

Improving the Fault Resilience of Overlay Multicast for Media Streaming

Guang Tan, Stephen A. Jarvis and Daniel P. Spooner
Department of Computer Science, University of Warwick,
Coventry, CV4 7AL, United Kingdom
{gtan,saj,dps}@dcs.warwick.ac.uk

Abstract

This paper addresses the problem of fault resilience of overlay-based live media streaming from two aspects: (1) how to construct a stable multicast tree that minimizes the negative impact of frequent member departures on existing overlay, and (2) how to efficiently recover from packet errors caused by end-system or network failures.

In particular, this paper makes two contributions: (1) A distributed Reliability-Oriented Switching Tree (ROST) algorithm that minimizes the failure correlation among tree nodes. By exploiting both bandwidth and time properties, the algorithm constructs a more reliable multicast tree than existing algorithms that solely minimize tree depth, while not compromising the quality of the tree in terms of service delay and incurring only a small protocol overhead; (2) A simple Cooperative Error Recovery (CER) protocol that helps recover from packet errors efficiently. Recognizing that a single recovery source is usually incapable of providing timely delivery of the lost data, the protocol recovers from data outages using the residual bandwidths from multiple sources, which are identified using a minimum-loss-correlation algorithm. Extensive simulations are conducted to demonstrate the effectiveness of the proposed schemes.

1 Introduction

Overlay multicast [4] has emerged as an effective technique to provide large-scale data dissemination over networks. While it is the case that shifting multicast functionality from routers to end hosts brings greater flexibility, the transient nature of the end hosts introduces problems of service reliability – in an overlay multicast tree, the (unannounced) departure of a member may result in data outages on all its downstream members. This paper considers this issue in the context of live media streaming, where the data is streamed from a single source to a large number of clients over a data delivery tree. Such an application has several characteristics that differentiate it from other applications (e.g., file transfers) and hence calls for special considera-

tions in system design: (1) It is bandwidth-intensive and yet the available bandwidth resources possessed by a multicast group may be far from rich. Each member has an out-degree (the number of immediate downstream nodes) constraint, and there may exist a large proportion of *free-riders* (i.e., zero-out-degree members) in the network [10] [13] [15]; (2) Multicast members exhibit a significant amount of heterogeneity in bandwidths [11] [13], and as a result the multicast tree nodes have a wide range of out degrees, which implies that the tree shape can be vastly different under various overlay construction methods; (3) Multimedia streaming does not require perfect reliability, and the packet error recovery can be performed in a best-effort manner.

Based on these observations, this paper proposes two techniques to enhance the fault resilience of live media streaming: the *Reliability-Oriented Switching Tree* (ROST) algorithm and the *Cooperative Error Recovery* (CER) protocol. The ROST algorithm is a proactive component in which the tree is adjusted toward a structure that minimizes failure correlation among tree nodes¹ – that is, the failure of a node will affect as few downstream nodes as possible. For it to be reliable, it is generally believed that the tree should be as short (and hence wide) as possible [9] [14] [12] [5] [3], under the constraint that no network congestion occurs near the nodes. While this is effective, we show that it is not optimal. Inspired by the use of member’s long-tailed lifetime distribution in tree construction [12], this paper proposes to combine the members’ bandwidth and time properties as a new criterion to adjust the tree. We define a metric called *bandwidth-time product* (BTP) as the product of one member’s outbound bandwidth² and its age, and move the nodes with large BTPs gradually up the tree in a distributed manner. Simulation results under realistic experimental settings show that ROST (1) reduces the average number of streaming disruptions per member by 36-57% compared to a centralized depth-optimal approach; (2) achieves the smallest

¹In this paper, every node in the overlay is a member of a multicast group, so we will use the term node and member interchangeably.

²The outbound bandwidth is the maximum outgoing bandwidth provided by the access link. For simplicity, it is also referred to as bandwidth.

end-to-end service delay (or tree depth) among three representative distributed algorithms, and only incurs a small increase in service delay of 10-15% compared to the centralized depth-optimal approach; and (3) introduces a very low protocol overhead.

The *Cooperative Error Recovery* protocol is a reactive mechanism that recovers from streaming disruptions incurred by the failures of upstream nodes. When a non-leaf overlay node fails, the affected nodes need to rejoin the tree, which involves failure detection and parent re-finding periods and usually lasts in the order of tens of seconds [4]. During these periods, the affected nodes must retrieve the lost data from other normal nodes before the receiving buffer is exhausted. Many techniques have been proposed to identify recovery nodes and request data from them [17] [2] [18] [16] [6]. However, they are all based on a single-source-based recovery mechanism. We propose to use multiple recovery nodes, which are identified using a minimum-loss-correlation algorithm, in order to recover from node failures. Our experiments demonstrate that substantial improvements can be achieved using this scheme.

This paper focuses on the single-tree based data delivery paradigm. Although there exist multiple-tree based approaches that improve fault-resilience by leveraging some specialized media encodings (e.g. multiple description coding [9]), using a single-tree provides a more general approach and we believe that the techniques developed under this scheme can also be applied to the multiple-tree case.

The remainder of this paper is organized as follows. The next section reviews related work in both reliable overlay construction and packet error recovery; Section 3 describes the ROST algorithm in detail; Section 4 presents the CER protocol; Section 5 introduces the simulation methodology; Section 6 analyzes the simulation results and Section 7 concludes the paper.

2 Related Work

2.1 Construction of Overlay Multicast Trees

Some earlier work on overlay construction for large-scale single-source multicast include NICE [1] and ZIGZAG [14]. However, these methods do not consider the out-degree limits of multicast nodes and thus are not suited to high bandwidth media streaming.

