Path Diversity for Overlay Multicast Streaming

Matulya Bansal and Avideh Zakhor
Department of Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, CA 94720
{matulya, avz}@eecs.berkeley.edu

***Abstract*— In this paper, we propose a new path-diversity based scheme for application layer multicast streaming over the Internet. Rather than building simple trees as in traditional multicast, we construct multicast k-DAGs, characterized by the property that each receiver has k parents. This multiplicity of parents, not only allows for streaming from multiple sources at the same time, thereby decorrelating losses, but also creates an opportunity to dynamically adapt streaming rates from these senders depending on the existing error conditions in the network. To exploit these possibilities, we use a simple rate-allocation algorithm and a packet-partitioning algorithm that allows a receiver to co-ordinate the sending of data from amongst its parents. Our results show that our scheme is effective in dealing with packet losses in the network, and increases the goodput of FEC-coded video data by 15-30%.**

I. Introduction

With the increasing user demand for multimedia content, video multicast is becoming more important for both network service providers and content distributors. The inability of IP multicast to provide the desired support for various multicast applications including video multicast applications has led to the evolution of several overlay based multicast solutions. These overlay or application-layer multicast solutions support the application requirements in several ways including network-adaptive routing, multipath routing, redundant routing etc. In this paper, we propose an overlay multicast scheme for streaming multimedia based on a new content distribution mechanism, called a k-DAG. A k-DAG is a Directed Acyclic Graph in which each receiver has k parents. This differs from the traditional multicast tree, where each receiver has a single parent. As observed in [4], [6], [7], this multiplicity of parents improves multicast streaming performance in two ways. First, by streaming multimedia from multiple parents, it is possible to de-correlate loss among various paths to the receiver, reduce loss-burstiness at the receiver, and therefore improve the effectiveness of schemes such as Forward Error Correction (FEC) [7]. Second, by adapting sending rates from each parent based on loss characteristics of each path, it is possible to minimize overall loss at the receivers.

This paper is organized in the following manner. In Section II, we describe the idea behind a k-DAG and the various trade-offs involved in constructing one. We then present a k-DAG construction mechanism in Section III, and a k-DAG maintenance algorithm in Section IV. In Section V, we present a simple rate-allocation algorithm that allows a receiver to determine streaming rates from each of its parents. This is followed by a receiver-based packet-partitioning algorithm that ensures that all the required data is received while avoiding duplication. Finally, we present simulations comparing our scheme with simple overlay multicast and a non-adaptive variant of our scheme in Section VI.

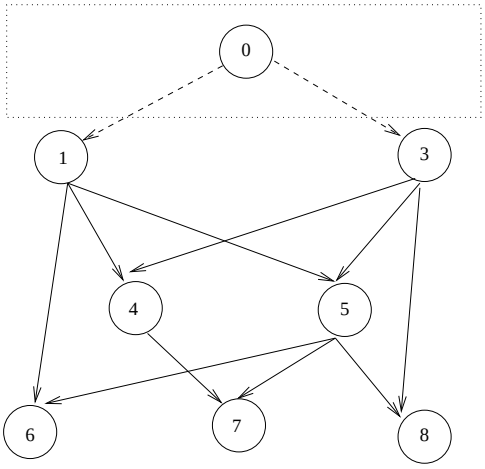

Fig. 1. A simple k-DAG with k = 2. Unlike a traditional multicast tree, every receiver here is connected to 2 parents. The dotted line shows how to handle a single source situation. In this case, the first k nodes that join this source serve as the source nodes for building the k-DAG downstream from then on.

II. k-DAGs

In this section, we present the data-structure for constructing multicast distribution networks such that every receiver has k senders. Since there are multiple parents per node, our proposed distribution network is no longer a tree, but a Directed Acyclic Graph (DAG). Figure 1 shows an example of a simple 2-DAG. We now begin with some definitions and assumptions.

Level of a node: We define the level of a node as the length in hops of the longest path from the source to this node. The level of the source nodes is assumed to be 0.

Degree of a DAG (dag_degree): We define a DAG to have a degree k if all its nodes have k parents. We denote dag_degree by k. Clearly $k = 1$ results in traditional multicast trees with one parent per receiver.

Fraction of bitstream bandwidth reserved by a receiver at each parent ($frac\_bw\_per\_parent$): For a given multimedia session, we define $frac\_bw\_per\_parent$ as the ratio

between the maximum bandwidth allocated by a parent to stream to a receiver, and the total bit rate of the multimedia content being streamed by the source. By definition, this quantity which we denote by b, is always smaller than 1. For the receiver to receive the entire video-stream, we require that $k * b \geq 1$.

Since end-nodes are likely to be limited in outgoing capacity, we assume that receivers proactively $reserve$ the maximum bandwidth they might need, i.e. b * bitstream rate, from any parent for the entire duration of a session. When a receiver $reserves$ bandwidth at a parent, the upstream capacity of that parent decreases by that amount as the parent dedicates that portion of its outgoing bandwidth for servicing a request from this receiver, should the need arise in the future. Though conservative, this assumption ensures fairness to the competing simple overlay multicast scheme. Despite reserving more than the bitstream rate across all parents, at any given point in time, data is streamed at precisely the bitstream rate.

A. DAG-degree, Bandwidth reserved and Depth of the DAG

The degree of the DAG is an important parameter of our proposed scheme. Intuitively, larger DAG-degrees will result in greater flexibility in adapting to losses. However, assuming that the upstream capacity for all nodes stays the same, distribution networks that use k-DAGs are likely to become deeper as $k * b$ becomes large. This is because the total bandwidth reserved across all parents for a given piece of content exceeds its bitrate in order to allow for rate-adaptation. Since the outgoing capacities of nodes are limited, more reserved bandwidth per receiver results in deeper DAGs as nodes at any given level can support fewer receivers.

We now derive the relationship between dag_degree k, frac_bw_per_parent b, and the minimum depth of the DAG d for a given number of nodes N. We assume the outgoing capacity to be the same for all nodes and denote it by c. Also, for simplicity, we assume that each node has all its parents at the same level.

