Scaling Multicast Transports Through End-to-end Topology Discovery

by

Sylvia Ratnasamy

B.E. (University of Poona, India) 1997

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Master of Science

in

Computer Science

in the

GRADUATE DIVISION
of the
UNIVERSITY of CALIFORNIA at BERKELEY

Committee in charge:

Professor Steven McCanne, Chair
Professor Anthony D. Joseph

1999

The dissertation of Sylvia Ratnasamy is approved:

Chair Date

Date

University of California at Berkeley

1999

Scaling Multicast Transports Through End-to-end Topology Discovery

Abstract

Scaling Multicast Transports Through End-to-end Topology Discovery

by

Sylvia Ratnasamy

Master of Science in Computer Science

University of California at Berkeley

Professor Steven McCanne, Chair

The IP service model retains its simplicity and robustness by deferring reliability and congestion control to higher layers through end-to-end transport protocols. While the IP unicast service has proven successful, extending end-to-end adaptation to multicast has been a difficult problem. Unlike the unicast case, multicast protocols must support large and heterogeneous receiver sets. While proposed approaches to multicast transports attempt to localize problems and/or organize receivers into a hierarchy through a divide-and-conquer approach, this approach succeeds only if the resulting hierarchy is congruent with the underlying routing tree topology. This implies the need for some level of topological information at the end systems which the IP multicast service unfortunately deliberately hides.

In this report, we explore the problem of inferring the required topological information using only end-to-end observations made at the end hosts. We present a centralized algorithm that infers the internal structure of a multicast distribution tree using a relatively simple model based on shared loss patterns and probabilistic analysis that eliminates "false sharing".

We then present a distributed Group Formation Protocol whereby receivers dynamically organize themselves into a multi-level hierarchy of multicast groups that corre-

sponds to the underlying routing tree. The Group Formation Protocol can serve as a core component across a wide range of multicast applications and protocols such as local recovery for reliable multicast, self organized transcoding, self organizing web caches, the optimal and dynamic placement of proxies, repeaters, designated receivers, recorders and so forth.

Contents

List of Figures **iv**
1 Introduction . . . 1
2 Inferring Structure from Shared Loss . . . 6
3 Tree Inference Algorithm . . . 9
3.1 Binary trees . . . 10
3.2 Arbitrary tree topologies . . . 11
3.3 Evaluation . . . 13
3.4 Tree inference algorithm as a network measurements tool . . . 16
4 A Group Formation Protocol . . . 19
4.1 Terminology . . . 19
4.2 Group Formation . . . 20
4.3 Recovery from incorrect grouping decisions . . . 25
4.4 Adapting to changes in link loss rates and routing topologies . . . 25
5 Evaluation . . . 26
5.1 Accuracy of GFP . . . 26
5.2 Effect of protocol parameters . . . 28
6 GFP as a reusable component . . . 31
7 Related Work . . . 34
8 Conclusions and Future Work . . . 37

Bibliography **39**

List of Figures

1 IP Multicast hides the underlying topology 3
2 Topological Optimizations 4
3 Shared Loss Pathology 7
4 Loss Model 7
5 Original tree topology 10
6 Inference of the logical tree 10
7 Possible relationships between a pair of nodes to be coalesced for arbitrary trees 12
8 Implementation modules 13
9 Binary trees: individual link loss rate is selected at random in the range [0%,10%] 15
10 Binary trees: individual link loss rate is selected at random in the range [0%,5%] 15
11 Tree inference algorithm for binary trees with 24 nodes and link loss rates uniformly selected in the range [0-10]% 16
12 Tree inference algorithm for binary trees with 24 nodes and link loss rates uniformly selected in the range [0-20]% 16
13 Packet pairs flowing through a bottleneck link. The vertical dimension is bandwidth, horizontal dimension is time 17
14 The bottleneck bandwidth seen by each interior node equals at least the maximum of the estimate of its downstream nodes 18
15 Evolution of a multi-level hierarchy 21
16 GFP performance for different distributions of link loss rates 28
17 GFP performance for different size topologies 28
18 Effect of LR_{min} on number of groups in the final hierarchy 29
19 Effect of $Thresh$ on the shape of the final hierarchy 29
20 $Thresh$ affects the shape of the hierarchy 29

Acknowledgements

I would like to thank my advisor Prof. McCanne for suggesting this topic, discussing all the ideas herein and for on the whole, being a great advisor.

This work greatly benefitted from a number of discussions with Sally Floyd, Vern Paxson, Steve Deering, Hari Balakrishnan, Matt Podolsky, Nick Duffield and Francesco LoPresti. I'd also like to thank Yatin Chawathe for patiently answering my never ending UNIX-related questions.

Finally, I'd like to thank Kapil Shrikhande for sitting through what must have been very boring monologues on the "importance" of topology inference and for the innumerable morning cups of coffee without which much of this work would never have been completed.

1 Introduction

The core foundation for the great technical success of the Internet is quite arguably the *end-to-end design principle* [36], which says that when layering a systems design one should assign a given function to the lowest layer in which it can be wholly and correctly implemented. When applied to internetworking of heterogeneous components and end-systems, this principle naturally leads to one of the most critical Internet design choices, namely its *best effort* network-layer delivery semantics. In this framework, end-to-end reliability cannot be wholly and correctly implemented at the network layer so network components are deliberately allowed to drop, randomly delay, replicate, or reorder packets, and richer services like reliable, sequenced delivery must be implemented on top of this unreliable delivery model. But because the requirements imposed on the network are so minimal, a seemingly very difficult problem (i.e., interconnecting millions of heterogeneous components across a world-wide internetwork) becomes much simpler: the federation of packet routers spread across the network need only make a best effort attempt at delivering packets toward their ultimate destination.

This end-to-end approach to network design cuts across the Internet architecture and has proven enormously successful in the decomposition of the TCP/IP protocol suite. TCP not only masks the unreliability of the underlying network but it carries out end-to-end congestion control to protect the network as a whole from uncontrolled overload [14], all in spite of the inherent unreliability of the underlying IP service model. But this is far from easy. The mechanisms for achieving end-to-end reliability, establishing and releasing connections, guaranteeing data integrity, and so forth are all rather complex. Moreover, properly realizing the dynamics of the feedback control system that forms the foundation of the TCP congestion control algorithm has likewise proven quite difficult [30]. Yet, despite this complexity, the TCP/IP division of labor between a *reliable* closed-loop transport service and an *unreliable* open-loop network service has enabled the incremental engineering and deployment of very large-scale networks and ultimately led to the success of the TCP-based global Internet [7].

Quite naturally, then, when Deering proposed IP Multicast [9], an extension to the Internet architecture to support efficient multi-point packet delivery, he appealed quite deliberately to this very same end-to-end design methodology. The IP Multicast service provides for efficient one-to-many packet transmission. A single packet transmitted by the source is delivered to an arbitrary number of receivers by replicating the packet within the network at fan-out points along a distribution tree rooted at the traffic's source [23]. Like unicast IP, the IP Multicast service provides only best effort packet delivery and richer multicast services must be built on top of this unreliable delivery model. But unlike unicast, in which a single source and receiver intercommunicate via a single (possibly dynamic) path through the network, in a multicast *session* with N hosts, there are roughly N^2 paths, one between every source host and every other recipient host.

This multiplicity of paths imposes great difficulty on the design of end-to-end multicast transports. The hallmark characteristic of end-to-end transport protocols like TCP is their ability to dynamically adapt to the network path and thereby match the load imposed by the end-systems to the resources available within the network. But "resource availability" is ill-defined in multicast because there is not single path and no single measure of available resource, and in fact, the different paths to diverse receivers can have widely divergent characteristics. Thus, a multicast sender is at a loss as to how to adjust its sending rate to match the diverse requirements of a heterogeneous set of network paths and end-systems.

In addition to the challenges imposed by *heterogeneity*, multicast communication also induces challenging problems that center upon *scale*. In a large multicast session, there are many and diverse sources of data and control and a naive protocol design can easily generate uncontrolled traffic transients and effects like the well-known *feedback implosion* problem, where a large number of receivers simultaneously generate feedback messages addressed to a single source. Fortunately, many of these problems of scale and heterogeneity are fairly well understood and a great deal of existing research addresses them [39, 25, 26,

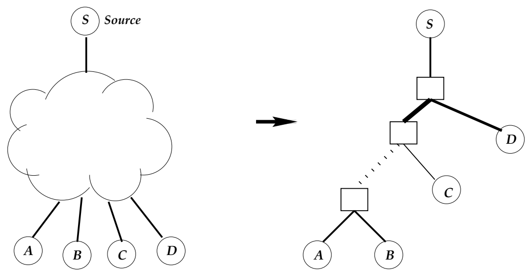

Figure 1: IP Multicast hides the underlying topology

11, 40, 44, 21, 17, 22]. However, despite all this research, end-to-end transport protocols for multicast that gracefully accommodate heterogeneity, perform well at very large scale, and include proven, Internet-compatible congestion control strategies still remain quite far from our grasp.

Given that no clear-cut, viable solution has emerged from all this work on multicast transports, we believe it is critical to better understand the fundamental difficulties that are carried within this problem space. A common trait of many proposed end-to-end multicast solutions is some scheme for exploiting the structure and topology of the underlying multicast distribution tree (e.g., Liu's work on local recovery [21] attempts to cluster receivers in common sub-trees while RLM [26] induces receivers that share common paths to subscribe to equivalent rate signals). Essentially, these prior works attempt to tackle both heterogeneity and scale simultaneously with a *divide and conquer* approach, e.g., by creating multiple loss recovery groups arranged in a hierarchy to localize the impact of control traffic. In this way, one large, heterogeneous, and difficult problem is reduced to many small, homogeneous, and simpler sub-problems. The goal of our work is to assess the raw difficulty of hierarchically decomposing a multicast distribution tree in this fashion in order to determine the viability of end-to-end transports built on this principle.

