Efficient Bandwidth Guaranteed Restoration Algorithms for Multicast Connections

William Lau[1], Sanjay Jha[1], and Suman Banerjee[2]

[1] University of New South Wales, Sydney, NSW 2134, Australia,
wlau,sjha@cse.unsw.edu.au
[2] University of Wisconsin-Madison, Madison, WI 53706, USA,
suman@cs.wisc.edu

Abstract. This paper defines a new restoration strategy for provisioning bandwidth guaranteed recovery for multicast connections in presence of link failures. The new restoration strategy is formulated into a new Integer Linear Programming (ILP) algorithm and is compared with other existing restoration strategies. We also present a new heuristic algorithm based on this new restoration strategy. Results show that our heuristic algorithm performs competitively close to the ILP-based algorithm, and is more bandwidth efficient than other existing heuristic algorithms that are based on different restoration strategies.

1 Introduction

Networks are fast becoming the central medium for delivering rich multimedia contents to the clients that have the need and willingness to pay for the contents. With the trend of wide-spread broadband network access, content providers [1, 2] are already starting to deliver network-based multimedia service to the end users. There is also a emerging trend for content providers to use terrestrial networks as complement to the traditional satellite networks [3] for delivering contents to sites that their satellites cannot reach. There is a potential of using multicast solutions in terrestrial network in place of satellite network in the middle and long term due to the following factors: First, long disruptions to service due to bad weather are eliminated. Second, optical backbone bandwidth is growing at a much faster pace than satellite networks. Third, the accessibility and reachability of terrestrial networks are growing at a tremendous pace. We believe that these factors will be the main driving force for supporting multicast in backbone networks. However, existing work based on best effort routing [4, 5] and overlays [6, 7] will not be able to satisfy the stringent requirements of content service providers such as bandwidth guarantees, minimal disruption time, and ability to scale to large number of multicast connections. There is a need for traffic engineering capability that can satisfy these requirements.

The focus of our work is on the intra-domain multicast connections where the source node and the destination nodes reside in the same network domain. Each domain's source node can be considered as a Point-of-Presence (PoP) for

the content provider. Contents are delivered to the PoP for distribution within the domain. Inter-domain solutions for connecting the PoPs are outside the scope of this paper. We assume that the underlying network architecture is connection-oriented, such as MPLS [8] or GMPLS[9], which supports Point-to-Multi-Point (P2MP) [10] label switching delivering capability. This is required to allow the restoration strategy more control over the routing of the traffic in order to better utilize the links in the network. Another assumption is that only bandwidth recovery guarantees for single failures are considered because single failures are the most common type of network failures [11]. However, due to the space limitations, only single link failures are investigated in this paper. This paper proposes a new restoration strategy for provisioning bandwidth guaranteed recovery for multicast connections over mesh-based backbone networks. This restoration strategy is based on pre-computing the routes to take during failure and pre-allocate the resources required for the traffic during failure. This form of pre-planning allows bandwidth guaranteed recovery and also lowers the disruption time of the multicast connection.

We use the online pre-planning technique [12] where each connection request is pre-planned on-demand. That is, as the request is made to the network, it is pre-planned based on the current resource state of the network and does not need prior knowledge of other requests that have not been admitted into the network. This better suits on-demand multicast applications. We derived a new Integer Linear Programming (ILP) formulation based on this new restoration strategy and compare the results with ILP formulations for other existing restoration strategies. The results show that our restoration strategy requires significantly less bandwidth than the other strategies and has higher number of accepted connection requests. We also propose a new heuristic algorithm based on the new restoration strategy, and the results show that the heuristic algorithm performs competitively close to the ILP-based algorithm. Our heuristic algorithm is compared with other existing heuristic algorithms that use different restoration strategies, results show that our heuristic algorithm is more bandwidth efficient.

The rest of this paper is organized as follows. The related work is discussed in section 2 and then the new restoration strategy is presented in section 3. The experimental setup is described in section 4. The results for comparing between the different restoration strategies are presented and analyzed in section 5. In section 6, a new heuristic algorithm is introduced and the performance is evaluated in section 7. This paper concludes in section 8.