For media data multicast, most algorithms try to build a *fault resilient* overlay. An important approach to achieving this is to build a short tree. Intuitively, the shortness helps to reduce the number of descendant nodes that will be affected by a failed node. An additional merit of this approach is that it generally leads to a small average service delay from the source. The *minimum depth algorithm* [5] [9] [12] is an example of this approach. It searches from the tree root downward to the leaf layer to identify a parent with spare bandwidth capacity for a new node to join. If there are

multiple choices, the nearest parent (in terms of network delay) is chosen. A variant of this algorithm [8] first selects a number of members randomly from the overlay, and then performs the minimum depth algorithm. Borrowing the idea of “fat-trees” in parallel architectures, Birrer et al. propose to build a fat tree [3] with similar characteristics to the short and wide tree. The *high-bandwidth-first algorithm* [5] achieves minimum tree depth by placing the nodes from high to low layers in a non-increasing order of bandwidths; that is, nodes do not have more bandwidth capacity than any node higher up in the tree. This algorithm achieves a global optimization. However, it imposes very high protocol overhead and is therefore not practical for real implementation.

In contrast to the depth-optimizing approach using the members’ bandwidth properties, Sripanidkulchai et al. propose another approach [12] which leverages the member’s time property: if the members’ lifetimes follow a distribution with a long tail [15] [11], then the older members are less likely to leave before the younger ones. This idea leads to the design of the *longest-first algorithm* [12], which selects the longest-lived member among those with spare bandwidth capacities as the new member’s parent. This algorithm, however, turns out to yield poor performance since it results in a tall tree.

2.2 Packet Error Recovery for Overlay Multicast

STORM [17] is a resilient multicast protocol for continuous-media applications, in which the media data is delivered using network-layer multicast, while the error recovery mechanism is built on an overlay. Each receiver maintains a list of recovery parents which provide the loss repair service. The idea of using multiple recovery parents is similar to our CER protocol. However, the selection of recovery parents and the recovery procedure are both different from our scheme. Lateral Error Recovery (LER) [16] aims to provide fast recovery for overlay multicast. In LER, all subtrees immediately under the root node (called *planes* in LER) are organized in a way such that a node in a subtree has a small network latency from its recovery nodes in other subtrees. Since failure correlation of these subtrees is small, the error recovery can be performed in a fast and reliable manner. This tree construction method may however result in large network stretch under the out-degree constraints.

Probabilistic Resilient Multicast (PRM) [2] is a multicast data recovery scheme that uses a technique called *randomized forwarding*. The randomized forwarding adds some random cross-tree edges on the overlay, so packets losses can be repaired in a proactive manner. PRM handles node failures by raising the forwarding probability of some recovery nodes to one. This is equivalent to using all the residual bandwidth of one recovery node. Cooperative Patching [6] uses a list of recovery nodes for each receiver when

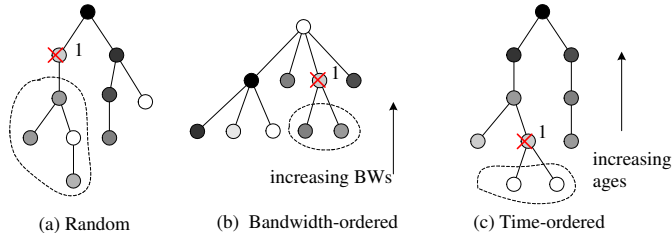


Figure 1: Examples of the three types of tree. Darker colors represent older nodes. The dashed lines represent the regions affected by a node failure.

recovering from parent failures. This technique focuses on the selection of recovery nodes.

LER, PRM and Cooperative Patching all use a single source to recover from upstream node failure, which differs from our work.

3 The Reliability-Oriented Switching Tree (ROST) Algorithm

3.1 Analysis of Existing Algorithms

As introduced in Section 2, the reliability of an overlay multicast tree can be optimized in two ways:

- *Depth-optimizing*: the tree is optimized in depth. Two representative algorithms are the minimum-depth algorithm and the high-bandwidth-first algorithm, of which the latter is the extreme case of the former. Since the high-bandwidth-first algorithm places the nodes in order of bandwidth, the constructed tree is called a *bandwidth-ordered (BO) tree*. An example of a BO tree is shown in Fig. 1(b).
- *Time-optimizing*: the nodes are placed in the tree according to their time properties (ages). A representative algorithm is the longest-first algorithm. An extreme case of this approach is a *time ordered (TO) tree*, in which the nodes are placed in a strict order of ages, that is, nodes are not older than any node found at higher levels in the tree. Fig. 1 (c) gives an example of this kind of tree.

The average number of nodes affected by a failure in the BO tree is small since the resulting tree is short. For example, in Fig. 1, a failed node 1 in the BO tree (see Fig. 1 (b)) causes less streaming disruptions on its descendants than in a random tree (see Fig. 1 (a)). However, this type of tree needs frequent disconnections and reconnections between nodes to maintain such a bandwidth layout. For example, if some node a in a layer i leaves, then the node, say b , with maximum bandwidth in layer $i+1$ should be moved to node a 's position, which further forces all of b 's children rejoin

the tree. This recursive rejoin imposes very high overheads on the multicast nodes.

The time-optimizing algorithm places the younger nodes under the older nodes, making use of the fact that under a long-tailed lifetime distribution, older nodes are more likely to stay longer in the network. The research in [12] has shown that this algorithm can give good prediction on the relative staleness of multicast members. However, a pure time-based algorithm like this will result in a tall tree [12], which greatly increases the failure correlation of the tree nodes and finally makes the tree more unstable.