Given the simplicity of the IP Multicast service model, how could we realistically perform this hierarchical decomposition? As illustrated in Figure 1 , we can think of the multicast service as a simple cloud-based model, where a source injects a packet into the

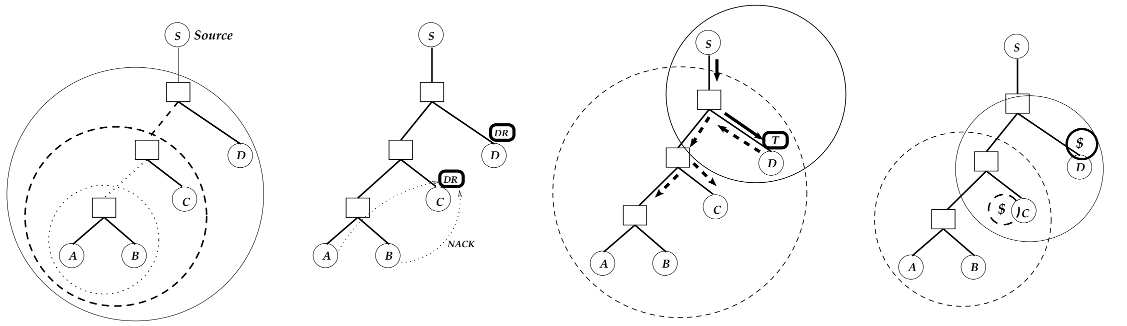

Figure 2: Topological Optimizations

cloud and each receiver receives a copy of that packet. In other words, the service model deliberately hides the underlying topology from the end-systems, which means that we have no easy mechanism for building an efficient hierarchy. If we could discover the underlying topology, as illustrated in the right-hand side of Figure 1, our problem would become almost trivial. That is, if we knew this topological structure, we could carry out a variety of topologically-sensitive protocol optimizations. For example, Figure 2 shows how we could form local recovery groups for reliable multicast; place "designated receivers" (DRs) in RMTP [21]; optimally position video transcoders in SOT [17]; or optimally overlap multicast-based Web caches [15, 27]. These optimizations are discussed in greater detail in Section 6.

But IP deliberately hides topology and we have no easy mechanism to discover it, which begs the question: should we extend the IP service model, or in particular the IP multicast service model, with mechanisms for explicitly acting upon or discovering the network's underlying topological structure ? Though we believe this question will remain open for some time, in this report we take the end-to-end position and argue that much of the structure of an underlying multicast distribution tree can be inferred through observations made only at end-systems. Toward this end, we investigate the question of how well one can infer topological information about a multicast distribution tree using strictly end-to-end observations, and show that with a relatively simple model based on shared loss patterns and probabilistic analysis that eliminates "false sharing" signals, it is possible to infer the entire routing tree topology [34].

Though our tree inference algorithm performs well in theory, it assumes a global exchange of information among all the participants in a multicast session, wherein an omniscient, centralized agent has complete knowledge of each receiver's loss patterns from the beginning of time. Though unrealistic in practice, we initially adopt this hypothesis quite conscientiously, in order to explore the fundamental limitations of an inference-based approach. Given the success of this initial investigation, the next challenge, which we tackle herein, is how we might leverage this ability to infer the multicast tree topology in a practical protocol framework.

Toward this end, we propose herein a core protocol building block that can be used across a wide variety of end-to-end multicast transports to ameliorate the performance hurdles imposed by heterogeneity and scale. We describe, simulate, and evaluate a distributed Group Formation Protocol (GFP) that leverages our earlier work on multicast tree inference to produce a topologically-sensitive protocol primitive.

In the next section, we relate our work to the current state of the art in self-organizing, multicast group-formation protocols, by illustrating where these approaches fall short and how we overcome these shortcomings with our probabilistic model. Section 3 describes our tree inference algorithm used to determine the underlying multicast tree structure by noting correlations of loss patterns across the receiver set. We then describe how GFP works and present the details of the protocol framework and sketch how GFP can be reused across and integrated into a variety of existing solutions for end-to-end multicast transport. Finally, we discuss our plans for future work and conclude.

2 Inferring Structure from Shared Loss

State of the art end-to-end multicast transports exploit the structure and topology of the underlying multicast distribution tree in an attempt to tackle both heterogeneity and scale. The ability to infer this structure is made possible by the multicast tree structured delivery model. Multicast packets flow along a distribution tree rooted at the source. The receivers form the leaves of the tree, the routers are the internal nodes in the tree and the links form the edges of the tree. A packet that is dropped along any link of the distribution tree, is lost by all the downstream receivers in the subtree rooted at that link. The tree structured delivery model thus introduces correlations in the packet losses seen by the different receivers. This loss correlation between receivers can be exploited in order to infer the tree structure that caused the observed loss patterns. For example, landmark works by Liu [22] and Kouvelas [17] leverage the fact that the multicast tree structured delivery model induces correlations in the end receiver packet losses attempting to cluster together co-located receivers through "shared loss thresholding". In these schemes, every receiver has an associated *loss fingerprint* or *lossprint* which is simply a list of sequence numbers of packets lost by that receiver. Receivers exchange lossprints over the multicast channel and monitor their local loss rate. If a receiver experiences a sufficiently high rate of loss, it initiates the formation a new multicast group. Other receivers join the new group if their lossprints share enough similarity with the group initiator's lossprint according to a thresholding rule.

Clustering receivers into groups through shared loss thresholding is thus built on the base assumption that shared loss implies spatial (topological) correlation. However, this assumption can fail in a subtle and misleading way. This is illustrated in Fig 3 where a source transmits to three receivers A, B and C. Receivers A, B and C share link L_1 and only A and B share L_2. If links L_3 and L_4 have high loss rates and L_5 has a relatively low loss rate then it is entirely possible that the probability of seeing shared losses between receivers A and C exceeds that between A and B which would lead us to falsely conclude

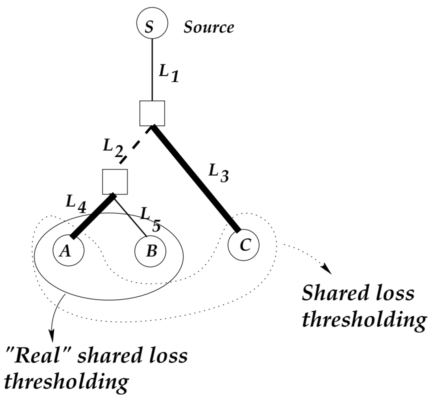

Figure 3: Shared Loss Pathology

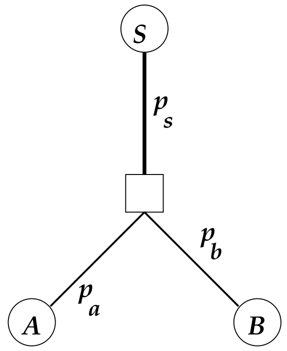

Figure 4: Loss Model

that A and C are more closely located than A and B. This *shared loss pathology* occurs because high loss rate links can cause physically disparate receivers to have higher shared loss than physically co-located receivers. Given this shared loss pathology, is it possible to somehow extract the true loss correlations between receivers?

Examining the different ways in which shared loss occurs reveals the flaw underlying shared loss thresholding. Shared losses arise in two ways: either due to a shared path or due to mere coincidence. To clarify, consider a pair of receivers A and B (Fig 4) that share the path from the source to their closest common ancestor. Packets lost along this shared path are truly indicative of the underlying tree structure. We call these *real* shared losses. In addition to real shared losses, distinct copies of the same packet can be lost along the independent paths from the closest common ancestor to each receiver. These coincidental shared losses are *not* caused by the underlying tree structure. We call these *false* shared losses. The failure modes that arise in the use of the net shared losses as metric for spatial locality are due to this "false sharing". *Real* shared losses, on the other hand, are a robust metric of spatial locality because the greater the extent of the shared path between a pair of receivers, i.e. the more closely located a pair of receivers, the greater will be the real shared losses between them.

At the end host however, a real shared loss is indistinguishable from a false one, given no additional information. However we can overcome this problem by utilizing lossprints and the simple loss model of Fig 4 to separate out the real shared loss from the total shared loss observed between a pair of receivers. Packet losses are modeled as being independent. The p's next to each link in Fig. 4 represent the individual link loss probabilities. In particular, p_s represents the probability of seeing real shared loss between receivers A and B with associated lossprints L_a and L_b.

The above model yields the following equations:

$$\begin{aligned} P_{ab} &= p_s + (1-p_s)p_a p_b \\ P_{a\bar{b}} &= (1-p_s)p_a(1-p_b) \\ P_{b\bar{a}} &= (1-p_s)(1-p_a)p_b \end{aligned}$$

where, P_{ab} is the probability of a shared loss (whether real or false) between A and B and $P_{a\bar{b}}$ and $P_{b\bar{a}}$ are the probabilities of a loss at one receiver but not the other. Let the number of measured shared losses between A and B be $|L_{ab}|$ where $L_{ab} = L_a \cap L_b$. We approximate P_{ab} as $|L_{ab}|/n$ where n is the number of packets transmitted at the source. Let the number of measured losses seen by A but not by B be $|L_{a\bar{b}}|$. We approximate $P_{a\bar{b}}$ as $|L_{a\bar{b}}|/n$. and similarly $P_{b\bar{a}}$ as $|L_{b\bar{a}}|/n$. Solving the above equations we derive the real shared loss probability as:

$$p_s = \frac{P_{ab}P_{b\bar{a}} + P_{b\bar{a}}P_{a\bar{b}} + P_{a\bar{b}}P_{ab} + P_{ab}^2 - P_{ab}}{P_{ab} + P_{b\bar{a}} + P_{a\bar{b}} - 1}$$

In Section 3.3, we present simulation results comparing the efficiency of inferring topological structure using real shared loss probabilities versus the use of shared loss probabilities. Our results prove that shared losses probabilities are not a good metric for topological proximity thus clearly indicating the need for the use of real shared loss probabilities as a metric for topologically sensitive protocol primitives.

Given that we can surmount the shared loss pathology, we now turn to the problem of inferring the topological structure of the underlying multicast routes through our knowledge of real shared loss probabilities.

3 Tree Inference Algorithm

This section describes our tree inference algorithm which reconstructs a logical representation of the multicast routing tree topology. The tree inference algorithm assumes global knowledge of receiver lossprints since the beginning of time and thus is not a practical protocol building block in its own right. However, we believe the process of exploring this extreme point sheds light on the difficulty of the problem and forms the foundation for follow-on work that exploits variations in the basic approach to trade off computational overhead for topological accuracy.