Assume we have p nodes at the first level with the total outgoing capacity being $p * c$. Each new node requires a capacity of $b * k$. Therefore number of nodes that can be supported at the next level is $(p * c)/(b * k)$. Similarly, the number of nodes that can be supported at the next level is $((p * c)/(b * k)) * (c/(b * k))$. This is a geometric progression. Therefore, the maximum number of nodes that can be supported for a k-DAG of depth d, including the p nodes at the topmost level, is given by:

$$p * \frac{(\frac{c}{b*k})^{d+1} - 1}{\frac{c}{b*k} - 1}. \quad (1)$$

Alternately, to support N nodes, the minimum depth of the DAG would be

$$\frac{log(\frac{N}{p} * (\frac{c}{b*k} - 1) + 1)}{log(\frac{c}{b*k})} - 1. \quad (2)$$

The above relations show that for a given number of nodes and a fixed outgoing capacity, the depth of a k-DAG required to support these increases with dag_degree, k and $frac\_bw\_per\_parent$, b.

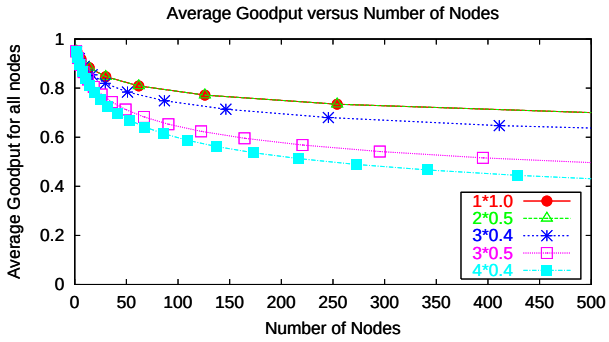

Fig. 2. This figure captures the relationship between the overall expected goodput or packet delivery ratio and $b * k$.

B. DAG-degree, Bandwidth reserved and expected loss

For the same average loss rate per link in the network, deeper trees suffer from increased average loss rates amortized over all the nodes. This is because the longer the path the data is streamed along, the more the number of lossy links it has to encounter enroute to the receivers, and hence greater the loss.

We now derive the trade-off between the reserved bandwidth at the upstream nodes and the increased loss rates due to increase in depth of the tree. Assume that at each level in the DAG, x percent of the packets get lost. Then total loss for a stream of bandwidth say B, as it traverses l levels or hops becomes

$$B(x + x(1-x) + x(1-x)^2 + ... + x(1-x)^{l-1}) = B * (1 - (1-x)^l).$$

Similarly, the amortized loss rate across all nodes for a k-DAG of depth l, containing the maximum number of nodes that it can support, can be shown to be

$$1 - \frac{(1-x) * (\frac{(\frac{c*(1-x)}{b*k})^l - 1}{\frac{c*(1-x)}{b*k} - 1})}{(\frac{(\frac{c}{b*k})^l - 1}{\frac{c}{b*k} - 1})} \quad (3)$$

From Equation (3), we see that the total loss suffered increases as the depth of the DAG increases. Substituting the depth of the DAG constructed for a given number of nodes from Equation (2), we can determine the relationship between loss, the dag_degree k and the bandwidth reserved per parent b. Figure 2 shows average goodput as a function of the number of nodes for different $k * b$ values. As expected, the amortized loss rate increases with an increase in $k * b$. The curves for cases $k = 1, b = 1$ and $k = 2, b = 0.5$ co-incide because $k * b = 1$ in both the cases and the depth and the amortized loss rates are the same.

III. K-DAG CONSTRUCTION

In this section, we present our k-DAG construction algorithm. The DAG construction algorithm shown as Algorithm

1 below, consists of three steps: Finding potential parents, probing these nodes for available network bandwidth and delay, and choosing k nodes from these as parents.

Before proceeding onto the details, we first present the idea of using special overlay nodes, which we simply refer to as Distributed Hash Table (DHT) nodes, for separating the data plane operations such as data forwarding, from control plane operations such as choosing new parents for a incoming node.

Algorithm 1 Steps in Node Join Algorithm

$request\_potential\_parents()$
$probe\_potential\_parents()$
$choose\_k\_parents()$

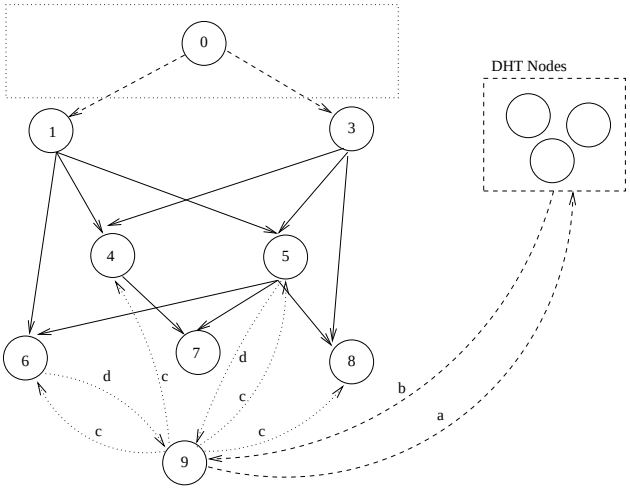

Fig. 3. The process by which a node joins a k-DAG. a) The node finds the address of the DHT nodes by a naming mechanism and requests a list of potential parents; b) The DHT responds by sending a list of nodes with available capacity; c) The node wishing to join the network then probes these parents for available bandwidth; d) The node chooses k of these to be its parents in the k-DAG;

A. Role of the DHT

We use DHTs primarily to facilitate node-joins and node-failure recovery. This is useful from an administrative standpoint as well, since it allows the sender or the owner of the content distribution network to select the join location of a new node in the tree, depending on the price this new node is willing to way, its outgoing capacity, stability or a combination of these.

Another advantage of indirection in the form of DHT nodes is that depending on the size of the multicast session, by choosing a suitable hash function, the control load can be distributed in a centralized or distributed manner. That is, for a small group, all key values may hash to a single node thus making the scheme effectively centralized, while for a large group, the workload might be distributed over a set of nodes, thereby achieving scalability.

1) DHT Mapping: We manipulate information in the DHT by mapping level number of the nodes to DHT nodes. Thus, a given DHT node(s) will maintain information about all the nodes in the DAG at a particular level. Even though this set grows exponentially with level number, it is possible to alleviate this problem by partitioning, either at the DHT or while assigning levels.

2) State maintained at the DHT: The DHT nodes merely maintain information about the level of each node that is a member of the multicast session, and its available capacity. This information is made use of during Node Join and DAG Maintenance phases, as described in Sections III-B and IV respectively.