In light of the benefits of time-ordering to tree reliability, a natural question is: Is it possible to incorporate this element into the bandwidth-based algorithm so that the reliability of the depth-optimal tree can be further improved? The answer to this question, of course, ultimately depends on how great the power of time ordering is. Besides which, a major challenge is to ensure that the tree depth does not significantly deviate from the optimal value as produced by the BO tree, so that the benefits from the time ordering are not cancelled out, or even exceeded, by the negative effect of the increased tree depth, as in the longest-first algorithm. In addition to the primary goal concerning the reliability and tree depth, some other desired properties of the expected approach include: (1) A small protocol overhead. In contrast to the high overhead of the BO algorithm, the new method should not impose heavy burden of parent re-finding on individual multicast members; (2) A distributed implementation. For large-scale overlay networks in which the nodes may arrive in flash crowds, centralized tree construction approaches like the BO algorithm or the algorithm used in [9] are generally limited in scalability; (3) A scheme for protecting against cheating/malicious behavior. Most of the previous approaches relying on information such as bandwidth or time do not consider the possibility of cheating behavior, and thus are potentially vulnerable to malicious attacks. The new scheme should prevent this in an effective way; and (4) Being simple to implement. Introducing extra switching operations in addition to basic multicast tree construction/fixing have been studied in previous work [7]. However, these methods generally require complex node coordinations. The new approach should consider this practical issue.

3.2 Basic Idea of ROST

The ROST algorithm uses a simple switching tree technique to optimize the overlay. The criterion guiding the switching operation is a metric called *Bandwidth-Time Product (BTP)*, which is defined as the product of a node's outbound bandwidth and its age. The basic idea of the algorithm is to move nodes with large BTPs higher in the tree so that better service quality (less stream disruptions and smaller service delay) can be offered to these nodes. Since

either a large bandwidth or a long service time helps to increase BTP, a node can be encouraged to contribute more bandwidth resource or longer service time as a trade for service quality. From the user’s perspective, this forms an incentive mechanism that helps increase overall system resources.

3.3 Key Operations

ROST is performed in a completely distributed manner. It includes three basic operations: Join, Leaving and BTP-based Switching.

Member Joining and Leaving When a new member joins the network, ROST assumes that there is a bootstrap mechanism that provides at least one active member in the group. The new member then queries the existing members for information about other participants until it obtains a certain number (say, 100) of known members or the procedure exceeds some time limit. It then sends a JOIN request to these members, who will respond with an ACCEPT message if they have spare bandwidths. If there are more than one possible parents, the new member chooses the one with the smallest tree depth as its parent (each member knows its own layer number in the tree). If multiple such parents exist at the same layer, it chooses the nearest parent in terms of network delay. When a member leaves, it may give notification to its neighbors or it may just leave abruptly. In either case, the children of the leaving node have to rejoin the tree by contacting other members.

BTP-based Switching The multicast source is pre-assigned an infinite BTP, and always remains at the top of the tree. When a new member initially enters the network, its BTP is 0. In most cases, the high layers of the tree are occupied and the new member becomes a low-layer node. As time goes on, a node’s BTP increases at a rate proportional to its bandwidth. If its bandwidth is larger than its parent, then there must be some time point in the future when its BTP exceeds its parent (if the parent does not leave before itself). At that time the algorithm will exchange the roles of these two nodes. Fig. 2 gives an example of this operation.

In Fig. 2 (a), node *a*’s BTP is 10 and has an out-degree of 2; node *b* has a BTP of 12 and an out-degree of 3. Node *b* is therefore moved up to become the parent and node *a* is moved down to become the child. Now that node *a* can support only two of the three nodes *d*, *e*, *f*, one child must be assigned a new parent. The algorithm chooses *f*, the node with the largest BTP and reconnects to node *b*, which now has a spare out-degree.

The switching is performed autonomously by all members. For every interval of a certain time (called a *switching interval*), a member compares its own BTP with its parent’s current BTP. If its BTP exceeds that of its parent, and its

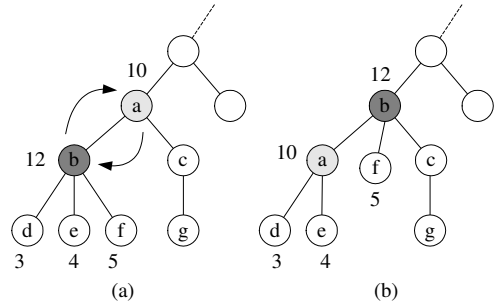


Figure 2: Illustration of the BTP-based switching operation. (a) Before switching; (b) After switching. The numbers represent the nodes’ BTPs.

bandwidth is no less than the parent’s bandwidth, then the switching operation is triggered. The bandwidth comparing avoids unnecessary switching since if the child has a smaller bandwidth, the BTP will eventually be exceeded by the parent, and it will ultimately be placed below the parent.

When a node decides to switch with its parent, it first tries to “lock” a set of relevant nodes, including its parent, its grandparent and all of its children and siblings, in order to maintain a consistent state of the nodes. If any of these nodes is already in the process of another switching, or operations such as overlay failure recovery, the lock cannot be acquired and the initiating node waits for a certain amount of time (say, 15 seconds) before it tries to check the switching condition and lock again. It can be seen that a switch operation involves an average overhead of $2d + 1$ in terms of the number of parent changes, where d is the average node out degree. By choosing a relative large switching interval (e.g., 15 minutes) this overhead can be made very small while preserving the advantages of this method, as will be demonstrated in the experiments.

The algorithm moves nodes up the tree in a gradual manner. This potentially prevents short-lived clients (which account for a nontrivial fraction of clients [15] [13]) from climbing up the tree upon joining, which may bring stream disruptions to many downstream nodes. In contrast, placing a new member at the leaf layer first and then adjusting its position according to its behavior can reduce this risk. The longer a node stays in the network, the safer it is to be moved up the tree.

3.4 Preventing Cheating or Malicious Behavior

Truth telling is critical for ROST. Without a mechanism to enforce this, a node can simply report that it has a large bandwidth or has stayed in the overlay for a long time in order to have itself gradually moved up toward the root of the tree. As a result, the ROST approach would benefit nodes that cheat, instead of providing incentives for nodes to contribute to the system. Worse still, a malicious node may easily attack the system by moving to a place near the root and then disrupting the streaming to most tree nodes.