The algorithm reconstructs a 'logical' representation of the multicast tree. A logical representation of a multicast tree is one in which each interior node is merely the closest common ancestor of all downstream receivers in the tree [42]. In reality each branch of the logical tree could consist of a series of links. In order to learn the exact topology of the tree we would have to enlist the help of each intermediate router along the path as is done in the traceroute and mtrace tools. Our algorithm is based only on end-to-end measurements using only information that is readily available at the end hosts and requires no special router support, as such, reconstructing a logical tree is as accurate as we can get. Knowledge of the logical tree is however sufficient for our purpose because all the receivers downstream of a given logical branch will see the same path characteristics such as the bottleneck bandwidth and loss rate irrespective of which component link of the logical branch caused the observed characteristics.

The tree inference algorithm reconstructs the routing tree in a bottom up fashion by successively aggregating together receivers that share the highest *real* shared loss probability. Receivers having similar loss patterns are aggregated together and represented by a single node one level higher in the tree.The aggregated nodes can then be regarded as a single node for further aggregation. The entire tree has been reconstructed when all the receivers have been coalesced in this manner into a single tree. For example: In order to rebuild the tree shown in figure 5, the algorithm initially begins with a set of individual

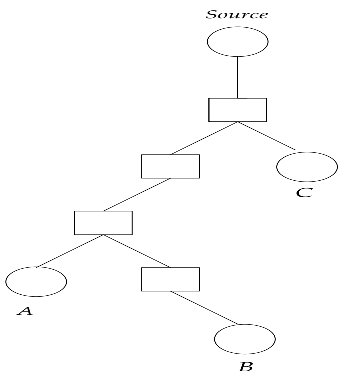

Figure 5: Original tree topology

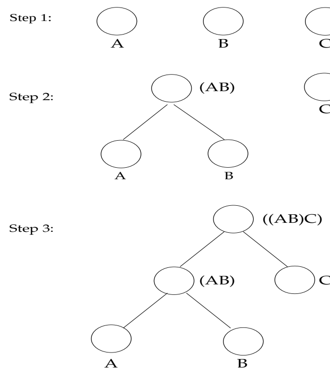

Figure 6: Inference of the logical tree

receivers A, B and C. Information obtained from the loss patterns of the three receivers indicates that A and B are more closely located than A and C or B and C. We thus aggregate A and B into a single "macro-node" (AB). Next, (AB) and C are aggregated to yield the logical tree $((AB)C)$ of figure 6.

Application of the aggregation techniques outlined above requires knowledge of two things: first, we need a selection criteria that is indicative of how closely located receivers are in the tree and second we need to know how many receivers are to be aggregated together into a single representative macro-node at each step of the tree building process. Real shared loss probabilities described in section 2 are used as selection criteria. We first develop the principles behind identifying a pair of receivers to be coalesced at each step of the selection process, thus yielding a binary tree and then generalize the principles to reconstruct trees with arbitrary fan-out at each interior node.

3.1 Binary trees

A binary tree is one in which every interior node in the tree has at most two children. As we coalesce a pair of receivers together at every step our algorithm reconstructs a logical binary tree in which every interior node has exactly two children.

Using the selection criteria defined in Section 3.1, the tree inference algorithm works as follows:

Input: A set of receivers $S = \{1, 2, ..., N\}$ with lossprints $L_1, L_2, ..., L_n$.

1. Compute the probability of seeing real shared losses between all pairs of receivers from the set S.

2. The pair of receivers, A and B, with the maximum probability of seeing real shared losses are combined together into a single macro-node (AB). Set $L_{(ab)} = L_a \cap L_b$ and replace A and B by (AB) in S.

3. Repeat the above steps until all the receivers in S have been fused into a single tree.

Our tree inference algorithm employs a greedy strategy of making the most likely merger at every step. Our results indicate that such a strategy works well in practice. Future work could look into algorithms that consider correlations across multiple nodes. In recent work, [43] compare the performance of top-down and bottom-up clustering algorithms for the reconstruction of the logical tree topology. Their results indicate that a bottom-up approach yields better results.

3.2 Arbitrary tree topologies

In the binary trees reconstructed in the previous section, each interior node has a fan-out of exactly 2. As such, the pair of nodes yielded by the selection criteria are always aggregated as sibling nodes and represented by their parent node for further aggregation. In an arbitrary tree topology, interior nodes have a fan-out of two or more. The selected pair of nodes can thus be aggregated either as sibling nodes as in the case of binary trees, or one of the selected nodes could be the parent node of the other. The two alternatives can be seen in the aggregation of node C and macro-node (AB) in figure 7.

The ability to distinguish between these two cases stems from the observation that in case 1 the probability of seeing real shared losses between A and B should, under ideal circumstances, equal the probability of seeing real shared losses between macro-node (AB) and node C i.e $p_s((ab), c) = p_s(ab)$. In case 2 the probability $p_s(ab)$ will be greater than

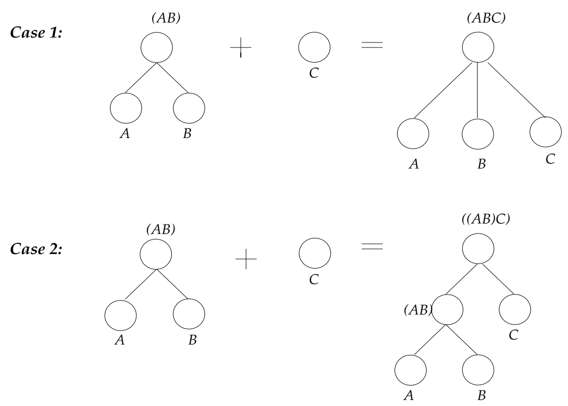

Figure 7: Possible relationships between a pair of nodes to be coalesced for arbitrary trees

$p_s((ab), c)$ because A and B share an additional link not shared by C. This added link adds to the real shared losses between A and B on account of which $p_s(ab) > p_s((ab), c)$. We could thus distinguish between the two subtrees by making the following check :

If $p_s((ab), c) = p_s(ab)$ then the nodes are coalesced as in case 1 else the nodes are coalesced as in case 2.

In reality, since we use the measured losses in order to approximate the probabilities P_{ab}, $P_{a\bar{b}}$ and $P_{b\bar{a}}$, the equality criteria for case 1 are too rigid. Strict adherence to the above rules would result in incorrect aggregations. In order to accommodate a certain amount of variation, we would like to identify situations in which $p_s((ab), c)$ "almost" equals $p_s(ab)$. We thus define an error margin α and modify the above rules to :

If $p_s((ab), c)$ is within $\alpha\%$ of $p_s(ab)$ then the nodes are coalesced as in case 1 else the nodes are coalesced as in case 2.

This decision rule could result in incorrect aggregations being made for subtrees as in case 2 if the additional real shared losses between A and B are responsible for less than $\alpha\%$ of the probability of seeing real shared losses between A and B. As α will typically be low, such errors will only occur if the loss rate along a shared link is very low. As the purpose of these aggregations is to identify nodes that can be grouped together for

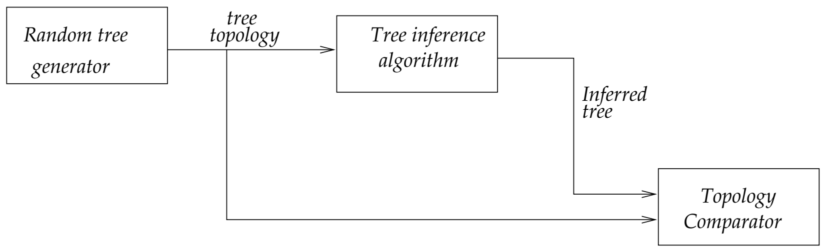

Figure 8: Implementation modules

the purpose of local loss recovery etc, such aggregations although not exact are actually acceptable because the low loss rate link is not the bottleneck causing loss, the problem links, if any, are further upstream and shared by receiver C i.e. for the purpose of local recovery A,B and C should be aggregated together and treated as belonging to the same loss recovery group.

3.3 Evaluation

Implementation

The implementation modules for the algorithms described in the previous sections are shown in Figure 8. For a given random tree topology, packets are dropped along each link in accordance with the link loss rate. The resultant receiver lossprints are fed to the tree inference module. The inferred tree is then compared to the original tree.

Random Tree Generator

The random tree generator module outputs a tree topology which is used to test the tree inference algorithms. Input parameters to this module are: maximum number of nodes ($Nodes_{max}$), maximum number of leaf nodes ($Leaves_{max}$) and the maximum fan-out ($Fanout_{max}$) of every node in the tree. The tree generator algorithm works as follows :

- Initially the tree consists of only the root at level 0. The number of children generated by the root is chosen at random from the range $[1 - Fanout_{max}]$. These child nodes are added to the tree as level 1 nodes.

- Each newly added child node can be either a leaf or an interior (non-leaf) node. A node may be a leaf node with probability $\frac{Leaves_{max} - Leaves_{current}}{Nodes_{max} - Nodes_{current}}$, where $Leaves_{current}$ and $Nodes_{current}$ are the number of leaves and nodes in the tree so far. Thus, as we approach the desired number of nodes in the tree, nodes have a high probability of being leaf nodes. If $Nodes_{max} = Nodes_{current}$ then, a node is a leaf node with probability 1. These heuristic rules ensure that the tree generation process terminates.

- An interior node adds child nodes to the next level in the tree. The number of child nodes generated by an interior node is selected at random in the range

  $[1, min(Fanout_{max}, Nodes_{max} - Nodes_{current})]$. In this way, starting from the root, child nodes are added to successive levels in the tree. The process terminates when the lowest level in the tree has only leaf nodes.