2 Related Work

Compared to unicast works, guaranteed bandwidth restoration strategies for multicast are limited. Most literatures focus on connectivity issues or probabilistic approaches. This section discusses the related algorithms.

Medard et al [13] propose an algorithm to compute service tree and a backup tree that is link (or node) redundant. Link redundant means that for a single link failure, all the nodes in the tree are still connected to at least one of the

tree from the source. Thus this is a stronger requirement than required by a multicast tree which requires only the leaf nodes by protected.

Kodialam et al. [12] propose an approximation algorithm based on the line restoration strategy. Line restoration refers to the use of backup path segments that re-route the multicast traffic around the failed components and then back onto the multicast tree. For bandwidth efficiency, backup bandwidth can be shared between backup path segments. In this paper, we compare the performance differences between line restoration strategy and our tree restoration strategy using simulation experiments.

Singhal et al. [14] propose several new multicast restoration strategies using online planning. The best performing restoration strategy from their work is called Optimal Path-Pair (OPP). OPP uses disjoint path pairs between the source and each destination. Their results showed that OPP performs significantly better than link redundant trees [13]. However, their algorithm does not share backup bandwidth between backup trees that protect different service trees. This paper compares OPP to our proposed tree restoration strategy.

3 New Restoration Strategy

We propose a new restoration strategy based on the online pre-planning technique [12] where each connection request is pre-planned on-demand. Thus planning is made one request at a time and the objective of the restoration strategy is to minimize the bandwidth requirement for provisioning bandwidth guaranteed recovery for this requested multicast connection. The problem of finding the least cost multicast tree is a well-known NP-Hard problem (Steiner Tree), therefore, we separate the computation of the service multicast tree from the computation of the backup multicast tree. A service tree is used for delivering traffic during normal operation of the network while the backup tree is used to deliver traffic during network failures. For the service tree computation, we use an existing Integer Linear Programming (ILP) formulation [15] that calculates the optimal least cost multicast tree. Given a service tree, we propose a new restoration strategy that protects the service tree from single link failures.

Assuming single link failures, it is possible to define a set of failure scenarios with each scenario covering a different link failure. Each failure scenarios are considered to be independent of each other, as they are not expected to occur simultaneously. Given a service tree, we can identify the subset of failure scenarios that affects the service tree. For each of these failure scenarios, we can calculate a new multicast tree that is used to deliver traffic during this failure. Thus there are multiple backup trees protecting one service tree and the backup tree used during a failure depends on the state of the network (the failure). During a failure, the traffic is switched from the service tree to the appropriate backup tree. Thus the objective is to find the set of backup trees that protect the given service tree, and also to minimize the backup bandwidth required. To minimize the backup bandwidth requirement of the network, we use two techniques: Backup bandwidth sharing between backup trees belong in different

Table 1. Mathematical Notations

Symbol	Definition	Symbol	Definition
V	set of vertexes.	E	set of edges.
$G(V, E)$	The network graph	FS	set of failure scenarios.
s	A failure scenario in FS.	m	service tree.
FS_m	set of failures affecting m	h	source node.
d	A destination node in the request.	D	The set of destination nodes.
(i, j)	link connecting i to j.	b	requested bandwidth.
A_{ij}	bandwidth available in link.	C_{ij}	backup bandwidth in link.
C^s_{ij}	backup bandwidth required in (i, j) for s.	E^d_m	set of edges connecting source to d on m.
E_s	set of failed edges in s.	E_m	set of edges on m.
D_s	set of destination nodes affected by s.	τ^s_{ij}	cost for using (i, j) in s.

failure scenarios, and Path Intermix [16] that allows the backup tree to reuse the bandwidth belonging to the service tree.

