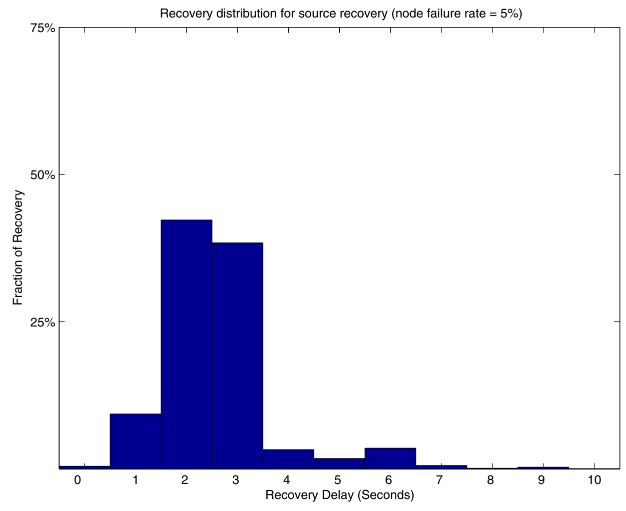
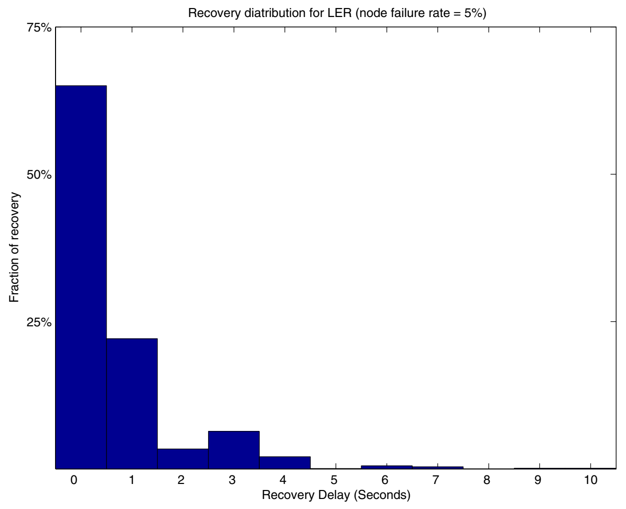


Fig. 11. Recovery delay for different recovery schemes



(a) Recovery distribution of source recovery.



(b) Recovery distribution of LER.

Fig. 12. Recovery distribution for schemes of reliable service.

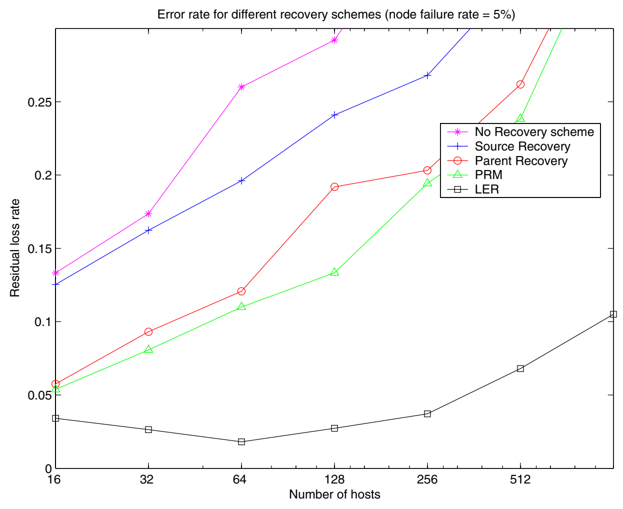


Fig. 13. Residual loss rate for different recovery schemes

lays are not considered. We show in Figure 11 the average recovery delay versus the number of hosts. The average recovery delay of the source recovery is rather independent of the number of hosts. This is because all the nodes ask the origin for retransmission. For a given network topology and random distribution of nodes, their average distance to the origin remains more or less the same. We can observe that our LER gives better performance compare with source recovery.