Results

Figures 9 amd 10 plot the performance of the tree inference algorithm when applied to binary trees. We plot the probability of correctly inferring the tree for an increasing number of packets transmitted at the source. As the number of transmitted packets increases, the number of loss samples collected at the receiver's end increases and the approximated probability of seeing real shared losses approaches its true value. We would thus expect

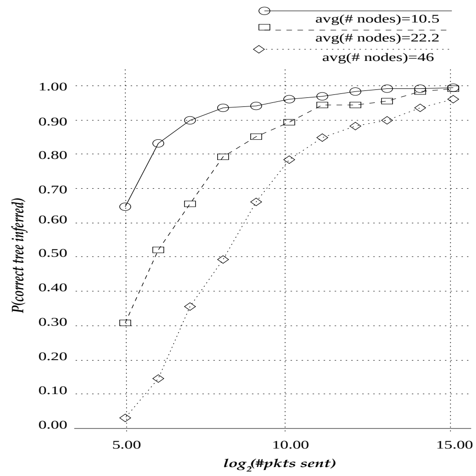

Figure 9: Binary trees: individual link loss rate is selected at random in the range [0%,10%]

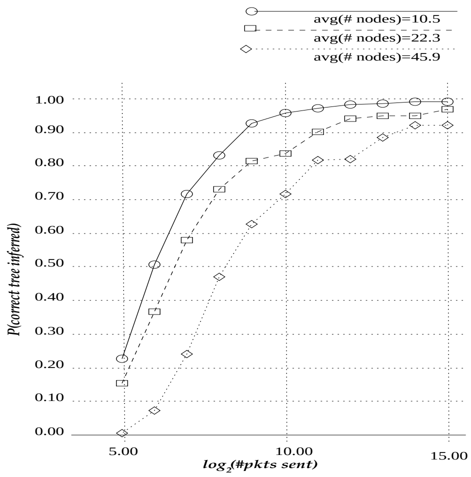

Figure 10: Binary trees: individual link loss rate is selected at random in the range [0%,5%]

the probability of inferring the correct tree to approach unity as the number of collected samples increases. Figures 9 and 10 plot the observed results for different sized trees with the link loss-rates selected as a uniform distribution within a selected range. We see that the observed probability of correctly inferring the tree does in fact converge towards one with an increasing number of transmitted packets.

In order to emphasize the need for real shared loss probabilities as metric for topological locality and the failure of shared loss probabilities in correctly evaluating topological locality, graphs 11 and 12 present simulation results comparing the performance of the tree inference algorithm using real shared loss probabilities versus that using shared loss probabilities.

Graphs 11 and 12 plot the probability of correctly inferring the tree for increasing numbers of packets transmitted at the source. As can be seen, for increasing numbers of collected loss samples, the probability of correctly inferring the tree approaches unity when using real shared loss as metric but not so in the cases using shared loss as metric. Even though one of the shared loss metric algorithms appears to converge to unity in graph 11,

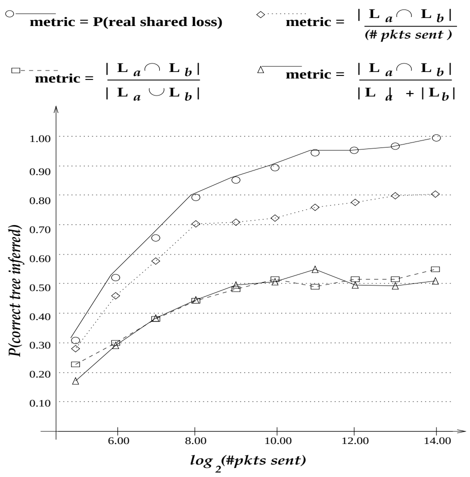

Figure 11: Tree inference algorithm for binary trees with 24 nodes and link loss rates uniformly selected in the range [0-10]%

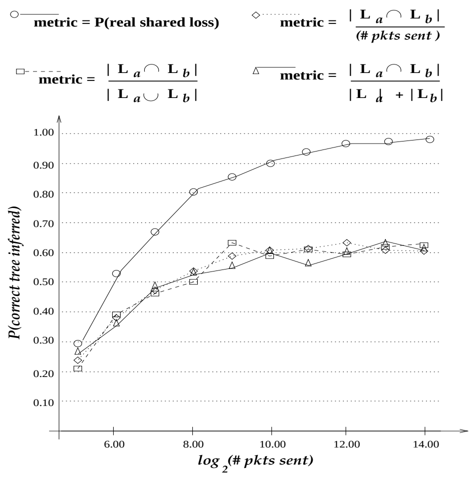

Figure 12: Tree inference algorithm for binary trees with 24 nodes and link loss rates uniformly selected in the range [0-20]%

graph 12 shows that for higher link loss rates the algorithm's performance drops drastically.

3.4 Tree inference algorithm as a network measurements tool

The tree inference algorithm's assumption of global knowledge of receiver lossprints prevents its use in practical distributed protocols. Even so, the algorithm is of potential use in its current form for network monitoring, debugging, and performance characterization using off-line processing. In this section we briefly outline two example scenarios within which the tree inference algorithm has served as a tool for multicast based inference of network characteristics.

Locating bottleneck links in a multicast tree

Transmission of a packet from a source to a receiver involves forwarding the packet along a series of consecutive links. Each link has a maximum rate at which it can forward packets. The maximum rate of the slowest link along the chain determines the bottleneck bandwidth along a given path. The ability to measure this bottleneck bandwidth value

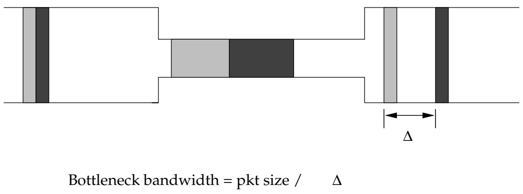

Figure 13: Packet pairs flowing through a bottleneck link. The vertical dimension is bandwidth, horizontal dimension is time

stems from the observation that as a packet is transmitted along a link, it is "spaced" out in time depending on the transmission rate of the link with the amount of spacing being inversely proportional to the capacity of the link [14]. The basic idea behind the packet-pair mechanism is as follows: if two probe packets travel together such that they are adjacent at the bottleneck link, with no packets intervening between them then, on emerging from the bottleneck link the inter-packet spacing will be proportional to the transmission time of the first packet over the bottleneck. This can be seen in Figure 13.

A significant amount of work [2, 16, 4, 29, 31] has been done in the area of bottleneck bandwidth estimation along a unicast path using different forms of the same packet probe idea.

In [34], we applied Paxson's Packet Bunch Mode (PBM)[31] technique for bottleneck bandwidth estimation to multicast communication. The traffic source in the multicast tree transmits a stream of back-to-back probe pairs. Each receiver measures the arrival times of packets at its end and uses the PBM algorithm to infer the bottleneck bandwidth of the path between itself and the source. Note that this method does not, by itself, in any way indicate where along the path the bottleneck is located. By combining the information obtained by the bottleneck estimation technique and the tree inference algorithm it is however possible to narrow down the possible locations of the bottlenecks in the multicast tree as follows:

Receivers appear as leaves in the reconstructed logical tree. PBM gives us an estimate of the bottleneck bandwidth between the source and each leaf node. The bottleneck

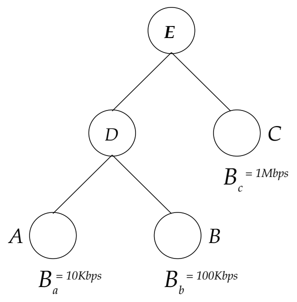

Figure 14: The bottleneck bandwidth seen by each interior node equals at least the maximum of the estimate of its downstream nodes

bandwidth seen by each interior node is at least equal to the maximum of the bottleneck bandwidth estimates seen by each of its downstream receiver nodes.

This can be easily understood by the simple example in Figure 7. Node D has to see a maximum rate of at least 100Kbps in order for receiver B to see a bottleneck rate of 100Kbps. This implies that the bottleneck limiting the rate seen by receiver A lies along the branch AD. Similarly node E has to see a rate of at least 1Mbps and hence the bottleneck seen by receiver B lies some where along the path $ED - DB$. We cannot narrow down the location of the 100Kbps bottleneck link any further because having removed link DA from consideration we are left with the same case as the unicast path and hence we cannot obtain a more precise estimate using only information obtained at the end hosts.

Estimation of individual link loss rates

In [3] the authors explore the use of end-to-end multicast traffic as measurement probes to infer network-internal characteristics and develop a Maximum Likelihood Estimator for packet loss rates on individual links based on the end-to-end losses observed by multicast receivers. Their work assumed a priori knowledge of the logical topology of the multicast routing tree. The authors have since employed our tree inference algorithm in order to satisfy this assumption.

4 A Group Formation Protocol

In a practical protocol, the tree inference algorithm's assumption of global knowledge of receiver lossprints would lead to infeasible control traffic overhead. Fortunately, the use of subgroups in multicast applications and protocols is an optimization that need not be perfect at all times to dramatically enhance performance. Thus, we can assume a coarse grained clustering of co-located receivers without requiring detailed knowledge of the exact routing tree topology. In this section we apply the lessons learned from the extreme case analysis of Section 2 to the design of a practical Group Formation Protocol (GFP) that achieves this coarse-grained clustering i.e. we relax the exchange of information at the cost of decreased topological accuracy and yet do so in a controlled fashion. .

The goal of GFP is to have the receiver set self-organize into a multi-level hierarchy of multicast groups wherein individual groups correspond to the homogenous sub-regions within the heterogeneous multicast tree and the hierarchy imposed is congruent with the underlying multicast routing tree. The metric used to identify group structures is the real shared loss probabilities introduced in Section 2. We have implemented a prototype protocol to explore the question of whether this basic loss based approach is feasible. A number of variations and enhancements to this basic approach are conceivable and would provide different tradeoffs.

4.1 Terminology

In order to facilitate the explanation of the detailed working of GFP, we first introduce some requisite terminology:

- Receiver LossPrint (LP_i) : The ordered list of packets lost by receiver I.
- Group LossPrint (LP_g) : The ordered list of packets lost by the group as a whole as reported by the group's representative.
- Receiver's Working LossPrint ($LP_i - LP_g$) : The ordered list of packets lost by a

receiver within its current group which is estimated as the losses seen by a receiver but not by its group representative.

- Receiver's Working Loss Rate (LR_i) : Receiver I's loss rate calculated with respect to its working lossprint. LR_i estimates the losses experienced by receiver I *within* its current data group.