3) Inter DHT Nodes Communication: We assume DHT nodes to communicate with each other to find nodes at a given level with available capacity. This communication can take place either in the form of periodic proactive or demand-driven reactive message exchange. We do not assume the information received from the DHT nodes, such as the available capacity at nodes, to be always consistent with the actual values at the multicast nodes themselves, even though we expect it to be correct in most cases.

B. Node Joins

Since our scheme is motivated by multimedia multicast and similar content distribution applications, we choose a DAG construction algorithm that allows nodes to join at any time during the lifetime of the multicast session. In the remainder of this section, we describe how a new node can join the multicast session. This is shown in Figure 3.

Since we construct a k-DAG rather than a simple multicast tree, we assume that there exist at least $h * k$ source nodes to begin with where h is the choice factor signifying the flexibility a new node has in choosing its k parents from the initial set of $h * k$ offered by the DHT. In case there is only a single sender, the first $h * k$ nodes to join the session can serve as these $h * k$ nodes. Figure 1 shows such a scenario where $h = 1$.

1) Finding a list of Potential Parents: The first step for a new node to join the multicast session is to find a set of potential parents. In order to do this, the node first needs to know the address of a node maintaining the DHT or alternately that of a member node. This can be obtained via any of the commonly used naming mechanisms such as a publicized URL, DNS etc.

This node then requests the DHT for a list of prospective parents. The DHT returns to it a set of nodes that have outgoing capacity available, based on some criterion such as price-willingness of this node, its stability, or outgoing capacity.

2) Probing Potential Parents: In order to ensure that our overlay links are IP-Network efficient, nodes evaluate the quality of their paths to prospective parents before selecting them. We use a slightly modified packet-pair probing technique [11] to estimate the available bandwidth and delay whereby a set of spaced packet-pairs is sent to potential parents to evaluate

path quality.

3) Parent Selection: Once a node has evaluated the quality of available paths, it chooses the parents it wants to stream from. Several criterion might be employed for this purpose including paths with maximum available bandwidth, or minimum delay, or paths that reduce jitter or minimize buffer requirements, or a combination of these. In our current implementation, nodes choose the k widest paths.

4) Updating state: Once a node has chosen its list of parents, it requests them to join in as a child. Depending upon the available capacity, this node may or may not choose to accept this child. This is because the advertised capacity by the DHT may not necessarily be consistent with the actual capacity at a given parent at all times. If accepted, the new node has completed finding one of its k parents. If rejected, it may decide to choose the next parent on its list of perspective parents, or else it may choose to request the DHT again. Once a node is accepted by k parents, it has successfully joined the k-DAG multicast session.

IV. DAG Maintenance Algorithm

Problems might arise when nodes fail in the network or abruptly terminate their multicast session. To mitigate the effect of such occurrences, we now present our DAG maintenance algorithm. While a parent node crash in a k-DAG results in a large number of $orphan$ nodes as compared to a simple multicast tree, the effect of a parent loss on an individual child is less pronounced since each child has multiple parents. With an outgoing capacity of c and fraction of bitstream bandwidth requested from each parent b, the crash of a fully-loaded node results in c nodes being affected in a simple multicast tree and c/b nodes being affected in a k-DAG where $b \leq 1.0$. However, in the traditional multicast case, any orphaned node suffers from a complete blackout while in case of a k-DAG, orphaned nodes only lose a fraction b of the total bitstream data. Moreover, orphaned receiving nodes can potentially mitigate the effects of a parent loss by reserving spare upstream bandwidth ahead of time and adjusting the rate from surviving parents, until a new parent is found.

Algorithm 2 Steps in Node Failure Recovery Algorithm

$rate\_adaptation()$
$request\_potential\_parents()$
$probe\_potential\_parents()$
$choose\_parent()$
$inform\_victim\_if\_required()$
$rate\_reallocation$

We now briefly discuss our node failure recovery algorithm as shown in Algorithm 2 above. When a node detects a parent loss, it immediately invokes a rate-reallocation algorithm, discussed in Section V, to minimize the impact of this loss. In case of a k-DAG where the upstream bandwidth reserved per sender is b, up to s node failures can be successfully tolerated, where

$$s = \max_{r}(k - r) * b > 1.0. \quad (4)$$

In order to find a new parent, a node simply sends a request for a list of new potential parents to the DHT nodes. The DHT nodes need to ensure there are no loops formed as a result of new parent-child relationships. This is where the notion of level comes in. By enforcing that a DHT node only choose a node whose level is less than or equal to the level of the requesting node as a potential parent, we guarantee no loops provided that the level values at all the actual nodes and that stored at the DHT are consistent. To ensure the consistency of level values, we enforce that no node whose level has changed within a certain time-interval respond positively to a parent request or request a new parent. This time interval is chosen to be the time it takes for a level update to propagate through the network.

If the DHT nodes, however, cannot find capacity at any of the nodes with a level lower than that of the requestor because they are already loaded to capacity, the DHT returns to the requestor a list of its siblings. The requestor then probes and chooses one or more of these nodes as its potential parents.

If a node higher up the DAG is chosen and consents, the requesting node has found a parent. If a sibling is chosen and has capacity, again the requesting node has found a new parent. However, since a sibling is chosen, the requesting node needs to update its level and potentially of its descendents. If a sibling does not have the necessary capacity because it is already fully loaded, it chooses one of its children as a victim, aborts it by telling it to find a new parent, and after allowing it sufficient time to adjust its upstream rates, accepts the requestor as its child. Intuitively, a child of a sibling might need to be aborted in favour of the requesting node because in order to avoid loops, an orphan node needs to be assigned a new parent which is not its descendent. By definition of level, this child has a greater level than the orphaned node and therefore the algorithm is guaranteed to terminate. Under extreme circumstances, there might be cascaded node failures with the number of orphaned nodes in the worst case being $max(outDegree) * height$ of the failed node, where $height$ of a node is defined as the maximum difference between the level of any node in the DAG and the level of this node; nonetheless, node failures and aborts cause the orphaned nodes to only lose a fraction b of the bitstream. In addition, if the product of degree of the DAG, k, and the fraction of bitstream bandwidth requested from each parent, b, is sufficiently high, as seen in Equation (4), a limited number of node failures may not result in any degradation of the FEC coded multimedia content at all.