Suppose each link in the network reserves bandwidth used during normal operations (service bandwidth) and bandwidth used during failures (backup bandwidth). Establishing a service tree involves allocating more service bandwidth on the links used by the tree, and establishing a backup tree involves allocating more backup bandwidth. All service trees affected by a failure scenario are required to have a corresponding backup tree that protects the service tree from the failure. The total bandwidth required by the backup trees on link (i, j) (connecting from node i to node j) in failure scenario s is denoted by C^s_{ij}. Since failure scenarios are independent of each other thus the backup trees in different failure scenarios are not expected to be used at the same time. This allows the backup bandwidth between failure scenarios to overlap (share), hence we deduced the following relationship: $C_{ij} = \max C^s_{ij}, \forall s \in FS$

Where FS denotes the set of failure scenarios and C_{ij} is the total backup bandwidth allocated on link (i, j). Further, during a failure, the service bandwidth allocated to the service tree is idle since the traffic is switched to the backup tree. Thus there is an opportunity for the backup tree to reuse the service tree's bandwidth. The concept of using different type of bandwidth on different links in a backup tree is called Path Intermix [16]. It is likely that a link failure only affects only a small portion of the service tree, that is, only a subset of the destination nodes are affected. The sub-tree that connects the source to the unaffected destinations is used as the *skeleton* that forms the basis of the backup tree. The skeleton structure reuses the service bandwidth. To complete the backup tree, the skeleton tree is connected back to the affected destinations.

We call this the *Skeleton-Tree* restoration strategy. We formulate this restoration strategy into a new ILP problem that, given a service tree, finds the set of backup trees that protect the service tree with minimum additional backup bandwidth requirement to the network. The notations used is given in Table 1 The Skeleton-Tree Restoration ILP Formulation is described below. The objective is to find the minimum additional backup bandwidth required for the set of backup trees. The variable z_{ij} denotes the additional backup bandwidth required on link (i, j).

$$\min\left(\sum_{(i,j)\in E} z_{ij}\right) \tag{1}$$

$$\sum_j x_{ij}^{sd} - \sum_j x_{ji}^{sd} = 0, \text{ for } i \neq h, d \ ,\forall d \in D_s \ ,\forall s \in FS_m \tag{2}$$

$$\sum_j x_{hj}^{sd} - \sum_j x_{jh}^{sd} = 1, \forall d \in D_s \ ,\forall s \in FS_m \tag{3}$$

$$\sum_j x_{dj}^{sd} - \sum_j x_{jd}^{sd} = -1, \forall d \in D_s \ ,\forall s \in FS_m \tag{4}$$

$$\sum_j y_{ji}^{s} \leq 1, \forall i \in V - \{h\} \ ,\forall s \in FS_m \tag{5}$$

$$\sum_j y_{jh}^{s} = 0, \forall s \in FS_m \tag{6}$$

$$x_{ij}^{sd} \leq y_{ij}^{s} \leq 1, \forall (i,j) \in E \ ,\forall d \in D \ , \forall s \in FS_m \tag{7}$$

$$x_{ij}^{sd} = 1, \text{ If } (i,j) \in E_m^d \ ,(i,j) \notin E_s \ ,\forall d \in D - D_s \forall s \in FS_m \tag{8}$$

$$x_{ij}^{sd} = 0, \text{ If } (i,j) \notin E_m^d \ ,(i,j) \notin E_s \ ,\forall d \in D - D_s \forall s \in FS_m \tag{9}$$

$$z_{ij} \geq \tau_{ij}^{s}(y_{ij}^{s}), \forall (i,j) \in E \ , \forall s \in FS_m \tag{10}$$

$$z_{ij} \leq A_{ij}, \forall (i,j) \in E \ . \ x_{ij}^{sd}, y_{ij}^{s} \in \{0,1\} \ . \ z_{ij} \geq 0 \tag{11}$$

The link capacity constraints and the non-zero additional backup bandwidth constraints are defined in equation 11.For each failure, we find a multicast backup tree that connects to all the destinations. To create a multicast tree, we first establish a path between the source and each of the destinations affected in the failure scenario. Equations 2, 3, and 4 are for flow conservation to establish the paths. The variable x_{ij}^{sd} determines whether link (i,j) is used on the path to destination d for backup tree in failure scenario s. Variable y_{ij}^{s} determines whether link (i,j) is used on the backup tree in failure scenario s. Equation 7 forces the backup tree to include the links used in the individual paths. Equations 5 and 6 enforce the tree constraint where there can only be one incoming links used for each node while for the source node, no incoming links are used. These equations are necessary to prevent loops cause by links that can be used with zero cost. Skeleton-Tree Restoration forces the unaffected part of the service tree to be reused. Equations 8 and 9 enforces this requirement. The cost for using link (i,j) in failure scenario s is defined by τ_{ij}^{s}.