A lossprint further includes the sequence number of the last packet received.

4.2 Group Formation

The initial multicast group structure associated with GFP consists of a single, well known, pervasive control group and a single data group. Using GFP the single data group evolves into multiple topologically localized data groups. At all times, a receiver is a member of a single data group and the common control group.

In our prototype protocol, the source transmits probe packets onto the common control group at a low rate that can be handled by even the lowest bandwidth path in the tree. An alternative future implementation could, in some way, use the actual data transmission rather than inject additional probe traffic. The losses experienced by the receivers on this control group drive the group formation process. While the group formation process is driven by the losses experienced on the common control group, the protocol messages (e.g.: group initiation messages) sent out by a receiver are multicast only to its local data group rather than globally. GFP is thus run independently within each data group but always with respect to the losses observed on the control group on account of which the derived structure corresponds to the routing tree rooted at the source.

Figure 15 illustrates the overall operation of GFP. A receiver I initiates the formation of a new data group G_1 because it experiences significantly high loss caused by $Bottleneck_1$ along the path from the source to itself. All the receivers downstream from $Bottleneck_1$ join the proposed group G_1. G_1 lies at the first level in the hierarchy of data groups. The existence of an additional bottlenecks causing loss ($Bottleneck_2$) within the

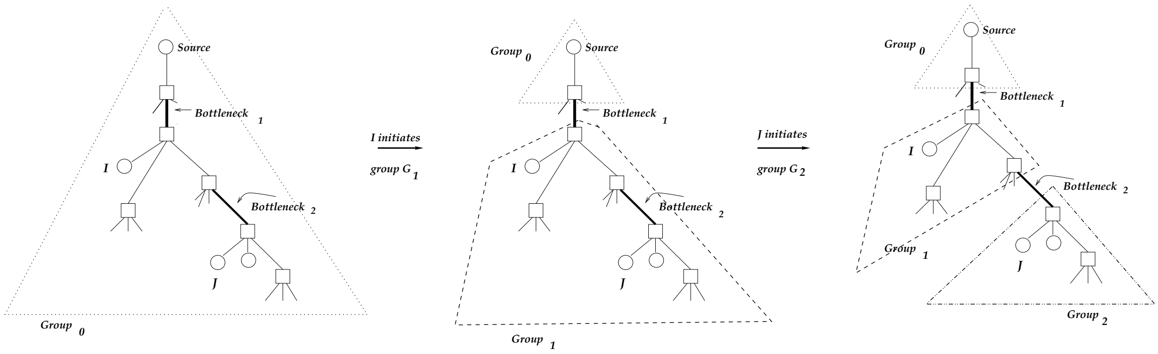

Figure 15: Evolution of a multi-level hierarchy

group G_1 causes receiver J to subsequently initiate the formation of a new group G_2. This group initiation message is multicast to the group G_1. The subset of receivers in G_1 that are downstream from $Bottleneck_2$ now leave G_1 and join G_2. G_2 lies at the second level in the hierarchy of data groups. This process continues until every receiver settle into a stable state data group *within* which there are no significant bottlenecks.

We first describe in detail how GFP operates within a single data group,how representatives are elected and finally how the pieces fit together to yield a multi-level hierarchy.

Group formation within a single data group

A receiver I builds its individual lossprint LP_i from the losses experienced on the control group. Every local data group has a group representative (described in Sec.4.2) that periodically multicasts a group lossprint (LP_g) onto its data group. A receiver factors out the reported group lossprint from its individual lossprint to obtain its working lossprint ($LP_i - LP_g$) that estimates the losses suffered by the receiver within its current data group.

Every receiver sets a timer as a linearly increasing function of its working loss rate. On timeout, a participant may initiate the formation of a new multicast group, join a previously proposed group, or take no action. A receiver may initiate the formation of a new data group only if its *working* loss rate exceeds a predefined loss rate threshold LR_{min} Group initiation is done by multicasting an INIT(G) message onto the receiver's current data group proposing the formation of a new data group G. The INIT(G) message includes the group initiator's working lossprint.

For every incoming INIT message, a participant I computes $p_s(i,g)$, the real shared loss probability between itself (receiver I) and the proposed group G using the model described in Section 2. $p_s(i,g)$ estimates the loss that receiver I shares with the proposed group G along the common path from the source to the closest common ancestor of I and the group's initiator. Until timer expiry, a receiver I tracks the proposed group G_{max} with which it shares the maximum value of $p_s(i,g)$, i.e. p_s^{max} and G_{max}. On timeout, if the receiver's value of p_s^{max} exceeds a predefined threshold *THRESH*, it joins the group G_{max}. If not, it initiates the formation of a new group G_{new} provided its working loss rate LR_i exceeds LR_{min}.

In summary, on timeout, a receiver I which is currently a member of the data group G performs the following check:

If ($p_s^{max} >$ *THRESH*)

    leave G

    join G_{max}

else-if ($LR_i > LR_{min}$)

    # I initiates the formation of G_{new}

    Multicast *INIT* message (*INIT* $/G_{new}/(LP_g - LP_i)$) onto G

    leave G

    join G_{new}

else

    do nothing

Selection of Group Representatives

This section describes the election of group representatives and their role in GFP. The ideal group lossprint would consist of exactly those losses incurred along the shared path between the source and the closest common ancestor of all the group members. While the ideal group lossprint could be computed as the intersection of each receiver's lossprint, this requires global knowledge of receiver lossprints which is infeasible in practice. Instead, GFP elects a single representative per data group that is responsible for announcing the group lossprint to its entire data group We choose the lowest loss rate receiver within the group to serve as the representative because such a receiver's lossprint most closely matches the ideal group lossprint. Further, the lowest loss rate member of the group is the best connected to the source and is likely to be the further upstream in the underlying routing tree. As shall be described in Section 6 such a receiver plays a key role in achieving reliability and congestion control in a number of proposed multicast applications and protocols.

To ensure that the initiator of a new data group has the lowest loss rate within that group timer values are selected as a linearly increasing function of the receiver working loss rates. Thus, the initiator can serve as the group representative thereby greatly simplifying the election of group representatives.

The use of a representative-based model induces a single point of failure. We rely on the application of "soft state" principles to achieve robustness in the face of representative crashes. A representative periodically multicasts *REP* messages announcing the group lossprint to its data group. Every group member sets a timer with value directly proportional to its working loss rate. If a receiver sees a *REP* message before its timer expires, it cancels its timer. If not, then on time out, a receiver assumes that the representative has stopped or crashed and starts transmitting *REP* messages. This ensures that when a representative crashes, the next lowest loss rate receiver is promoted to representative.

In addition to periodic *REP* messages, group representatives periodically multicast *INIT* messages onto the common control group at a very low rate. *INIT* messages contain the representative's local data group address and the representative's lossprint LP_r. As described in sections 4.3 and 4.4, these messages allow for adaptation to changing routing topologies, incorrect grouping decisions etc. Further, these *INIT* announcements allow late joiners in the session to learn about existing groups and find their correct data group through the normal application of the GFP procedure described in the previous section.

As in any "announce-listen" [7, 37, 1, 38, 32, 23] type protocol, the rate at which these periodic *INIT* control messages are announced involves a trade-off between the amount of the receiver's bandwidth allocated to control traffic and the latency with which receivers react to dynamic changes in routes, recover from incorrect grouping decisions etc.

Evolution into a multi-level hierarchy

We now describe how the group formation process within a single group and the representative election process work together to yield a multi-level hierarchy. The initial data group in which all receivers start out constitutes the first level in the hierarchy of data groups. There is no group representative and the group lossprint is empty. Thus, a receiver's initial working lossprint is its own lossprint itself. The presence of bottlenecks within the original data group that are not shared by the entire group causes subsets of receivers below common bottlenecks to break off and form new data groups by the application of the process described in section 4.2. These new groups form the second level members of the hierarchy.

On formation of a new data group, the group initiator serves as the group representative and periodically multicasts its individual lossprint onto the new data group. The representative's lossprint serves as the group lossprint. As the group initiator (representative) is itself downstream of the common bottleneck, the group lossprint now includes the losses suffered along the bottleneck. Consequently, the working lossprints computed by the new group members no longer reflect the losses suffered along the shared bottleneck thus yielding lower working loss rates. If however, there still exist lossy paths *within* a new data group then the subset of receivers below such paths will still have significantly high working loss rates and will break off into a new data group one level lower in the hierarchy. If there are no more bottlenecks within the data group, the receiver working loss rates will be below

LR_{min}, no new groups will be initiated and the group structure will stabilize.

GFP is thus run independently within each data group and the multi-level hierarchy evolves in parallel across the receiver set until every receiver settles into its final stable state data group within which there are no significant bottlenecks.

4.3 Recovery from incorrect grouping decisions

The limited observations used to make decisions and the limited number of loss samples per exchanged lossprint could result in occasional incorrect grouping decisions. Hence a recovery process is required to correct bad grouping decisions. The recovery algorithm is a two-step process. First, the receiver detects that it has made a mistake in joining its current data group. Having done so it then chooses a better group to join.

A member I of a group G_{l-1} at level $l-1$ joins a group G_l at level l because the estimated shared loss between the representative of G_l and receiver I along the common path from the representative of G_{l-1} exceeds the value *Thresh*. Thus, a receiver I in a level l group should estimate a shared loss rate of at least l times *Thresh* between itself and its current group representative. For every incoming *REP* message containing group lossprint LP_g, a receiver I with lossprint LP_i recomputes the real shared loss probability $p_s(i,g)$. If receiver I finds that its estimate of $p_s(i,g)$ is repeatedly less than $l * Thresh$, it concludes that it is in an incorrect data group and leaves the group G_l. To decide upon which group to join, the receiver simply runs GFP as normal and listens for new *INIT* messages. By listening to the periodic *INIT* announcements on the control group, the receiver can by the application of the normal GFP procedure discover its appropriate data group.