The above node-failure recovery algorithm ensures recovery from multiple node failures in a distributed fashion without the possibility of occurrence of loops. Once a node has chosen and received acceptance from a new parent, it performs rate-reallocation and packet-partitioning, as described in the next section, and updates the DHT with the new information.

V. Rate Allocation and Packet Partitioning Algorithm

This section describes the rate adaptation algorithm used by the receiver to optimally allocate the sending rates among its parents. Our rate-allocation algorithm is similar in spirit to that in [7] with some minor differences. Synchronization issues are handled within the packet partitioning algorithm in a manner similar to the techniques described in [7].

A. Initial Rate Assignment

When a node initially joins the multicast session, it is unaware of the expected loss rates from its parents. However, during the probing process to choose its parents, it has an estimate of the available bandwidth and delay. So, to begin with, a receiver divides the stream bandwidth equally amongst its parents. That is, for a dag_degree k, it streams $1/k$ of the bitstream from each of its parent.

B. Loss Estimator

We use an exponential moving average loss estimator for estimating losses. We sample loss rates every five seconds by counting the number of packets lost in this period. However, since we stream from senders at different rates, to make consistent comparisons across senders, we need to weigh the sample by the proportional rates at which data is being received from the senders. Hence we calculate loss_rates by

$$loss\_rate[i+1] = (1-\alpha)*loss\_rate[i]+\alpha*\beta*cur\_loss\_rate$$

where α is the averaging parameter and β is the fraction of the bitstream currently being streamed from this parent. cur_loss_rate is the instantaneous loss rate and is obtained by dividing the difference between the number of expected and received packets by the number of expected packets over this five second interval.

C. Rate Reallocation

During the course of the multicast session, network conditions might change resulting in bandwidth fluctuations across nodes. In our scheme, a receiver performs rate-reallocation whenever the loss-rate from a parent exceeds an experimentally determined threshold, or the quality of path between any two parents differs significantly.

D. Rate Allocation Algorithm

We use a rate allocation algorithm similar to the one used in [7]. As in [7], through the rate allocation algorithm, we minimize the overall packet loss experienced by a receiver by sending as many packets as possible along the path that experiences the lowest loss subject to maximum bandwidth that the receiver had reserved from this parent. This approach is applicable whenever $b*k > 1$ wherein it is possible to reduce the overall loss rate experienced by a receiver by requesting more of the video stream from the parent upstream the less error-prone path. This essentially corresponds to a water-filling algorithm.

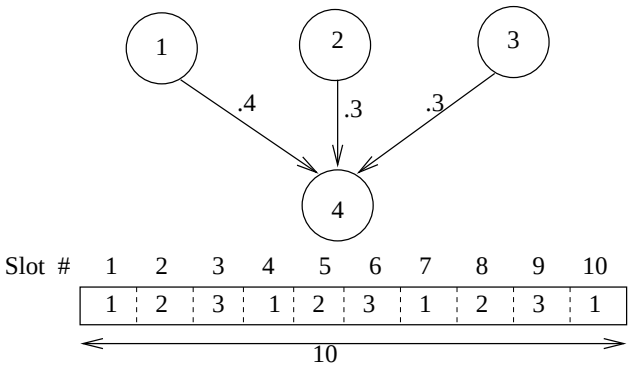

Fig. 4. Proposed Packet Partitioning Algorithm.

To quantify this further, assume that we have a k-DAG, and that there exists a node with k parents. Further assume that this node has measured the loss rates from each of its parents $1, 2, ..k$ to be $l_1, l_2, ..., l_k$, and that they are in an increasing order. With the given rate reallocation mechanism, the overall loss rate experienced by this receiver is bounded by $\sum_{p=1}^{i} l_p * b$ where $i = \min_r \sum_{p=1}^{r} b_p \geq 1$.

E. Packet Partitioning

Once a node has decided the streaming rates for each of its parents, a packet-partitioning algorithm is needed to determine which packet is sent by which parent. Unlike the proposed partitioning algorithm in [7] which is sender-based and would have maintained a $O(n^2)$ state in the number of receivers in a multicast scenario, the algorithm we use requires $O(n)$ state in the number of receivers at the expense of not minimizing packet delay. We believe that this trade-off is specially suitable for a multicast scenario where excess load on the end-systems needs to be at a minimum.

Figure 4 shows an example of our packet-partitioning algorithm. The slotsize in this example is 10. The receiver node 4, has 3 parents, i.e. nodes 1, 2, and 3. After loss measurements, it has chosen to stream 0.4 of the bitstream from node 1, 0.3 from node 2 and 0.3 from node 3. The receiver first discretizes the request rates and assigns a proportional number of slots from a slotsize to each parent. With a slotsize of 10, 4 slots are assigned to node 1, and 3 each to nodes 2 and 3. Requested slots are interleaved to minimize bursty losses. This information, along with the requested rate, is then sent in a rate_request packet to the parents. As an example, node 1 sends packets 1, 4, 7, 10, 11, 14, 17 ..., node 2 sends 2, 5, 8, 12, 15 ... and node 3 sends packets 3, 6, 9, 13, 16 ... to receiver 4.

F. Informing Parent Nodes

When a receiver node chooses to change rates, it sends out a new rate_request packet to its parents. The information in this rate_request packet is similar to the information sent after

packet-partitioning, and consists of the new rate requested by the receiver from this sender, and the sequence numbers this sender is responsible for sending to this receiver. The parents switch to this new request pattern after receiving the request but continue to send according to the old pattern for a short period of time in order to avoid losses during the transition.

VI. Performance Evaluation

We evaluate the performance of our scheme using NS-2 [10]. We compare our proposed scheme with conventional overlay multicast, and a non-adaptive variant of our scheme.

A. Simulation Parameters

We generate the topologies for our simulations using the BRITE[18] topology generator. We use two different types of topologies for our simulations, based on Albert-Barabasi model and Two-Tier Hierarchical model. For compactness though, we present results only for Albert-Barabasi model. We have found the results for the Hierarchical model to follow similar trends. We use 500 network nodes and 250 overlay nodes. All the overlay nodes are multicast receivers. Only one DHT node is used. The overlay nodes and their order of joining are both randomly chosen for each simulation. We use 7 source nodes for all the simulations. The bitstream rate is 128Kbps and the packet size is 512 bytes. A (21,7) FEC code is used for protecting the data. The outgoing capacity of all the nodes is fixed to be twice as large as the video bitstream rate and the slotsize is 20. Loss rates are updated, and the decision to reallocate rates is evaluated every 5 seconds. All simulations are run for a duration of 30 minutes. Every point in a plot corresponds to an average of 100 repetitions. The source is started at the beginning of the simulation and all the overlay nodes join the multicast session within the first 4-5 minutes. Losses are introduced once all the nodes have joined the session.