$$\tau_{ij}^{s} = \begin{cases} 0, & \text{if } (i,j) \notin E_s \text{ and } ((i,j) \in E_m \text{ or } C_{ij}^{s} + b \leq C_{ij}) \\ b - C_{ij} + C_{ij}^{s}, & \text{if } (i,j) \notin E_s \text{ and } C_{ij}^{s} + b > C_{ij} \\ & \text{and } A_{ij} \geq b - C_{ij} + C_{ij}^{s} \\ \infty, & \text{otherwise.} \end{cases}$$

The first case is when the link can be used with zero cost because of path-intermixed or there is enough backup bandwidth for sharing. The second case occurs when there is not enough backup bandwidth on cover the all of the requested bandwidth. Thus the link cost is the additional bandwidth that needs to be allocated to cover the whole request. The final case occurs if the link being considered is down or there is not enough available bandwidth on this link to satisfy the request.

Backup trees for different failure scenarios can share backup bandwidth with each other thus equation 10 allows the backup bandwidth to overlap. The ad-

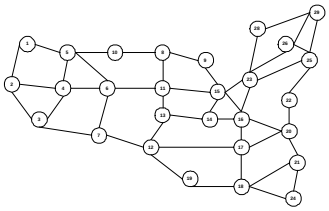

Fig. 1. Network1: US Long-Distance

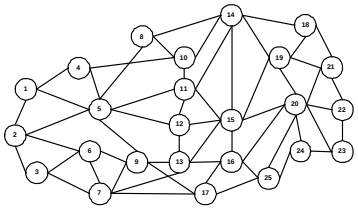

Fig. 2. Network2: Toronto Metropolitan

ditional backup bandwidth required on each link is thus the maximum of the additional backup bandwidth required by the backup trees being calculated.

4 Experimental Setup

For the simulations, two network topologies are used and are shown in Figures 1 and 2. Network1 and Network2 [17] are networks taken from practical topologies. Without loss of generality in all networks, all links are bi-directional and the link weights are set to 1. For all experiments, the results are collected from the average of results from 10 different request sets. Each request set has a 1000 multicast connection setup requests. Requests are randomly generated and have uniform bandwidth demand of one unit each. The number of destinations per request is uniform within a request set. The number of destinations is varied across simulation runs. The number of destinations will vary from 2 to 8 in increment steps of 2. When the number of destinations reaches 10 or more, the simulation run-time for each request set goes for over 48 hours for the Skeleton-Tree ILP algorithm. Thus we limit the number of destinations to 8. The ILP formulations in the experiments are solved using CPLEX 8.1 and the heuristics algorithms are implemented using C++. The experiments are executed on a PIII 1Ghz Linux Server with 1GB of RAM.

There are two types of experiments, unlimited link capacity scenario, and limited link capacity scenario. Unlimited capacity experiments are used to show the bandwidth efficiency of the algorithms. This is important when the multicast connections share the network with other traffic. Limited capacity experiments show how much requests can be admitted in the network given limited resources allocated for provisioning multicast connections. For the limited link capacity scenario all the links in the network will have the same capacity for simplicity. The number of destinations is fixed to 6 in the limited capacity experiments. The link capacity for each network topology is different due to the size and density of the network. Network1's link capacity varies from 0 to 450 units with incremental steps of 50 units. Network2's The link capacity varies from 0 to 250 units with incremental steps of 50 units.

Two metrics are used for analysis of results: Total bandwidth, and Number of Requests Accepted. Total bandwidth refers to the sum of the bandwidth required

by all the service tree and all the backup paths/trees. Number of Requests Accepted is the number of requests that can be admitted into the network within the resource constraints. A request is blocked if the algorithm cannot find a feasible service tree or the set of feasible backup paths/trees required.