4.4 Adapting to changes in link loss rates and routing topologies

We expect GFP to operate over large time-scales for multicast sessions that are fairly long-lived. As such, the GFP group structure needn't adapt to transient network congestion and other short-lived variations in network conditions. Yet, GFP must adapt to long-lived variations in network conditions and routes. On account of changing link loss rates and routing topologies, new bottleneck links may appear and existing ones dissapear. The emergence of a new bottleneck link raises no new problems because the downstream receivers will break off into a new data group by the normal application of GFP. The dissapearance of a previously existing bottleneck however implies that two data groups should ideally be merged to control the number of data groups. The process for merging two equivalent groups is as follows:

- For every REP message received on its local data group G, a receiver I computes the real shared loss probability $p_s(i,g)$.
- For every incoming $INIT(G_{new})$ message received on its control group, a receiver I computes the real shared loss probability $p_s(i,new)$.
- If $p_s(i,new)$ is repeatedly within $\pm 10\%$ of $p_s(i,g)$ then the groups G_{new} and G are treated as being essentially equivalent and the receivers switch to the group with the lower IP address.

Thus, GFP builds a hierarchy of multicast groups wherein every receiver *independently* discovers its local group, recovers from incorrect grouping decisions and monitors periodic group announcements such that the group structure as a whole adapts to variations in network conditions.

5 Evaluation

5.1 Accuracy of GFP

A true and comprehensive evaluation of any scalable network protocol can only be achieved by its actual large scale deployment over the Internet. Given the practical difficulties involved in accomplishing this, we instead verify, through simulation, that GFP does in fact faithfully structure receivers in accordance with the underlying topology for a range of workloads and network topologies.

We implemented GFP in the VINT network simulator *ns* [24], using the TIERS topology generator [10]. TIERS models a network using three levels of hierarchy, or tiers, referred to as WAN, MAN and LAN levels. MAN nodes represent *stub* domains that only carry traffic that originates or terminates in their domain and WAN nodes represent *transit* domains that interconnect lower level stub domains.

A GFP participant computes the value of $p_s(i, g)$ afresh for every incoming lossprint and makes a group initiation/join decision based on this value. A participant's decision causes the hierarchy to evolve by a single level, i.e. the information conveyed by a single lossprint controls the evolution of the hierarchy by a single layer. It is thus important to understand how much loss information a receiver must acquire and exchange in order to make useful grouping decisions as this impacts the protocol's rate of convergence to a final stable hierarchy.

Figure 17 plots the probability with which the hierarchy built by GFP correctly matches the underlying tree for different lossprint lengths for network topologies of different sizes. The computed hierarchy is said to be *correct* if for every group G in the final hierarchy, the closest common ancestor of all the group G members has only members of G as downstream receivers. As expected, GFP performance improves with increasing lossprint lengths because more information is utilized per grouping decision. As GFP serves as a protocol optimization rather than being critical for correct protocol operation we need not always achieve perfect results and a trade-off can be made between the desired degree of accuracy and the amount of bandwidth allocated to traffic due to *INIT* messages.

The ability to extract topological information depends on the distribution of link loss rates along the paths from the source to the different receivers. A typical transmission might originate from a low loss rate environment such as a LAN, traverse a few shared wide-area, lossy links (eg: trans-atlantic, trans-continental links) and be ultimately delivered to end receivers on well-connected LANs. Intuitively, GFP should perform well in such a setting on account of the significant shared loss along the wide area path. At the same time, GFP should still perform satisfactorily in a worst-case scenario where the range of

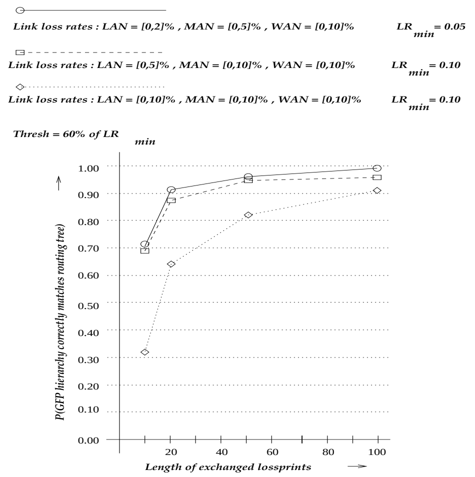

Figure 16: GFP performance for different distributions of link loss rates

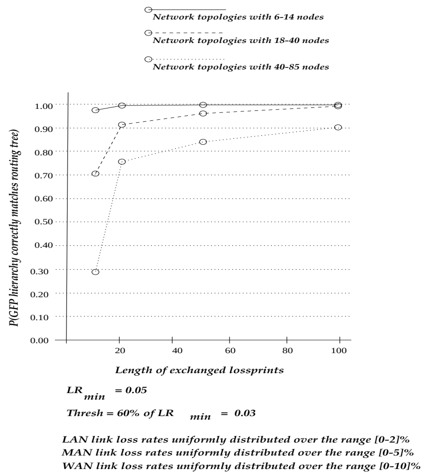

Figure 17: GFP performance for different size topologies

link loss rates is not clearly demarcated between the shared and non-shared path.

Figure 16 illustrates GFP performance for different ranges of link loss rates for the three different types of links generated in TIERS topologies: wide area (WAN) links, local area (LAN) links and intermediate MAN links. The data indicates that the best performance is obtained for cases where the difference in loss rates between different link types is the most pronounced and performance degrades as this difference is reduced thus corroborating our intuition.

5.2 Effect of protocol parameters

The two key parameters that control the operation of GFP are the minimum working loss rate threshold LR_{min} and *Thresh* used in making group initiation / join decisions. LR_{min} controls the degree of heterogeneity with local data groups and *Thresh*, the shape of the hierarchy. In this section, we study how the structure of the final hierarchy is affected by varying these parameters.

The value of LR_{min} controls the permissible degree of heterogeneity within a single data group. If there exists a path within a data group with loss rate greater than LR_{min}

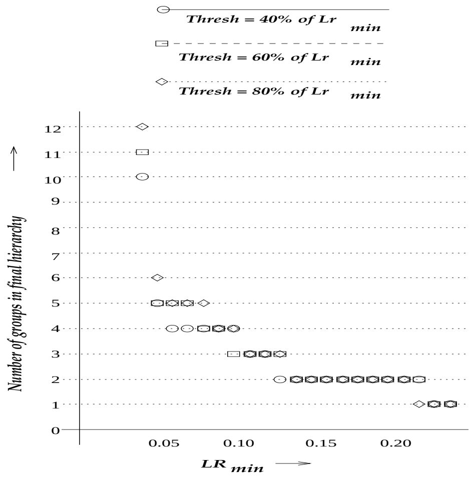

Figure 18: Effect of LR_{min} on number of groups in the final hierarchy

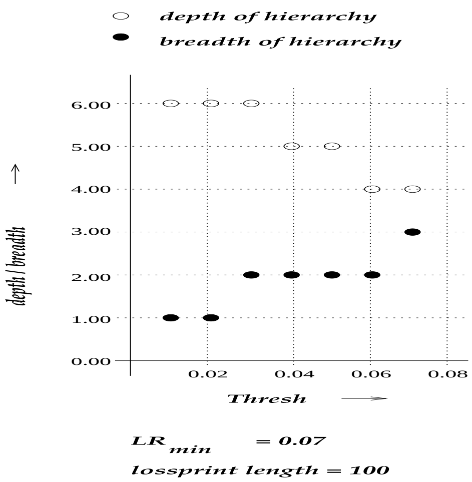

Figure 19: Effect of $Thresh$ on the shape of the final hierarchy

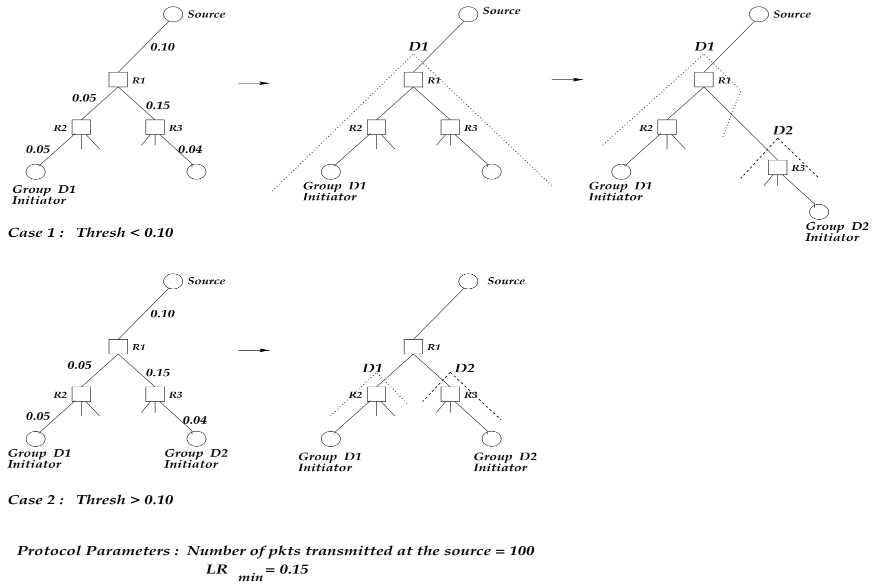

Figure 20: $Thresh$ affects the shape of the hierarchy

then a receiver downstream from that path initiates the formation of a new group. The hierarchy settles when the loss rates of receivers within any single group is less than LR_{min}. For a given tree topology, LR_{min} also influences the number of members in each group and the total number of data groups in the final hierarchy. A high value of LR_{min} results in fewer data groups with a larger more heterogeneous member set while a lower value results in a larger number of smaller more homogeneous groups. Fig 18 illustrates this trade-off. For a topology with 1 WAN, 4 MANs and 8 LANs (with a total of 40 nodes) the graph plots the number of groups (including the original data group) in the final stable hierarchy for increasing values of LR_{min}. Link loss rates are set to 4%, 6% and 10% for LAN, MAN and WAN links respectively. As expected, the number of groups decreases with increasing values of LR_{min}.