ROST addresses these problems using a *reference node* mechanism. Using this mechanism, a node can verify the BTP of another node by inquiring of some nodes called *referees*. Each node is associated with two types of referees: age referees and bandwidth referees. When a node initially joins the overlay, its parent records its joining time to $r_{age} > 1$ randomly chosen nodes, called age referees, which then establish heartbeat connections with the new node and act as its age witnesses. When a node needs to show its age information, it simply tells others about the addresses of its age referees, which can then be consulted for the trustable age information. Note that a node’s age referees cannot be designated by itself, in order to prevent possible collusion; while the parent can do this because it has no incentive to collude with a child which is a potential competitor for its own tree position.

To ensure reliable bandwidth information, a newly arriving node also obtains two sets of nodes from parent: a *bandwidth measurer set* and a *bandwidth referee set*. The former is a set of nodes with enough spare incoming bandwidths that can be used to measure the new node’s effective outgoing bandwidth. To do this, the new node concurrently transmits testing data to these nodes, who can measure the partial bandwidths and jointly form an aggregated bandwidth measure on the parent. The parent then saves this value to the $r_{bw} > 1$ bandwidth referees. Later on, when the bandwidth information is needed by others, the node only need to provide the addresses of the bandwidth referees.

Both r_{age} and r_{bw} are greater than 1 for the purpose of fault tolerance. When a node discovers that a referee leaves or breaks down, it asks its parent to assign a new referee, which then synchronizes with the existing active referees. Note that in an asynchronous environment like the Internet, the age information maintained by the multiple referees need not be strictly consistent, since the difference is upper bounded by a heartbeat interval, which is small compared to the age of a node.

The above mechanism adds extra complexity and overhead to the overlay network. However, it is important to recognize that the cheating and malicious access problems are not unique to the ROST algorithm, but rather, common to all forms of overlay network where centralized authority is unavailable and peers’ information can not be obtained by simple probing. Our design here not only serves as an important complement to the basic ROST algorithm, but also provides a solution to the same problem probably encountered in other overlay-based applications.

4 The Cooperative Error Recovery (CER) Protocol

Due to network congestion, transient or permanent intermediate failures of routing services, packet errors (mainly losses) are inevitable. To restore the normal streaming, a

member needs to rejoin the tree. This process involves failure detection, contacting with multiple nodes (e.g., 50) to select an appropriate parent, and potentially some waiting time when concurrent join requests compete on some parent. Taking into account all these factors, this process can take a time in the order of tens of seconds [4]. Yang et al. [18] have proposed a proactive approach to expedite this process by computing a rescue scheme in advance. However, in a large-scale Internet-based system which is dynamic in nature, this still remains as a general problem.

To address this problem, we propose to use multiple cooperative error recovery sources, called a *minimum-loss-correlation group* (MLC group), to help a node that suffers from a stream disruption find the lost data while it is looking for a new parent. As the name suggests, the MLC group has the property that a node failure or packet missing on one node is unlikely to affect other nodes within the same group. This is important because in a tree structure with a high degree of flow dependence, a packet loss occurring on a certain node may affect many downstream nodes. By carefully choosing a set of nodes with small loss correlation, the loss recovery can be performed more efficiently.

4.1 Minimum-Loss-Correlation Recovery Group

Loss correlation is caused by the common network path shared by two multicast members. The failure of any entity on the shared path, including the underlying physical links and intermediate overlay nodes, could result in streaming disruptions on both of these members. However, it is difficult for an overlay protocol to identify the shared physical links between two general overlay nodes. Also because the failure probability of physical links is much lower than that of overlay members, we only consider the loss correlation at the overlay level.

We assume a tree $T = (V, E)$, where V and E are the sets of all nodes and edges respectively, and define the loss correlation function $w : V \times V \rightarrow I$, where I is the set of non-negative integers, and $w(v_1, v_2)$ represents the number of common edges between the tree paths from the root r to v_1 and v_2 . The MLC group problem is thus to find a set of nodes K such that $\sum_{v_i, v_j \in K} w(v_i, v_j)$ is minimum. We solve this problem in two steps. First, a node constructs a partial tree using the information of other nodes maintained by itself. Recall that during the multicast process, nodes periodically exchange neighbor information with each other, so each node will know about a medium-sized (e.g., 100) subset of other nodes. The information of each node includes its own address, the addresses, layer numbers and out degrees of all its ancestors. An example of a node using these information to construct a tree T is shown in Fig. 3.

The second step is to find the desired MLC group. We assume that the i -th level of the tree T is a node set L_i , with L_0 consisting of only the root node r . Each node has a children set $C_i = \{c_{i0}, c_{i1}, \dots\}$, and a descendants set

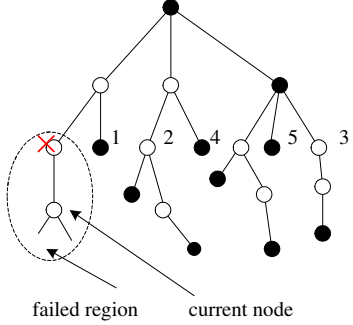


Figure 3: Illustration of finding an MLC group. The solid circles represent the external nodes known by current node.

$D_i = \{d_{i0}, d_{i1}, \dots\}$. The proposed algorithm first identifies a set G' of root nodes of K subtrees, from which the members of the MLC group G can be derived. The following steps describe the algorithm.

Algorithm 1 Finding the minimum-loss-correlation recovery group

- 1: Initialize the MLC root set $G' \leftarrow \emptyset$ and MLC group set $G \leftarrow \emptyset$
 - 2: Starting from L_0 , find the first level L_i in T such that $|L_i| < K \leq |L_{i+1}|$;
 - 3: For each $v_i \in L_i$, randomly pick a child of v_i , say, c_{ij} , let $G' \leftarrow G' + \{c_{ij}\}$ and $C_i \leftarrow C_i - \{c_{ij}\}$, until $|G'| \geq K$.
 - 4: For each $v_i \in G'$, randomly pick a descendant $d_{ij} \in D_i$, and let $G \leftarrow G + \{d_{ij}\}$ and $D_i \leftarrow D_i - \{d_{ij}\}$.
-

The number sequence 1, 2, \dots , 5 in Fig. 3 gives an example of the process of identifying a MLC group G' . G can then be derived from G' by randomly selecting descendants from the subtrees rooted at all $v_i \in G'$. The randomized selection is used for the purpose of load balancing and for also providing alternatives for the isolated nodes in search for the nearest recovery nodes.