5 Simulation Results Part I

The first set of simulation results compare between three solutions: Optimal Path-Pair [14], Line Restoration [15], and Skeleton-Tree Restoration. Figures 3-4 show the total bandwidth requirements from the unlimited capacity experiments. Simulations for Skeleton-Tree Restoration only go up to multicast group size of 8 for Network1 and Network2. Going beyond group size of 8 increases the computation time exponentially. Results show that Skeleton-Tree Restoration has the lowest total bandwidth requirement, and is up to 10% less than Line Restoration in Network1 and 6% less in Network2. Note that Skeleton-Tree Restoration uses the same algorithm to compute the service tree as Line Restoration, thus the difference between the two solutions is the restoration strategy used. Compared with Skeleton-Tree Restoration, Line Restoration uses up to 26% more backup bandwidth in Network1 and 21% more in Network2. Thus Skeleton-Tree Restoration is a more bandwidth efficient strategy as compared with Line Restoration.

Line Restoration retains multicast efficiency by restoring the service tree and maximize reuses of the service bandwidth, however, it does not give the restoration algorithm enough freedom to spread out the backup bandwidth. The line segment will always be restricted to be near the link failure. Skeleton-Tree Restoration gives the algorithm complete freedom to choose the way the affected destinations join back the skeleton tree. Comparing Skeleton-Tree Restoration with Optimal Path-Pair, Skeleton Restoration uses up to 31% less total bandwidth in Network1 and 40% less in Network2. The difference is from the more efficient service tree algorithm used in Skeleton-Tree Restoration and also the efficiency gained from backup bandwidth sharing between backup trees.

Figure 5-6 show the number of requests accepted from the limited capacity experiments. Results show that Skeleton-Tree Restoration and Line Restoration perform within 1% of each other in Network1. In Network2, Skeleton-Tree Restoration accepts up to 4% more requests than Line Restoration. The reason why these two algorithms perform so close to each other is because the main factor contributing to the blockings is the service tree algorithm. That is, nearly all the blockings is from failing to find a feasible service tree. Although the backup bandwidth is minimized by the strategies, the service tree algorithm will eventually utilize the resources of the links in the congested areas (hot-spots). Thus the links on the hot-spots will always be fully utilized under all the algorithms. Note that finding a feasible service multicast tree is much harder than a unicast service path as the number of links required to set up a multicast tree is significantly higher (need to connect to 6 destinations in the simulations). Thus multicast service tree algorithm is observed to impose a higher impact on blockings than the backup restoration strategies because the service bandwidth contributes to