Thresh controls the shape of the final hierarchy. This is illustrated in Figure 20. The numbers next to each link indicate the link loss rate. The value of LR_{min} is set to 0.15. If *Thresh* is lower than 0.10, the subtree rooted at R1 breaks off into a single second level data group D1 because all the receivers downstream from router R1 share at least 0.10 loss along the common path from the source to R1. The subtree rooted at R3 subsequently breaks off into a third level data group D2 because the link R1-R3 has a loss rate greater than LR_{min}. Thus the final hierarchy consists of a 3 level hierarchy of data groups. A value of *Thresh* greater than 0.10 causes the subtrees rooted at R2 and R3 to break off into two distinct data groups D1 and D2 both of which reside at the second level in the hierarchy yielding a two level hierarchy in the stable state. Thus, a high value of *Thresh* yields a hierarchy that is more "broad" than "deep" while a lower value of *Thresh* will yield one that is more "deep" than "broad". Figure 19 plots simulation results obtained by running GFP on a binary tree topology of depth seven with a single LAN at every level of the tree. All the link loss rates were set to 5%. The graph plots the depth and breadth in the final hierarchy for increasing values of *Thresh*. The depth of the hierarchy is the number of levels in the final stable state GFP hierarchy. The breadth of the hierarchy is the maximum number of data groups at any given level in the final hierarchy.

6 GFP as a reusable component

The efficacy of end-to-end multicast transport protocols depends critically upon their ability to scale efficiently to large numbers of possibly heterogeneous receivers. Several research multicast protocols and applications attempt to achieve this goal by identifying sets of co-located receivers in order to enhance loss recovery, congestion control and so forth. In this section, using a few existing applications and protocols as examples, we describe how GFP serves as an important reusable building block across a wide range of these proposed schemes. While we haven't implemented the proposed schemes or worked out the finer details, we believe the basic approach should work well in practice.

A key challenge in the design of a reliable multicast protocol is its loss recovery algorithm, which has proven difficult to scale to a large number of receivers. For example, the global loss recovery component of SRM multicasts retransmission requests and replies to the entire group and thus scales problematically [33] as the entire tree participates in the recovery process. To solve this problem a number of schemes attempt to achieve *local recovery*. The key idea behind local recovery is to identify loss neighborhoods of receivers that share similar loss patterns and confine error recovery to this neighbourhood without disturbing the rest of the tree.

Liu et al. [22] propose the the use of separate local multicast groups in order to confine the scope of error recovery traffic in SRM. Multiple local groups are associated with a source where each group is responsible for error recovery of one or more lossy links. For a specific source, lossy links are related to one another as either ancestors/descendant or siblings and hence membership of the local groups should ideally be either perfectly disjoint or nested. Shared loss thresholding is used to form local recovery groups. The GFP data groups are essentially the same as Liu's local recovery groups and avoid the shared loss pathology incurred by their use of shared loss thresholding. The desired nested effect can be achieved using GFP if receivers incrementally join new proposed data group without leaving their current ones.

The Reliable Multicast Transport Protocol [21] addresses the loss recovery problem by organizing members into a hierarchy. Internal nodes in the hierarchy called Designated Receivers (DRs) cache data packets for later retransmission. Receivers are grouped into local regions each of which is assigned a DR. Acknowledgments are sent not to the source, but to the DR in th region. RMTP therefore provides both implosion avoidance and local recovery. However for RMTP to perform well, the hierarchy of members must be very closely correlated to the underlying multicast distribution tree and DRs need to be optimally distributed over the tree. The dynamic configuration of receivers and DRs is outlined as future work. The integration of GFP with RMTP enables this dynamic configuration of receivers into the required hierarchy with GFP representatives playing the role of DRs.

A number of emerging research proposals advocate the use of application aware agents or *proxies* to serve as intermediaries between a source and receivers as a means of coping with the heterogeneity inherent in multicast sessions on account of varied network and end host capabilities.

In [5] Chawathe et al. present a general architecture for proxy-based reliable multicast called the Reliable Multicast proXy (RMX) model. Their prototype RMX implementation for a shared whiteboard application for hand-held PDAs relies on the existence of a well-known service cluster that supports the RMX. Algorithms for the dynamic placement of RMX agents is outlined as future work.

Self-Organized Transcoding (SOT) [17] adapts continuous-media applications to varying network conditions by the self organization of groups of receivers with bad reception and provides rate adjustment through the use of transcoders. In SOT, co-located receivers experiencing loss caused by a bottleneck link, elect a representative which will locate an upstream receiver with better reception at the far end of the bottleneck to act as transcoder. The transcoding receiver multicasts a customized version of the stream to a new address and receivers adversely affected by the bottleneck switch to the new group. SOT uses shared loss as the metric for group identification.

Handley [12] proposed a relay-based congestion control architecture for bulk data transfer in which a high loss rate receiver elects itself to act as representative through the use of SRM-style random timers weighted by the number of losses experienced. Representatives initiate the formation of new subgroups and receivers use shared loss in deciding whether to join a proposed subgroup. The elected representative performs an expanding ring search in order to locate an upstream receiver with better reception to act as relay for the new subgroup.

In SOT, RMX, Relay-based congestion control, and in fact, any agent-based scheme, protocol stability requires that *all* receivers within the same loss subtree are co-ordinated in their actions such as switching groups, tuning in to the same proxy/transcoder etc. Further, enlisting such agents raises the problem of placing them intelligently and dynamically throughout the network. GFP enables solutions to both problems: co-ordination is achieved by building a hierarchy of data groups that explicitly corresponds to the different loss subtrees and strategically located representatives enable the intelligent placement of transcoders and proxies.

A multicast-based adaptive web caching infrastructure has been proposed [15, 27] to meet the exponential growth of the World Wide Web. Zhang et al.[27] outline an adaptive web caching system in which Web servers and cache servers self organize into overlapping local multicast groups. A request for a web page is first multicast to the requestor's local group and in the event of a miss, the request is successively forwarded through a chain of overlapping cache groups until it reaches a cache group with the required page or the origin server of the requested page. The GFP yields the desired structure with the group representative's serving as ideal points of overlap. GFP can achieve overlapping hierarchies if group representatives retain their membership in the parent group even after the formation of new lower level groups.

7 Related Work

Route tracing tools developed so far exploit certain features within the routers in order to infer the path from source to destination. The traceroute tool built by Van Jacobson discovers the path between a source and receiver of unicast traffic by using the ttl field of an IP packet header to force intermediate routers to send an error indication (ICMP time exceeded) packet back to the source thus exposing the routers within the network to discover the path between the source and receiver. The pathchar tool, also developed by Jacobson, estimates the bandwidth, delay, average queue and loss rate of every hop between any source and destination on the Internet. Pathchar uses the same basic technique as traceroute and measures the time between the transmission of an IP packet from the source and the return of the corresponding ICMP packet from an intermediate router. Analysis of the timing data reveals the characteristics of each link along the path.

Estimation of the topology of the multicast tree can be done using the tool "mtrace". mtrace discovers the multicast path from a source to a receiver using an MTRACE tracing feature implemented in multicast routers that is accessed as an extension to the IGMP protocol. A trace query is passed hop-by-hop along the reverse path from the receiver to the source, collecting hop addresses, packet counts and routing error conditions along the path, and returning the response to the requestor as a standard unicast packet.

A number of schemes proposed in the research literature rely on the aggregation of receivers in a multicast session for scalability and propose schemes for accomplishing goals similar to those of GFP. In this section we compare the underlying aggregation metric / technique used by these schemes with GFP and its metric of real shared loss probabilities. As group formation is an enabling piece and not the principal goal of much of the work described here, none of these schemes have, to our knowledge, evaluated the accuracy of the group formation process in terms of group structure conforming to the underlying routing structure.

In [12, 17, 22, 27] groups are formed based on the measured *net* shared loss between receivers. Such schemes suffer from the shared loss pathology described in Section 2 which is overcome by our use of real shared loss probabilities.

A number of schemes [6, 41, 35, 18, 13] rely on the use of *time-to-live* or TTL-based scope to limit the reach of traffic. Real-work measurements [41] however indicate that TTL is not a good measure of locality as the number of reachable hosts does not increase linearly with TTL values. Further, TTL count by itself does not provide any information regarding receiver losses nor does it indicate where the bottleneck links lie relative to the different receivers. It might however be possible to combine TTL scoping and loss based approaches such as GFP for enhanced robustness and scalability.

The Tracer protocol [19] uses the MTRACE router function in order to organize the receivers of a multicast group deterministically into a logical tree structure in order to achieve effective error recovery and congestion control. In Tracer each receiver sends an MTRACE query to the source of the tree. With the existing implementation of MTRACE, this could cause scaling problems due to an implosion of MTRACE queries at the source. Further, this places a heavy load on the source which has to unicast replies back to every receiver. In order to improve the efficiency of tracing in Tracer, Levine et al propose the addition of source-based multicast tracing to IGMP.

Static configuration of receivers as in RMTP [21] , the use of service clusters with well known locations [1, 5] for agent-based solutions etc while good first-cut solutions may not scale well to large receiver groups with multiple service clusters without the dynamic and optimal configuration of receivers.

In [3] the authors explore the use of end-to-end multicast traffic as measurement probes to infer network-internal characteristics and develop a Maximum Likelihood Estimator for packet loss rates on individual links based on the end-to-end losses observed by multicast receivers. Their work assumes a priori knowledge of the logical topology of the multicast routing tree and is thus orthogonal and complimentary to GFP.

Finally, recent research proposals [28, 20, 8] advocate the inclusion of new forwarding services within the network in order to better support end-to-end transports. These schemes enable the acquisition of topological information at the cost of modifying the existing IP service model and are an alternative approach to our end-to-end solution. It is worthwhile to explore both approaches to better understand the trade-offs involved.

8 Conclusions and Future Work

A recurrent theme running through a large number of proposed research multicast protocols is the notion that it is possible to design solutions that both scale efficiently and handle heterogeneity gracefully by adopting a *divide and conquer* style approach that aims at topologically localizing problems. In this paper, we have studied the fundamental feasibility of such an approach and demonstrated that through the application of a simple probabilistic model it is indeed possible to infer detailed knowledge of the internal structure of a multicast distribution tree. We applied this model to the design of a practical protocol primitive, namely a Group Formation Protocol whereby receivers self organize into a multi-level hierarchy of topologically localized multicast subgroups. Our simulation results indicate that GFP faithfully captures the structure of the underlying topology.