B. Metrics

We use the following metrics for measuring the effectiveness of our scheme.

Goodput or Packet Delivery Ratio This is the ratio of number of packets received to the number of packets expected.

FEC goodput We define FEC goodput as the ratio between the number of non-redundant FEC blocks successfully received to the number of non-redundant FEC blocks expected by the multicast receivers. This metric evaluates and compares the performance of different schemes from an application standpoint.

Average delay experienced by the end-nodes Average delay is the average time it takes for a data packet sent by the source to reach a receiver. As seen later, there is a trade-off between loss rate and average delay experienced by nodes.

Average jitter experienced by the end-nodes We define jitter as the standard deviation in the delay experienced by a receiver. Average jitter, calculated across all the receivers, indicates the buffer requirements at the receivers. Having multiple senders is likely to result in increased jitter. Even though with increasing end-node capacities, buffer size is no longer an issue, we characterize the tradeoff between required buffer size and path diversity.

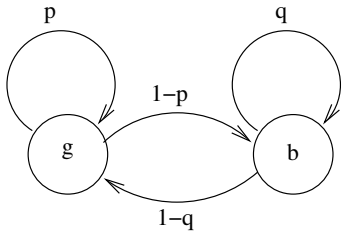

Fig. 5. This figure shows the two state Markov model we use for modelling losses in the network. State g represents the good state, while state b represents the bad state. No packets are lost in the good state, while $error\_rate$ percent of the packets are lost in the bad state. $error\_rate$ is a parameter of the simulation. We set $p = 0.85$ and $q = .75$ for our simulations.

Average Control Overhead We define control overhead as the ratio between the number of control packets sent and the data packets received.

C. Loss Model

We use a two state Markov loss model shown in Figure 5 for generating losses. We vary the probability of losing packets in the bad state as a parameter, while keeping the probability of losing packets in the good state at 0. So an error rate of 30 percent corresponds to the probability of a loss in the bad state being 0.3. In all our error experiments, we randomly choose and inject 20 percent of all the links with a specified error rate for a duration of 5 minutes after which the error rate on these links is set to 0 and a new set of links is chosen. We fix the probability of moving from good state to good state to be 0.85, while the probability of moving from bad state to bad state is fixed at 0.75.

D. Legend

We refer to all the schemes by a dag_degree * frac_bw_per_parent notation. So, a $1 * 1.0$ scheme is simple overlay multicast, a $2 * 0.5$ scheme uses a 2-DAG and splits the bitstream evenly between the two senders, and a $3 * 0.4$ scheme builds a 3-DAG and in which each receiver can stream upto 40% of the bitstream from each parent.

E. Link Losses and Path Diversity

We examine the effect of using multiple upstream senders per multicast receiver on burst lengths when losses prevail in the network. To evaluate this effect, we construct a simple multicast tree, a 2-DAG with $b = 0.5$, a 4-DAG with $b = 0.25$, and a 5-DAG with $b = 0.2$ and stream FEC coded data at 128kbps from the sources. The total bandwidth reserved at all the parents per receiver in each case is also 128kbps, hence no rate-adaptation is possible.

Figure 6 shows the percentage of lost packets that occur in a burst of size greater than s, as a function of s, when 20% of the links in the network suffer from a loss rate of 30%. While a simple multicast tree experiences more that 50% of its losses

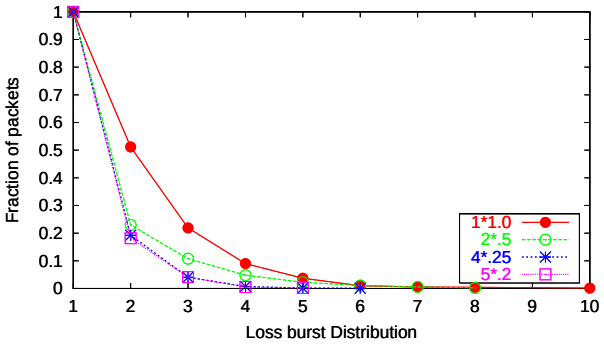

Fig. 6. Loss burst distribution for a 100 overlay node network when 20% of the links were injected with an error rate of 30%. For each lost burst size s, this plot shows the fraction of packets that were lost in a burst of size s or more.

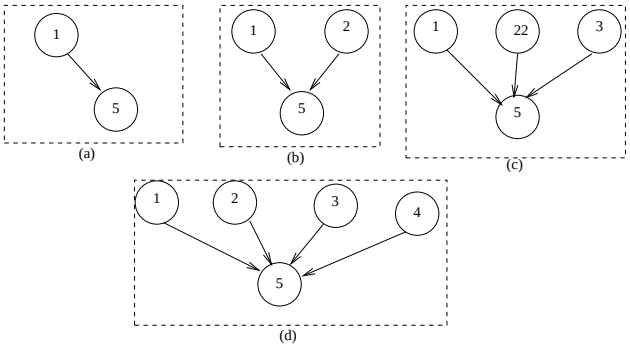

Fig. 7. Setup for observing microscopic behaviour. Node 5 is chosen for observation in all the cases. Its upstream connectivity in a 1-DAG is shown by (a), in a 2-DAG by (b), in a 3-DAG by (c) and in a 3-DAG by (d). Errors are introduced on the link between nodes 1 and 5. The remaining part of the DAGs is not shown for simplicity.

in bursts of length 2 or more, the corresponding percentages for a 2-DAG is less 25%, and for 4-DAGs and 5-DAGs, it further reduces to around 20%. Similar results were obtained for other values of loss rates. Thus path-diversity is successful in reducing bursty losses in the network. Intuitively, this can be explained by considering that in a k-DAG with $k > 1$, the senders split the bitstream and interleave the packets. As such, unless all the links to the parents are experiencing loss at the same time, loss burst-lengths at the receiver are reduced.