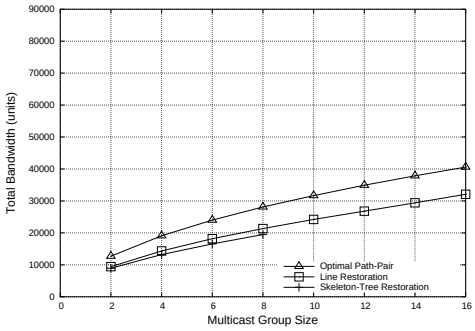

Fig. 3. Total Bandwidth Requirement for Network1

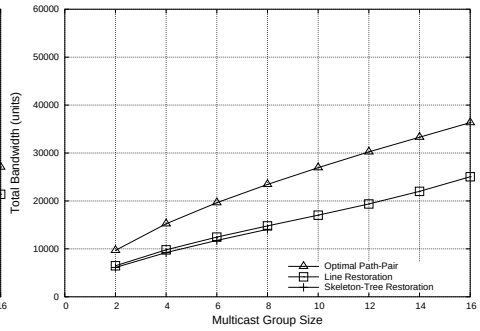

Fig. 4. Total Bandwidth Requirement for Network2

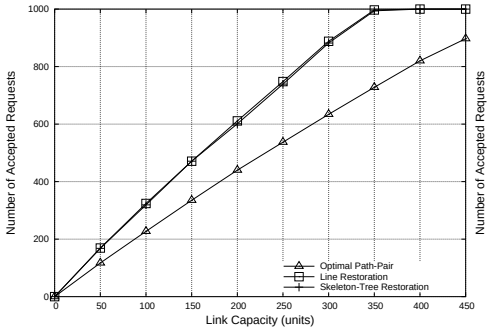

Fig. 5. Number of Request Accepted for Network1

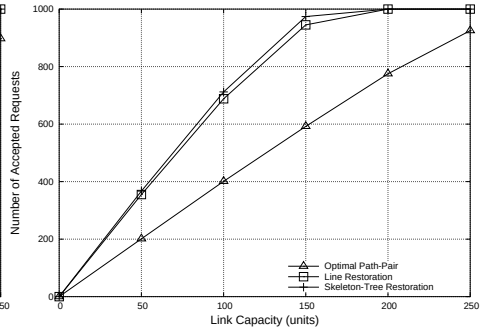

Fig. 6. Number of Request Accepted for Network2

most of the total bandwidth requirement. Comparing Skeleton-Tree Restoration with Optimal Path-Pair, Skeleton-Tree Restoration has up to 45% more accepted requests in Network1 and 103% more in Network2. The bandwidth efficiency of Skeleton-Tree Restoration allows substantially more requests to be accepted.

6 Heuristic Restoration Algorithm

Integer Linear Programming based algorithms are computationally expensive and are known to have exponential worst case times. The Skeleton-Tree ILP algorithm can take hours to compute for each set of 1000 requests with multicast group size of 8, and can go beyond 48 hours for group sizes of 10 and over. This is definitely not desirable for online-based solutions that require on-demand response time. Therefore, we propose an approximation algorithm for the Skeleton-Tree ILP algorithm. This algorithm is called *Approximate Multicast Restoration Algorithm* (AMRA) described in Algorithm 1. The same notation is used as shown in Table 1, however, we use l to represent a link rather than (i, j).

Data : Network Graph: $G(V, E)$; Failure scenario set: FS; Service tree: t;
Result : New Network Graph: $G(V, E)$; Backup tree set: BT; Failure scenario set: FS;
Initialize $BP = \emptyset$;
Find the set of failures FS_t affecting the Service tree;
for *each* $s \in FS_t$ **do**
  Find the set of failed links E_s in the current failure scenario;
  Remove all failed links $l \in E_s$ from $G(V, E)$;
  Find the skeleton of the service tree $tree_{skeleton}$;
  Recompute the network links cost τ_l^s;
  Find the list of affected destinations D_s;
  for *each* $d \in D_s$ **do**
    If d is already traversed in $tree_{skeleton}$ then skip to next destination;
    Find the shortest path p_d from any node on $tree_{skeleton}$ to d;
    $tree_{skeleton} = p_d \cup tree_{skeleton}$;
    For all links used in p_d, change the link cost to ∞;
  end
  Add $tree_{skeleton}$ to BT;
  Update $G(V, E)$ with the new $tree_{skeleton}$;
  Insert failed links back into $G(V, E)$;
end

Algorithm 1: Approximate Multicast Restoration Algorithm

The input to the algorithm is the network resource state $G(V, E)$, the list of failure scenarios FS, and the service tree t. The expected output of the algorithm is the set of backup trees BT, and the updated network resource state. First, we find the list of failure scenarios that affect the service tree FS_t. For each of the failure scenarios, we remove the failed links (E_s) and then we calculate a backup tree that restores traffic to all destinations. The logic is to first form the skeleton of the backup tree from the unaffected portion of the service tree (the links that are used to connect the source to the unaffected destinations). We then find the list of affected destinations D_s. The special case of the skeleton tree is when all the destinations are affected thus the skeleton tree only includes the source node. We then go through each affected destination sequentially and find the least cost path that connects the skeleton tree back to the destination node. This path is then added to the skeleton tree and the links used in the path will now have infinite cost (prevent loops). This process continues until all the destinations are connected to the skeleton tree. This skeleton tree then becomes the new backup tree. The algorithm ends when we find the set of backup trees that protect the service tree from all failure scenarios.