4.2 The Loss Recovery Protocol

Explicit loss notification For each packet in the stream, there is a delivery deadline and playback deadline for a specific member. The playback deadline is the delivery deadline plus the application's buffering time. Any packet missing the playback deadline is meaningless. When a member detects a delivery deadline missing, it regards this as a packet loss, and may try to re-find it within the playback deadline. First, it needs to determine whether or not the packet loss or streaming disruption is due to its parent so as to avoid duplicate error recovery or unnecessary rejoins. An approach called *Explicit Loss Notification* (ELN) is used to address this problem. In this approach, each multicast member, upon detecting a packet loss, sends a notification

packet containing only the missed sequence number to its children, who then infer that the packet loss does not originate from their parent. The notification packet is further propagated downstream so that all the descendant nodes can count on the upstream recovery. If a member continuously detects large gaps (e.g., sequence gap > 3) between the sequence of both normal data and ELN packets, there must be a parent failure or link congestion/failure occurring and this member simply launches the rejoin process. Note that a ELN packet only contains a sequence number (or a series of sequence numbers when necessary) and hence will bring negligible extra overhead on the network compared to the normal streaming traffic.

Repairing the lost data A member places the nodes of its recovery group in order of network distance. Upon detecting a packet loss, it sends a packet repair request to the first recovery node. The request also contains a list of other recovery members. The first recovery node searches its buffer or waits a certain time for the requested packet to arrive. If found or received, the requested packet is sent back to the requesting node, otherwise the first recovery node sends back a negative acknowledgement (NACK) packet and at the same time, it forwards the request to the second recovery node, which then performs the same repair operation. This process continues until the requested packet is discovered or all recovery nodes are contacted. All repaired packets are sent back to the intermediate nodes in addition to the original requesting node.

If a member detects a parent failure, it still sends a loss repair request to the first recovery group node, asking for a recovery at a full streaming rate. If the first node has only a residual bandwidth of $\epsilon_1 < 1$ (here we assume the full stream rate is 1), it takes responsibility for sending all packets that satisfy: $(n \bmod 100) < 100\epsilon_1$, where n is the sequence number. The first recovery node then passes the request on to the second recovery node, which has a residual bandwidth of ϵ_2 and then takes care of repairing packets whose sequence numbers satisfy $100\epsilon_1 \leq (n \bmod 100) < 100(\epsilon_1 + \epsilon_2)$. The process continues until the sum of all residual bandwidths of the examined recovery nodes is no less than 1, or all recovery nodes have been contacted.

5 Simulation Setup

An event-driven simulator has been developed to study the performance of different algorithms. The GT-ITM transit-stub model [19] is used to generate an underlying network topology consisting of 15600 nodes. Link delays between two transit nodes, transit nodes and stub nodes, and two stub nodes are chosen uniformly between [15, 25] ms, [5, 9] ms and [2, 4] ms, respectively. Of all the 15360 stub nodes, a fraction of them are randomly selected to partici-

pate in the multicast tree. The server’s location is fixed at a randomly chosen stub node.

In all simulations, the media streaming rate is assumed to be 1. The root node has a bandwidth of 100, resembling the capability of a powerful source server. Other nodes’ outbound bandwidths follow a Bounded Pareto distribution [11] [12] [10], whose shape, lower bound and upper bound parameters are set to 1.2, 0.5 and 100, respectively. This means that 55.5% of the members are effectively “free-riders”, and a small number of “super-nodes” exist with out-degrees larger than 20. The nodes’ lifetimes follow a log-normal distribution with the location and shape parameters set to 5.5 and 2.0, respectively, which are chosen according to the statistical findings in [15]³. According to the *Little’s Law*, the node arrival rate λ is determined from dividing M by the mean value of lifetime, i.e. 1809 seconds. For the ROST algorithm, the default switching interval is 360 seconds.

The following five algorithms are implemented: (1) Minimum-depth algorithm as introduced in Section 2. A new member always chooses a parent highest in the tree from up to 100 nodes in the network. (2) Longest-first algorithm as introduced in Section 2. A new member always chooses the oldest parent from up to 100 nodes in the network. (3) and (4) *Relaxed bandwidth-ordered algorithm* and *relaxed time-ordered algorithm*. These are two variants of the BO and TO algorithms as introduced in Section 2. Since the (strict) BO and TO trees have unacceptably high protocol overheads in terms of average number of reconnections for a single node during its lifetime, a modification is made to make the compared scenarios more realistic: when a member joins/rejoins the tree, it always searches from the high to low layers to see if there is a smaller-bandwidth or younger node, and if so, the located node is replaced with the new one. The evicted node, and possibly together with some of its children in the case of time ordering, are forced to rejoin the tree. This results in bandwidth/time ordering among parents and children, but a node may have smaller bandwidth/age than another non-child node in the next layer. Since these two variants still follow the basic ideas of bandwidth/time ordering, they are used for performance comparisons. Note that both algorithms assume a central administrator providing global topological information. (5) ROST algorithm, as introduced in Section 3.

6 Performance Evaluation

This section first compares the ROST algorithm against the other four tree construction algorithms from the point of view of different performance criterion, and then studies

³Ref. [13]) suggests using a Pareto distribution to model the tail distribution of node lifetime. Since both lognormal and Pareto distributions share a similar characteristic in the tail: that longer lived peers have longer expected residual lifetimes, we only consider the lognormal model here.

the performance of the CER protocol.