F. Microscopic Behaviour

To understand the microscopic behaviour of the schemes, we perform a simple experiment in which errors are introduced on a single link between a node and one of its parents in k-DAGs of different degrees. Figure 7 shows the chosen node 5 and its upstream connectivity for different cases. The rest of the DAG is not shown for simplicity. None of the parents of node 5 share a link in the underlying network. An error rate of 40% is introduced on the link between node 1 and node 5 for a period between 100 to 150 seconds.

Figure 8 shows the sequence number of packets lost at node 5 for different schemes over a window of 25 seconds. We

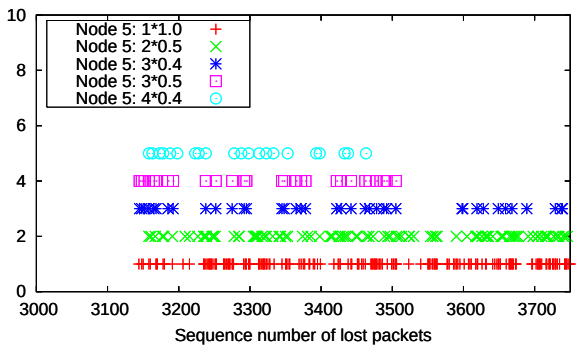

Fig. 8. Sequence numbers of packets lost between time=95 to 120 seconds for different schemes

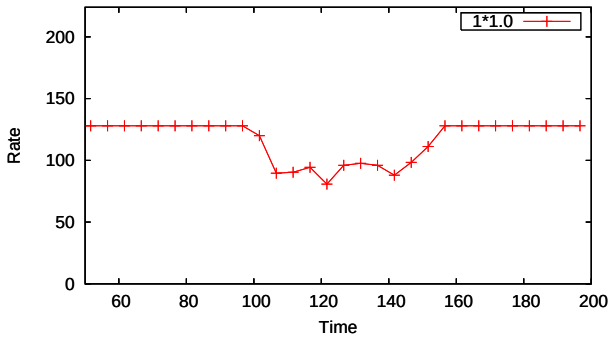

Fig. 9. Received rate at node 5 for $1 * 1.0$ scheme

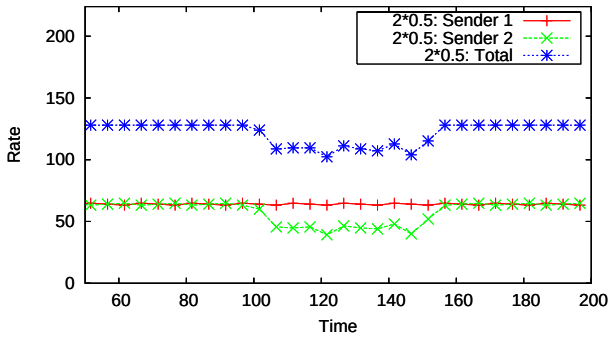

Fig. 10. Received rate at node 5 for $2 * 0.5$ scheme

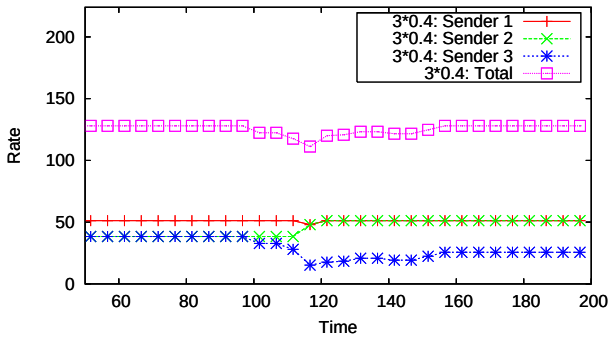

Fig. 11. Received rate at node 5 for $3 * 0.4$ scheme

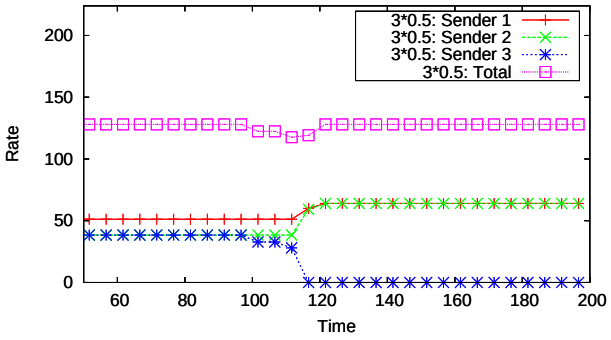

Fig. 12. Received rate at node 5 for $3 * 0.5$ scheme

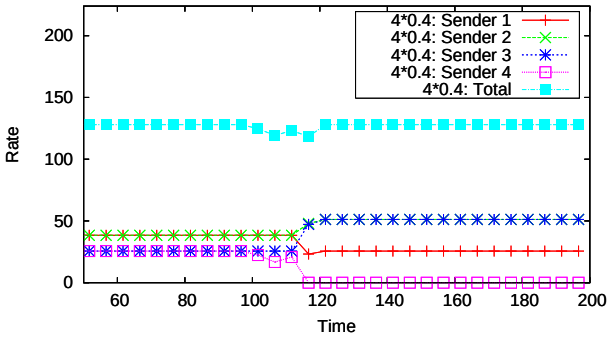

Fig. 13. Received rate at node 5 for $4 * 0.4$ scheme

observe that an increase in dag_degree results in fewer and less bursty losses. We also observe that unlike schemes $1 * 1.0$ and $2 * 0.5$, schemes $3 * 0.5$ and $4 * 0.4$ successfully adapt to the loss conditions in the network and thereby avoid losses later into the experiment, even though the error conditions still prevail on the lossy link. Scheme $3 * 0.4$ also adapts, however it is limited by the capacity it has reserved and therefore has to stream from the parent upstream the lossy link at a reduced rate. Figures 9, 10, 11, 12, and 13 show the received rate at node 5 as a function of time during the course of the experiment. Unlike $1*1.0$ which performs poorly, $3 * 0.5$ and $4 * 0.4$ are able to maintain a received bandwidth above 115kbps. Of these two, scheme $4 * 0.4$ performs better since even during the transition period, it streams only one-fourth of the bitstream from the affected node. Scheme $2 * 0.5$ also performs poorly, even though it experiences fewer overall losses than $1 * 1.0$ as it streams at only half the bitstream rate from the affected link. Scheme $3 * 0.4$ performs better than schemes $1 * 1.0$ and $2 * 0.5$ but worse than schemes $3 * 0.5$ and $4 * 0.4$ because of its limited ability to adapt.