We compare the distribution of recovery delay of the lost packets for lateral recovery and source recovery in Figure 12. The number of hosts is 256. Obviously, the distribution of lateral recovery is skewed towards the left, leading to its low recovery delay. Most of the recovery can be done within a short time (one second). For source recovery, on the other hand, most of the recovery is done with a longer time. However, with lateral recovery, a small fraction of losses has high delay (a longer tail), due to the fact that some retransmission requests have to go to the plane sources and even the origin for recovery. However, the tail is overall not too serious.

Next we analyze the performance to provide streaming applications. In streaming applications, hosts are fault tolerant and the packets become useless until some playout deadline δ is reached. We analyze the residual loss rate for the hosts and δ is set to be 2 seconds. We compare the schemes in the Figure 13, we show the residual loss rate versus the number of hosts. The loss rate for the bare scheme (without any retransmission) is also shown. Generally the loss rate increases with the number of hosts increases. We can observe that parent recovery performs better than source recovery. This is because the distance between the successive nodes in the delivery tree is much smaller than the distance between a host and the origin.

PRM employs the same recovery idea as the parent recovery. In addition, PRM added some extra links and duplicate packets to enhance the performance, as a result it is slightly better than the original parent recovery. LER maintains the loss rate at a relatively low level and thus increases the quality of service. We can observe that under the condition of node failure, our LER approach is relatively less affected. The figures show that our system significantly outperforms the vertical recovery schemes and PRM.

V. Conclusion

In this paper, we propose and study *lateral error recovery* (LER) to recover packet loss for application-level multicast (ALM). We have considered reliable service (where packet losses have to be completely recovered) and streaming service (where there is a time constraint to recover lost packets).

In lateral error recovery, nodes are divided into independent planes. Each plane forms an ALM tree. A node recovers its losses by retransmission from some recovery neighbors close to itself in the other planes. Since data is sent along the plane trees independently, recovery neighbors have a much lower error correlation with the errored node as compared with the traditional vertical recovery.

We have described the operation of our system and compared our scheme with a previously proposed scheme Probabilistic Resilient Multicast (PRM) and two vertical recovery schemes, namely source recovery and parent recovery. We show that the average recovery delay with LER for reliable service can be greatly reduced. For streaming applications, in contrast to PRM and vertical recovery schemes, the residual loss rate of our scheme remains low even when the number of hosts increases. Also we have shown that the performance of LER is relatively robust to the node/link failure. Lateral error recovery is simple and effective, and only incurs low overheads in terms of physical link stress.

References

[1] Victor O. K. LI and Zaichen Zhang, "Internet multicast routing and transport control protocols," *Proceedings of the IEEE*, vol. 90, no. 3, pp. 360–391, March 2002.

[2] Lakshminarayanan Subramanian, Ion Stoica, Hari Balakrishnan, and Randy H. Katz, "OverQoS: Offering QoS using overlays," in *1st HotNets Workshop*, October 2002.

[3] T. S. Eugene Ng and Hui Zhang, "Predicting internet network distance with coordinates-based approaches," in *Proceeding of INFOCOM 2002*, New York, June 2002.

[4] Suman Banerjee, Seungjoon Lee, Bobby Bhattacharjee, and Aravind Srinivasan, "Resilient multicast using overlays," in *ACM SIGMETRICS 2003, the International Conference on Measurement and Modeling of Computer Systems*, San Diego, California, June 2003.

[5] Suman Banerjee, Bobby Bhattacharjee, and Christopher Kommareddy, "Scalable application layer multicast," in *Proceeding of SIGCOMM 2002*, Pittsburgh, August 2002.

[6] Dimitrios Pendarakis, Sherlia Shi, Dinesh Verma, and Marcel Waldvogel, "ALMI: An application level multicast infrastructure," in *Proceeding of 3rd USENIX Symposium on Internet Technology and Systems*, San Francisco, March 2001, USENIX.