Comparison of Tree Reliability Service reliability is measured by the average number of streaming disruptions experienced by a single node during its lifetime in the steady state of a multicast tree. The experiments consider the extreme case in which every node departs abruptly without notification to others, and hence results in a disruption on each of its descendants. This metric reflects the stability of a tree in the most uncooperative and dynamic environment. Fig. 4 compares the performance of the five algorithms under different sizes of networks.

It can be seen that the minimum-depth algorithm and the longest-first algorithm perform the worst, because they either are completely reliability-ignorant or operate very conservatively in ordering the nodes in ages. The relaxed BO algorithm has substantially improved reliability over these two methods because of the reduced tree depth. Yet it is defeated by the relaxed TO algorithm in all cases, which has better reliability owing to the time ordering.

The ROST algorithm appears to be a scheme yielding the best result. Compared to the relaxed BO algorithm, the number of disruptions has been reduced by 36-57%; and compared to the relaxed TO algorithm, which is the second best algorithm, the reduction is up to 40% in certain cases. This shows how the ROST algorithm outperforms both the BO and TO algorithms by combining their strengths. A further observation is that the average service disruptions per node is much less sensitive to the network size as compared with the minimum-depth, longest-first and relaxed TO algorithms, which do not consider bandwidth ordering and therefore vary more significantly in tree depth.

Fig. 5 provides more distribution information about the reliability of an 8000-node network. Fig. 6 shows the accumulative number of stream disruptions experienced by a typical member under the difference algorithms. This member has a moderate bandwidth and a long lifetime in order to observe the network over a long period. It joins the overlay after the network enters a steady state. One can see that with the ROST algorithm, the frequency of stream disruptions (i.e., the slope of the line) becomes smaller as the member ages, which reflects how ROST benefits the long-lived members. Although with only an average bandwidth, the member gradually ascends the tree and becomes less and less frequently interrupted by the dynamics of other members.

Service Delay and Network Stretch This set of experiments examine the quality of the tree produced by the various algorithms in terms of end-to-end service delay and network stretch. The *average service delay* measures the average of all nodes’ end-to-end service delays along the overlay paths. The *average stretch* is the average of all nodes’

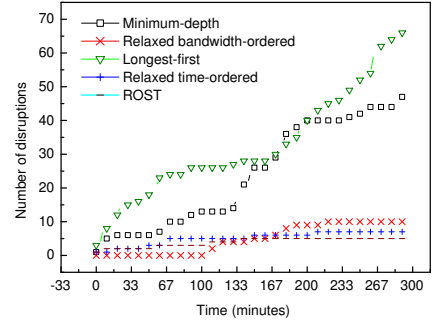
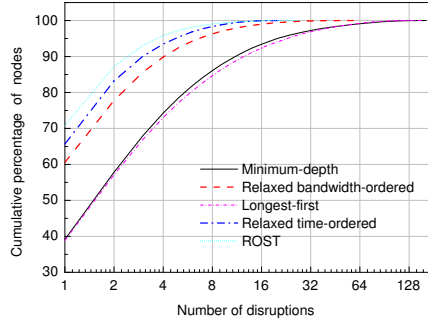
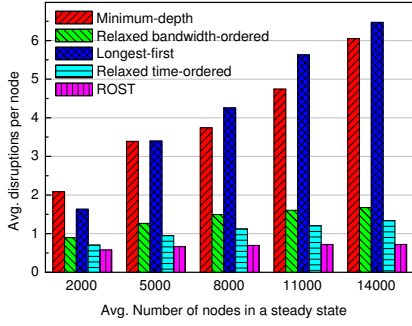


Figure 4: Avg. number of disruptions per node. Figure 5: CDF of avg. number of disruptions. Figure 6: Accumulative number of disruptions of a typical member over time.

stretches, which is defined as the ratio of one nodes' service delay to the delay along the direct unicast path in the underlying network [4] [1].

Fig. 7 shows that the ROST algorithm achieves the best result in terms of both metrics among the three distributed algorithms (the other two are the minimum-depth algorithm and the longest-first algorithm). This reflects how bandwidth ordering in ROST benefits the tree depth.

Compared with the relaxed BO tree, the ROST algorithm has a small increase in the two defined metrics of 10-15%. This is because the ROST algorithm optimizes the layout in a more confined space (only along the child-parent paths regardless of the bandwidth order between siblings), and hence yields a more sub-optimal bandwidth layout. However it should be pointed out that the best performance of the relaxed BO algorithm relies on a centralized controller owning the global topological information, which makes it impractical for large-scale networked systems.

Fig. 8 shows the average stretch of nodes under various network sizes, which agree with the observations from Fig. 7. Fig. 9 shows the service delay of a typical member with the same property as assumed in the experiments with Fig. 6. It can be seen that under the ROST and relaxed TO algorithms, the examined member's delay becomes smaller as time progresses, implying a higher and higher position in the multicast tree. In contrast, the delay fluctuates with no convergence with the other three algorithms which do not consider time ordering.

Comparison of Protocol Cost Both bandwidth ordering and time ordering require reconnections between nodes to optimize the structure of the tree, thus introducing a protocol overhead. This overhead is measured in the average number of reconnections brought by the optimizing mechanism on a single node during its lifetime. Fig. 10 compares the protocol overheads of the five algorithms. Note that the minimum-depth algorithm and the longest-first algorithm do not impose any protocol overheads at all.

The results show that the ROST algorithm performs best among the three algorithms that do incur protocol overheads. Besides which, the the ROST algorithm requires far less than one reconnection for a single node during its lifetime. This indicates that ROST is very efficient in general. Recall that the average node lifetime is 1809 seconds and the default switching interval is 360 seconds. These translate to 5 switches per node, which is clearly larger than the measured overhead. The reason behind this is that a switching interval does not necessarily correspond to an actual switching operation; rather, it only provides a possible opportunity for switching. In an overlay that has evolved for a long time, many high-bandwidth or long-lived nodes have already occupied the high positions in the tree, so most nodes have been left fewer chances to climb up the tree.