G. Link Error Losses

We examine the overall performance of our scheme by injecting losses with rates ranging between 0-50% on 20% of links in the network. Figure 14 shows FEC goodput achieved by different schemes under varying loss conditions in the network. The results confirm the observations made in the

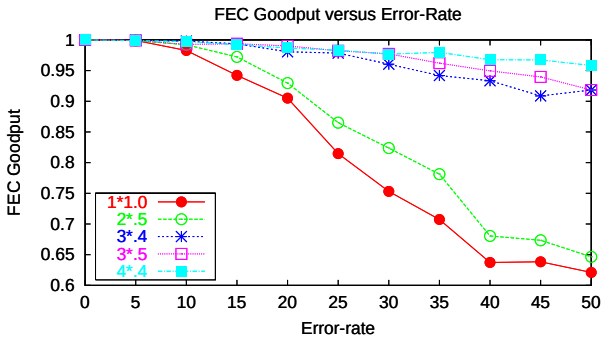

Fig. 14. FEC Goodput with rate-adaptation.

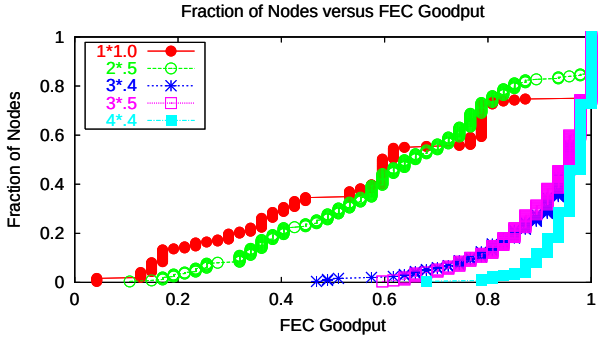

Fig. 15. The cumulative fraction of nodes that receive a FEC Goodput less than f, as a function of f.

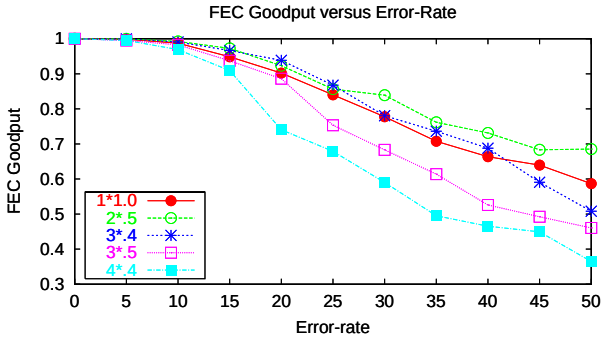

Fig. 16. FEC goodput without rate-adaptation.

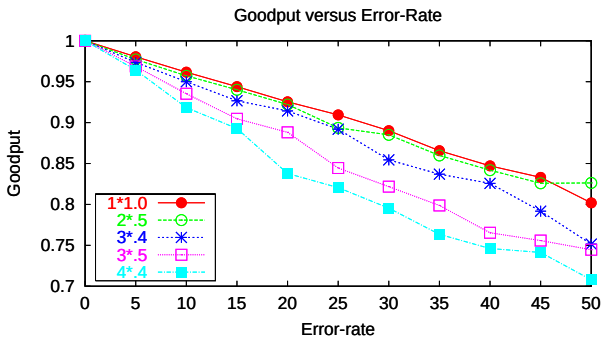

Fig. 17. Goodput without rate adaptation for different schemes.

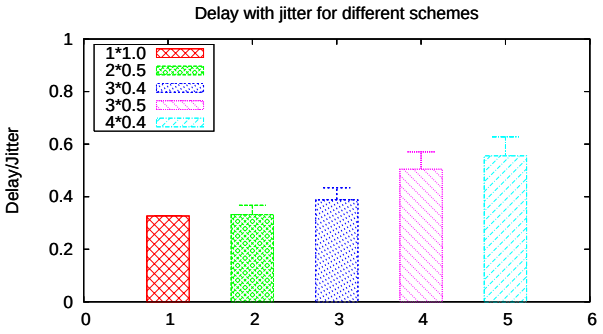

Fig. 18. Delay and Jitter for Different Schemes. The bars show the delay while the vertical line between the bars and the small horizontal line at the top shows jitter.

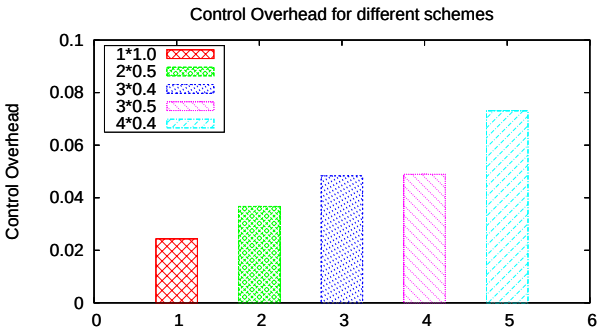

Fig. 19. Control Overhead for different schemes.

earlier experiment. Scheme $4 * 0.4$ has the highest FEC goodput, followed closely by $3*0.5$ and $3*0.4$. $3*0.4$ manages to perform well for FEC goodput even with its limited ability to adapt because FEC coding successfully mitigates the effect of a limited number of packet losses. Also, the $2*0.5$ scheme without rate-adaptation outperforms simple overlay multicast because of the loss decorrelation property of path-diversity.

Figure 15 shows the cumulative FEC goodput distribution across all nodes for a case where 20% of the links are injected with 50% loss rate. As seen, schemes that employ path diversity with rate adaptation outperform simple overlay multicast across nodes. More importantly, even nodes that are deeper in the k-DAG than they would have been in a simple multicast tree due to the fact that $k * b > 1$, achieve a gain in FEC goodput.

Figures 16 and 17 show FEC goodput and goodput values without rate-adaptation respectively. As seen, without rate-adaptation, losses due to increased depth of the DAG offset the advantages of path diversity both for goodput and FEC goodput metrics. Figure 18 compares the delay and jitter values for different schemes. As expected, average delay increases with $b * k$ while jitter increases with k.