[7] Paul Francis, Pavlin Radoslavo, Robert Lindell, and Ramesh Govindan, "Your own internet distribution YOID," http://www.isi.edu/div7/yoid/.

[8] Jorg Liebeherr, Michael Nahas, and Weiheng Si, "Application-layer multicasting with delaunay triangulation overlays," *IEEE Journal on Selected Areas in Communicastions*, vol. 20, no. 8, pp. 1472–1488, October 2002.

[9] Yang hua Chu and Sanjay G. Rao Srinivasan Seshanand Hui Zhang, "A case for end system multicast," *IEEE Journal on Selected Areas in Communicastions*, vol. 20, no. 8, pp. 1456–1471, October 2002.

[10] Miguel Castro, Peter Druschel, Anne-Marie Kermarec, and Antony I. T. Rowstron, "Scribe: A large-scale and decentralized application-level multicast infrastructure," *IEEE Journal on Selected Areas in Communicastions*, vol. 20, no. 8, pp. 1489–1499, October 2002.

[11] K.-F. Wong, S.-H. Chan, and W.-C. Wong, "Lateral error recovery for application-level multicast," *ACM Sigcomm 2003 (Poster session)*, 25-29 August 2003.

[12] Matthias Grossglauser, "Optimal deterministic timeouts for reliable scalable multicast," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 3, pp. 422–433, April 1997.

[13] Don Towsley, Jim Kurose, and Sridhar Pingali, "A comparison of sender-initiated and receiver-initiated reliable multicast protocols," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 3, pp. 398–406, April 1997.

[14] Jörg Nonnenmacher, Ernst W. Biersack, and Don Towsley, "Parity-based loss recovery for reliable multicast transmission," *IEEE/ACM Transactions on Networking*, vol. 6, no. 4, pp. 349–361, August 1998.

[15] Jörg Nonnenmacher and Ernst W. Biersack, "Scalable feedback for large group," *IEEE/ACM Transactions on Networking*, vol. 7, no. 3, pp. 375–386, June 1999.

[16] Ching-Gung Liu, Deborah Estrin, Scott Shenker, and Lixia Zhang, "Local error recovery in SRM: Comparison of two approaches," *IEEE/ACM Transactions on Networking*, vol. 6, no. 6, pp. 686–699, December 1998.

[17] Martin S. Lacher, Jörg Nonnenmacher, and Ernst W. Biersack, "Performance comparison of centralized versus distributed error recovery for reliable multicast," *IEEE/ACM Transcations on Networking*, vol. 8, no. 2, pp. 224–238, April 2000.

[18] María Calderón, Marifeli Sedano, Arturo Azcorra, and Cristian Alonso, "Active network support for multicast applications," *IEEE Network*, pp. 46–52, May/June 1998.

[19] Sneha K. Kasera, Supratik Bhattacharyya, Mark Keaton, Kiane Kiwior, Steve Zabele, Jim Kurose, and Don Towsley, "Scalable fair reliable multicast using active services," *IEEE Network*, pp. 48–57, January/February 2000.

[20] Brian Whetten and Gursel Taskale, "An overview of Reliable Multicast Transport Protocol II," *IEEE Network*, pp. 37–47, January/February 2000.

[21] Eunsook Kim, Shin Gak Kang, and Jongwon Choe, "A router-assisted session tree configuration mechanism for reliable multicast," *IEEE Communications Letters*, vol. 6, no. 9, pp. 464–466, September 2002.

[22] Steven Fortune, "A sweepline algorithm for voronoi diagrams.," *Algorithmica*, vol. 2, pp. 153–174, 1987.

[23] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proceedings of INFOCOM 1996*, San Francisco, CA, 1996.

[24] Venkata N. Padmanabhan and Lili Qiu, "Netwrok tomography using passive end-to-end measurements," *DIMACS Workshop on Internet and WWW Measurement, Mapping and Modeling*, February 2002.