Effects of Switching Interval Fig. 11 shows the impact of various switching intervals on the performance of an 8000-node system. As expected, a smaller interval provides more adjusting opportunities for the overlay and thus the streaming reliability is higher. Because of the implicit bandwidth ordering, a small interval also leads to a small average service delay and network stretch. These benefits, however, come at the expense of an increase of protocol overhead, as shown in the bottom-right sub-figure in Fig. 11. Also note that the protocol overhead is fairly small (0.15 reconnections per node) even when the interval takes the smallest value (i.e., 480 seconds).

Effects of Recovery Group Size This section examines the effect of different recovery group sizes on the user-perceived quality of service and the requirement on the user buffer through packet-level simulation. The data is propagated from the tree root at a constant rate of 10 packets per second after the network enters a steady state. By default, each node has a playback buffer size of 5 seconds, or 50 packets, hence every lost packet must be repaired within 5

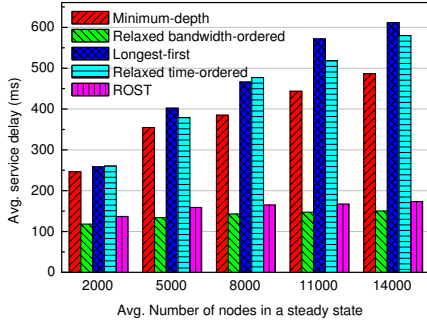


Figure 7: Avg. network delay vs. network size.

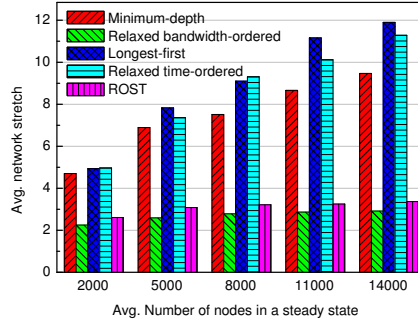


Figure 8: Avg. stretch vs. network size.

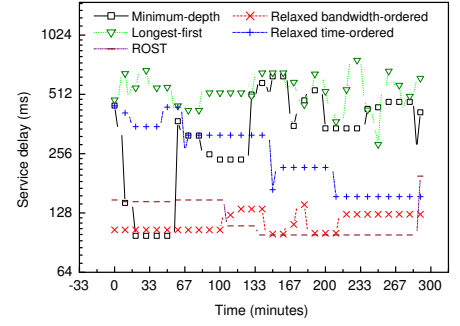


Figure 9: Service delay of a typical member over time.

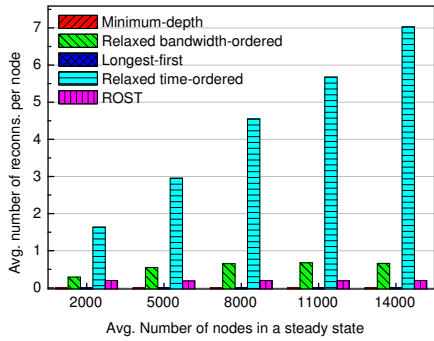


Figure 10: Comparison of protocol overheads.

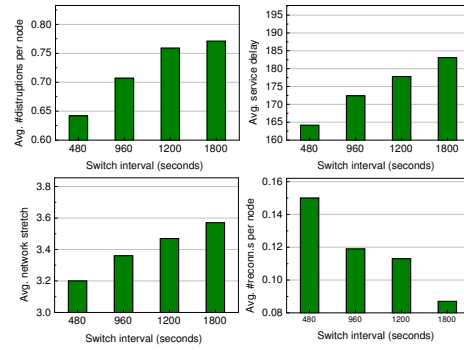


Figure 11: Effect of switching interval.

seconds. It is assumed that a member needs 5 seconds to detect a failure of its parent, and another 10 seconds to rejoin the tree; thus a failure recovery takes 15 seconds in total. We only consider packet losses incurred by node failures. A node’s residual bandwidth is uniformly distributed in 0-9 packets/second, and it only uses the residual bandwidth to help others in error recovery.

A metric called *starving time ratio*, defined as the ratio of the total streaming disruption time to the whole view time since the playback begins, is used to evaluate the quality of service perceived by a user under the workload assumed in Section 5. Fig. 12 presents the average starving time ratios of all multicast members for varying recovery group sizes. The tree is constructed using the minimum-depth algorithm. The result shows that, compared with the group size of 1, a small increase to a group size of 3 can reduce the average starving time by an order of magnitude ($< 0.2\%$ for all network sizes).

Fig. 13 depicts the relationship between the user’s buffer size and starving time ratio. Clearly, a larger buffer size can better accommodate streaming dynamics. However, a large buffer size also means a long startup delay, and hence worse quality of service in terms of interactivity. Again

we can see that a small increase in the recovery group size can dramatically reduce the required buffer size. For example, for the one-recovery-node case, the buffer size must be ≥ 27 seconds to make the average starving time ratio $\leq 0.55\%$, whereas for the two-recovery-node case, the buffer size needs only to be 5 seconds to meet the same requirement.

Evaluation of ROST+CER In this section, we compare ROST+CER against a general overlay multicast scheme, in which the tree is constructed using the minimum-depth algorithm, and the packet losses are recovered from a single source. We vary the recovery group size from 1 to 3, and examine the average starving time ratio under the two schemes. Fig. 14 gives the results with a 95% confidence interval. It can be observed that for each group size, the use of ROST+CER significantly reduces the average starving time ratio. On average, the ratio is reduced by 8-9 times. One can also see that, even with a recovery group size of 1, the ROST+CER scheme performs better than a Minimum-depth+Single source scheme with two recovery group members, which again reflects the effectiveness of ROST.