GFP works with the existing IP service model and preserves the forwarding semantics of the current Internet architecture. We believe that GFP can serve as a reusable core protocol building block across a wide range of end-to-end multicast applications and protocols proposed in the research literature.

We envision three principal directions along which work on GFP can proceed: refining and evaluating the performance of GFP itself, validating the potential of GFP as a reusable component in existing multicast applications and protocols and finally, GFP plays a key role in our current work on the development of a framework for topologically sensitive multicast reliability and congestion control that we term *scattercast*.

The key motivation for scattercast is to tailor the original data delivery process to meet the varied receiver characteristics in the hope that achieving this will greatly ease subsequent recovery issues such as packet retransmission etc. The use of GFP within such a framework allows us to separate out the homogeneous pockets of receivers within a session into different multicast data groups and organize them into a multi-level hierarchy. The original data transmission is then achieved through a combination of unicast and multicast forwarding. The source *unicasts* data to the group representatives at the second level in the hierarchy. Every representative multicasts the data within its local data group and unicasts data to its sibling representatives at the next level in the hierarchy. Such a framework enables the implementation of reliability and flow/congestion control at two levels: *within* each homogeneous data group (e.g: using SRM style global recovery, transmitting at the lowest common denominator rate etc) and *between* homogeneous regions by the coarse grained clustering of receivers (GFP). Further, scattercast allows us to leverage off existing Internet-compatible unicast transport mechanisms for congestion control and reliability along the path between data group representatives.

Bibliography

[1] Elan Amir, Steven McCanne, and Randy H. Katz. An active service framework and its application to real-time multimedia transcoding. In *Proceedings of SIGCOMM '98*, Vancouver, BC CANADA, September 1998.

[2] Jean-Chrysostome Bolot. End-to-end packet delay and loss behavior in the Internet. In *Proceedings of SIGCOMM '93*, pages 289–298, San Francisco, CA, September 1993. ACM.

[3] R. Caceres, N. Duffield, J. Horowitz, D. Towsley, and T. Bu. Multicast Based Inference of Network-Internal characteristics: Accuracy of Packet Loss Estimation. In *Proceedings IEEE Infocom '99*, New York, NY, March 1999.

[4] Robert Carter and Mark Crovella. Measuring bottleneck link speed in packet-switched networks. Technical Report TR-96-006, Boston University, Boston, MA, March 1996.

[5] Yatin Chawathe, Steve Fink, Steven McCanne, and Eric Brewer. A Proxy Architecture for Reliable Multicast in Heterogeneous Environments. In *Proceedings of ACM Multimedia '98*, September 1998.

[6] Dah-Ming Chiu, S. Hurst, M. Kadansky, and J. Wesley. TRAM: A Tree-based Reliable Multicast Protocol. Technical Report SML-TR-98-66, Sun Microsystems Laboratories, Palo Alto, CA, July 1998.

[7] David D. Clark. The design philosophy of the DARPA Internet protocols. In *Proceedings of SIGCOMM '88*, Stanford, CA, August 1988. ACM.

[8] Adam Costello and Steven McCanne. Search Party: Using Randomcast for Reliable Multicast with Local Recovery. In *Proceedings IEEE Infocom '99*, New York, NY, March 1999.

[9] Stephen Deering and David Cheriton. Multicast routing in datagram internetworks and extended LANs. *ACM Transactions on Computer Systems*, 8(2):85–110, May 1990.

[10] Matthew Doar. A better model for generating test networks. In *Proceedings of GLOBECOM '96*, London, UK, November 1996.

[11] Sally Floyd, Van Jacobson, Steven McCanne, Ching-Gung Liu, and Lixia Zhang. A reliable multicast framework for light-weight sessions and application level framing. In *Proceedings of SIGCOMM '95*, pages 342–356, Boston, MA, September 1995. ACM.

[12] Mark J. Handley. A congestion control architecture for bulk data transfer, September 1997. Presentation at the Reliable Multicast Research Group meeting.

[13] Markus Hofmann. Enabling group communication in global networks. In *Proceedings of Global Networking*, Alberta, Canada, June 1997.

[14] Van Jacobson. Congestion avoidance and control. In *Proceedings of SIGCOMM '88*, Stanford, CA, August 1988.

[15] Van Jacobson. SIGCOMM '95 Middleware Workshop: How to kill the Internet, August 1995.

[16] S. Keshav. *Congestion Control in Computer Networks.* PhD thesis, University of California, Berkeley, September 1991.

[17] Isidor Kouvelas, Vicky Hardman, and Jon Crowcroft. Network adaptive continuous-media applications through self organised transcoding. In *Proceedings of the Network and Operating Systems Support for Digital Audio and Video*, Cambridge, U.K., July 1998.

[18] Brian Neal Levine, David B. Lavo, and J.J. Garcia-Luna-Aceves. The case for reliable concurrent multicasting using shared ACK trees. In *Proceedings of ACM Multimedia '96*, Boston, MA, November 1996. ACM.

[19] Brian Neal Levine, Sanjoy Paul, and J.J. Garcia-Luna-Aceves. Organizing multicast receivers deterministically by packet-loss correlation. In *Proceedings of ACM Multimedia '98*, Bristol, UK, September 1998. ACM.

[20] Dan Li and D. R. Cheriton. OTERS (On-Tree Efficient Recovery using Subcasting): A reliable multicast protocol. In *Proceedings of the Sixth IEEE International Conference on Network Protocols*, pages 237–245, Austin, Texas, October 1998.

[21] John C. Lin and Sanjoy Paul. RMTP: A Reliable Multicast Transport Protocol. In *Proceedings IEEE Infocom '96*, pages 1414–1424, San Francisco, CA, March 1996.

[22] Ching-Gung Liu, Deborah Estrin, Scott Shenker, and Lixia Zhang. Local error recovery in SRM: Comparison of two approaches. Technical Report USC-CS-97-648, University of Southern California, 1997.

[23] Steven McCanne. Scalable multimedia communication with Internet multicast, lightweight sessions, and the MBone. Technical Report CSD-98-1002, University of California, Berkeley, CA, March 1998.

[24] Steven McCanne and Sally Floyd. *The LBNL/UCB Network Simulator*. Lawrence Berkeley Laboratory, University of California, Berkeley. Software on-line[1].

[25] Steven McCanne and Van Jacobson. *VIC: video conference*. Lawrence Berkeley Laboratory and University of California, Berkeley. Software on-line[2].

[26] Steven McCanne, Van Jacobson, and Martin Vetterli. Receiver-driven layered multicast. In *Proceedings of SIGCOMM '96*, pages 117–130, Stanford, CA, August 1996. ACM.

[1] http://www-nrg.ee.lbl.gov/ns/

[2] ftp://ftp.ee.lbl.gov/conferencing/vic

[27] Scott Michel, Khoi Nguyen, Adam Rosenstein, Lixia Zhang, Sally Floyd, and Van Jacobson. Adaptive web caching: towards a new global caching architecture. In *Computer Networks and ISDN Systems*, November 1998.

[28] Christos Papadopoulos, Guru Parulkar, and George Varghese. An error control scheme for large-scale multicast applications. In *Proceedings IEEE Infocom '98*, San Francisco, CA, March 1998.

[29] Vern Paxson. End-to-end routing behavior in the Internet. In *Proceedings of SIGCOMM '96*, pages 25–38, Stanford, CA, August 1996. ACM.

[30] Vern Paxson. Automatic packet trace analysis of TCP implementations. In *Proceedings of SIGCOMM '97*, pages 167–179, Cannes, France, September 1997. ACM.

[31] Vern Paxson. *Measurements and Analysis of End-to-End Internet Dynamics.* PhD thesis, Lawrence Berkeley Laboratory, 1997.

[32] Suchitra Raman and Steven McCanne. A model, analysis, and protocol framework for soft state-based communication. In *Proceedings of SIGCOMM '99*, Cambridge, MA, September 1999. ACM.

[33] Suchitra Raman, Steven McCanne, and Scott Shenker. Asymptotic scaling behavior of global recovery in SRM. In *Proceedings of SIGMETRICS '98/PERFORMANCE '98 Joint International Conference on Measurement and Modeling of Computer Systems*, Madison, WI, June 1998.

[34] Sylvia Ratnasamy and Steven McCanne. Inference of multicast routing trees and bottleneck bandwidths using end-to-end measurements. In *Proceedings IEEE Infocom '99*, New York, NY, March 1999.

[35] A. Rosenstein, J. Li, and S. Y. Tong. MASH: The Multicasting Archie Server Hierarchy. In *Computer Communication Review.* ACM, July 1997.

[36] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4), November 1984.

[37] Eve M. Schooler. A multicast user directory service for synchronous rendezvous. Computer Science department, California Institute of Technology, September 1996.

[38] Angela Schuett, Suchitra Raman, Yatin Chawathe, Steven McCanne, and Randy Katz. A soft-state protocol for accessing multimedia archives. In *Proceedings of the Eight International Workshop on Network and OS Support for Digital Audio and Video*, Cambridge, UK, July 1998. ACM.

[39] Henning Schulzrinne, Steve Casner, Ron Frederick, and Van Jacobson. *RTP: A Transport Protocol for Real-Time Applications.* Internet Engineering Task Force, Audio-Video Transport Working Group, January 1996. RFC-1889.

[40] Thierry Turletti. The INRIA videoconferencing system (IVS). *ConneXions*, 8(10):20–24, October 1994.

[41] X. Rex Xu, Andrew C. Myers, Hui Zhang, and Raj Yavatkar. Reliable multicast support for continuous-media applications. In *Proceedings of the Seventh International Workshop on Network and OS Support for Digital Audio and Video*, St. Louis, MO, May 1997. ACM.

[42] Maya Yajnik, Kim Kurose, and Don Towsley. Packet loss correlation in the MBone multicast network. In *Proceedings IEEE Global Internet '96*, London, England, November 1996.

[43] Xiaowei Yang and Li wei Lehman. Inferring characteristics of multicast trees, Dec 1998. MIT 6.896 Topics in Computer Networks term project and paper.

[44] R. Yavatkar, J. Griffioen, and M. Sudan. A reliable dissemination protocol for interactive collaborative applications. In *Proceedings of ACM Multimedia '95*, San Francisco, CA, November 1995. ACM.