Figure 19 shows the control overhead for the different schemes. As expected, the control overhead is proportional to dag_degree and is quite low, i.e. less than 7%.

VII. Related Work

Multicast Streaming has recently been an active area of research and as such, several interesting overlay multicast schemes have been proposed. In this section, we present a brief overview of existing related work and compare our scheme to them.

Narada [3] builds a dynamic DVRMP style overlay multicast tree for video conferencing. Even though it adapts the overlay topology to changing network conditions, every receiver is essentially connected to a single parent.

Splitstream [1] addresses the issue of handling node failures by building multiple multicast trees such that any node is an interior node in at most one of the trees. Each tree corresponds to a layer in MDC coded multimedia stream. Our scheme is different in that it handles FEC coded streaming multimedia. In addition, Splitstream does not explicitly distinguish between node failures and network losses.

Co-Op Net [2] is similar in spirit to Splitstream. It too builds multiple multicast trees each for streaming a single MDC layer. However, Co-Op Net is centralized with tree management operations based at the source rather than being peer-to-peer.

In terms of motivation, distributed video streaming (DVS) [4], [6], [7], [8], [9] comes the closest to our work. DVS uses both path diversity and rate-adaptation for streaming video content from multiple sources to a receiver. In this paper, we have extended this idea to multicast by introducing k-DAGs.

Informed Content Delivery [5] addresses the problem of enabling large file transfers over overlay networks. It uses collaborative transfers between receivers to exploit potentially rich connectivity between the receivers. However, the main focus is primarily on traditional data transfer applications rather than multimedia streams limiting the applicability of their coding and reconciliation mechanisms.

VIII. Conclusion

In this paper, we have proposed a new mechanism for application layer multicast streaming of multimedia data over the Internet. The proposed content distribution structure, k-DAGs, allows receivers in the multicast session to have multiple parents. This differs from traditional multicast approach, where each node has only one parent. This multiplicity of parents enabled by k-DAGs enables two possibilities. First, by streaming data from multiple senders, we are able to decorrelate losses experienced by the end-nodes, thereby improving the performance of FEC coded streams. Second, by dynamically adapting the requested video streaming rates between its various parents, a receiver can mitigate the effect of outages in the network. We use a simple rate-allocation algorithm and a packet-partitioning algorithm to enable these possibilities. Our results indicate that our scheme improves FEC goodput of streaming multimedia by 15-30%. This performance gain comes at the cost of increased delays as well as higher jitter values. While with increasing end system capabilities, we do not expect jitter to be a major issue, the small additional

delay experienced, we believe is a small cost for the additional performance achieved.

ACKNOWLEDGMENTS

The authors would like to thank Thinh Nguyen and David Harrison for the useful discussions during the course of this project.

REFERENCES

[1] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron and A.Singh, "Splitstream: High-Bandwidth multicast in a cooperative environment", *SOSP'03*, Lake Bolton, New York, October 2003.

[2] V. N. Padmanabhan, H. J. Wang, P. A. Chou and K. Sripanidkulchai, "Distributing Streaming Media Content using Co-operative Networking", *ACM NOSSDAV*, Miami Beach, Florida, May 2002.

[3] Y. Chu, S. Rao and H. Zhang, "A case for end system multicast", *In Proc. of ACM Sigmetrics*, pp 1-12, June 2000.

[4] T. Nguyen and A. Zakhor, "Distributed Video Streaming Over the Internet", *SPIE*, San Jose, California, January 2002.

[5] J. Byers, J. Considine, M. Mitzenmacher and S. Rost, "Informed Content Delivery Across Adaptive Overlay Networks", *SIGCOMM 2002*, Pittsburg, Pennysylvania, August 2002.

[6] T. Nguyen and A. Zakhor, "Path Diversity with Forward Error Correction (PDF) System for Packet Switched Networks", *IEEE INFOCOM 2003.*

[7] T. Nguyen and A. Zakhor, "Distributed Video Streaming with Forward Error Correction", In *International Packet Video Workshop*, Pittsburg, Pennysylvania, April 2002.

[8] J. Apostolopoulos, "Reliable video communication over lossy packet networks using multiple state encoding and path diversity", *In Proceedings of The International Society for Optical Engineering*, January 2001, vol. 4310, pp. 392-409.

[9] J. Apostolopoulos, "On multiple description streaming with content delivery networks", *IEEE Infocom 2002*, June 2002, vol. 4310.

[10] K. Fall, "NS Notes and Documentation: The VINT Project", 2000.

[11] S. Keshav, "Packet-Pair Flow Control", *IEEE/ACM Transactions on Networking*, February 1995.

[12] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee and S. Khuller, "Construction of an Efficient Overlay Multicast Infrastructure for Real-Time Applications", *Infocom 2003*.

[13] X. R. Xu, A. C. Myers, H. Zhang and R. Yavatkar, "Resilient Multicast Support for Continuous-Media Applications", in *IEEE/NOSSDAV*, 183-194, 1997.

[14] G. Kwon and J. Byers, "Smooth Multirate Multicast Congestion Control", *IEEE Infocom 2003*, April 2003.

[15] Z. Wang and J. Crowcroft, "Quality-of-Service Routing for Supporting Multimedia Applications", *IEEE Journal on Selected Areas in Communications* 14(7): 1228-1234 (1996).

[16] M. Jain and C. Dovorolis, "End-to-end available bandwidth: Measurement methodology, dynamics and relation with TCP throughput", *SIGCOMM* 2002, Pittsburg, Pennysylvannia, August 2002.

[17] Z. Wang and J. Crowcroft, "Bandwidth-Delay Based Routing Algorithms", *IEEE GlobeCom* 1995, Singapore, November 1995.

[18] A. Medina, A. Lakhina, I. Matta and J. Byers, "BRITE: An Approach to Universal Topology Generation", In Proceedings of *MASCOTS 2001*, Cincinnati, Ohio, August 2001.

[19] D. Wu, Y. T. Hou, W. Zhu, Y. Zhang and J. M. Peha, "Streaming Video over the Internet: Approaches and Directions", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol 11, No. 3, March 2001, pp. 269-281.

[20] Y. Wang and Q. Zhu, "Error Control and Concealment for Video Communication", *Proceedings of the IEEE*, 86(5):974 997, May 1998.