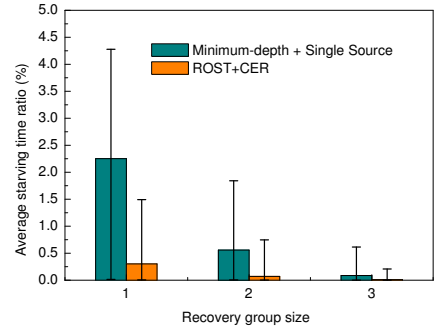
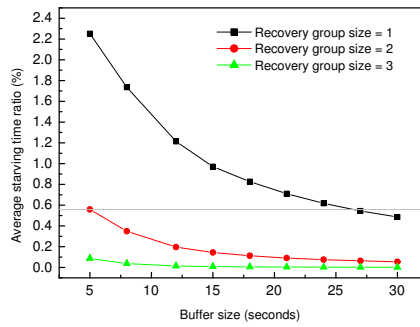
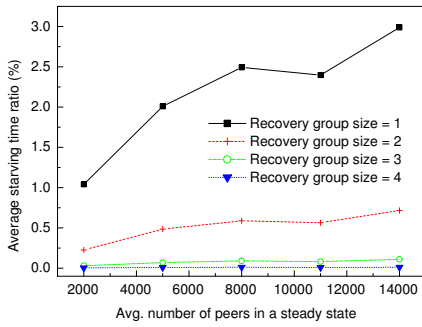


Figure 12: Avg. starving time ratio vs. group size.

Figure 13: Avg. starving time ratio vs. buffer size.

Figure 14: Evaluation of ROST+CER.

7 Conclusions

This paper addresses the fault resilience for overlay multicast using two techniques: (1) A proactive algorithm called ROST that minimizes the failure correlation among multicast tree nodes by gradually switching the tree toward a structure partially ordered in bandwidth and partially ordered in time; (2) A reactive component that recovers from streaming disruptions incurred by upstream member failures using a CER protocol. The experimental results demonstrate the superiority of the proposed schemes.

8 Acknowledgments

We are grateful to the anonymous reviewers for their excellent feedback. This research was sponsored in part by grants from the NASA AMES Research Center (administered by USARDSG, contract no. N68171-01-C-9012), the EPSRC (contract no. GR/R47424/01) and the EPSRC e-Science Core Programme (contract no. GR/S03058/01).

References

- [1] S. Banerjee, B. Bhattacharjee, C. Kommareddy. Scalable Application Layer Multicast. *ACM SIGCOMM 2002*.
- [2] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan. Resilient multicast using overlays. *ACM SIGMETRICS 2003*.
- [3] S. Birrer, D. Lu, F. E. Bustamante, Y. Qiao and P. Dinda. FatNemo: Building a Resilient Multi-Source Multicast Fat-Tree. In *Proc. of the Ninth International Workshop on Web Content Caching and Distribution (WCW)*, October 2004.
- [4] Y. Chu, S. Rao, and H. Zhang. A Case for End System Multicast. *Proc. of ACM SIGMETRICS*, June 2000.
- [5] M. Guo, M. Ammar. Scalable live video streaming to cooperative clients using time shifting and video patching. *Proc. of INFOCOM 2004*.
- [6] M. Guo, M. H. Ammar and E. W. Zegura. Cooperative Patching: A client based P2P architecture for supporting continuous live video streaming. *Proc. of the 13th International Conference on Computer Communications and Networks (ICCCN), 2004*. J. Jannotti,
- [7] D. Helder and S. Jamin. End-host Multicast Communication Using Switch-tree Protocols. In *Proc. of International Conference on Global and Peer-to-Peer Computing on Large Scale Distributed Systems, 2002*.
- [8] V. N. Padmanabhan, Helen J. Wang, Philip A. Chou. Resilient Peer-to-Peer Streaming. *Proc. 11th IEEE International Conference on Network Protocols (ICNP)*, 2003.
- [9] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai. Distributing Streaming Media Content Using Cooperative Networking. *ACM NOSSDAV*, May 2002.
- [10] S. Saroiu, P. Gummadi and S. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. *Proc. of Multimedia Computing and Networking (MMCN)*, 2002.
- [11] S. Sen and J. Wang. Analyzing peer-to-peer traffic across large networks. *IEEE/ACM Trans. on Networking*. Vol. 12, No. 2, April 2004.
- [12] K. Sripanidkulchai, A. Ganjam, B. Maggs and H. Zhang. The feasibility of supporting large-scale live streaming applications with dynamic application end-points. *Proc. of ACM SIGCOMM*, 2004, Portland, Oregon, USA.
- [13] K. Sripanidkulchai, B. Maggs and H. Zhang. An analysis of live streaming workloads on the Internet. *Proc. of the 4th ACM SIGCOMM IMC*, Oct., 2004. Italy.
- [14] D. A. Tran, K. A. Hua, and T. T. Do. A peer-to-peer architecture for media streaming. *IEEE JSAC*. Jan. 2004.
- [15] E. Veloso, V. Almeida, W. Meira, A. Bestavros, and S. Jin. A Hierarchical Characterization of A Live Streaming Media Workload. *IEEE/ACM Trans. on Networking*, 12(5), 2004.
- [16] K. Wong, W. Wong, G. Chan, Q. Zhang, W. Zhu, and Y.-Q. Zhang. Lateral Error Recovery for Application-Level Multicast. *Proc. of IEEE INFOCOM 2004*.
- [17] X. R. Xu, A. C. Myers, H. Zhang and R. Yavatkar. Resilient Multicast Support for Continuous-Media Applications. *Proc. NOSSDAV*, 1997.
- [18] M. Yang and Z. Fei. A Proactive Approach to Reconstructing Overlay Multicast Trees. *Proc. IEEE INFOCOM 2004*.
- [19] E. W. Zegura, K. Calvert and S. Bhattacharjee. How to Model an Internetwork. *Proc. of IEEE INFOCOM '96*, San Francisco, CA.