To find the shortest path from the skeleton-tree to a destination, we use the algorithm described in Algorithm 2. This is essentially just a loop that goes through all the nodes in the skeleton tree (V_t) and finds the shortest path from the nodes to the destination. The shortest path with the least cost is returned. The cost for using link l during failure s is given by τ_l^s:

$$\tau_l^s = \begin{cases} \infty, & \text{if } l \in E_{skeleton} \\ 0, & \text{if } l \notin E_s \text{ and } l \in E_m \text{ and } l \notin E_{skeleton} \\ 0, & \text{if } l \notin E_s \text{ and } C_l^s + b \leq C_l \\ b - C_l + C_l^s, & \text{if } l \notin E_s \text{ and } C_l^s + b > C_l \\ & \text{and } A_l \geq b - C_l + C_l^s \text{ and } l \notin E_{skeleton} \\ \infty, & \text{otherwise.} \end{cases}$$

```
Data      : Network Graph: G(V, E); Skeleton tree: t; destination: d;
Result    : Least cost path: p;
path_min = 0;
for each v ∈ V_t do
    Find the shortest path p_new from v to d;
    if cost of p_new < path_min then
        p = p_new;
    end
end
```

Algorithm 2: Shortest Path to Skeleton Tree

The first case is for preventing the links ($l \in E_{skeleton}$) already on the skeleton tree from being re-used. This stops loops formation. The second case is for links that belong to the service tree that can be used for path intermix. These links must not be part of the current skeleton tree and is not one of the failed links. Path intermix allows these links to be used without additional cost. The third case is when there is enough backup bandwidth to cover the request in addition to what has already been used by this failure scenario. The fourth case is where the backup bandwidth can only cover part of the requested bandwidth. Thus the cost for using this link will be the additional backup bandwidth needed to cover the rest of the requested bandwidth. The final case is when sufficient bandwidth is not available on the link to cover the additional bandwidth requirement.

The time-complexity of AMRA is $O(|FS_t|(|D||V|^2 \log |V| + |E|))$ (shortest path $O(|V| \log |V|)$).

7 Simulation Results Part II

ARMA approximates the Skeleton-Tree ILP Restoration algorithm but does not compute the service tree. We use an existing approximation algorithm (Nearest Neighbor First) [12] for calculating the service tree. Both algorithms are polynomial-time thus the combination is a polynomial-time solution.

Simulations in this section compare our proposed heuristic algorithm with the ILP based algorithm and another existing heuristic algorithm: Skeleton-Tree Restoration ILP, AMRA, and Line Restoration Approximation (LRA) [12]. The first set of simulations is for unlimited link capacity networks with varying multicast group sizes. The total bandwidth requirement results are presented in Figures 7-8. Compared with Skeleton-Tree Restoration, AMRA requires up to 4% more bandwidth in Network1 and 10% more in Network2. The difference between the two solutions is mainly from the higher service bandwidth requirement. This is because Skeleton-Tree Restoration uses an ILP algorithm that finds the optimal least cost service tree whereas AMRA uses an approximation algorithm only. Comparing with the other heuristic solution, LRA requires up to 15% more bandwidth than AMRA in Network1 and 14% more in Network2. Since AMRA and LRA use the same service tree algorithm, the results show the backup bandwidth efficiency of the skeleton-tree restoration strategy as compared to line restoration strategy.

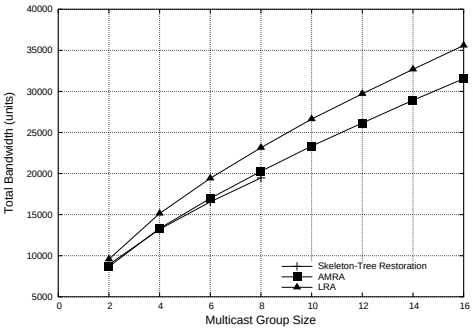

Fig. 7. Total Bandwidth Requirement for Network1

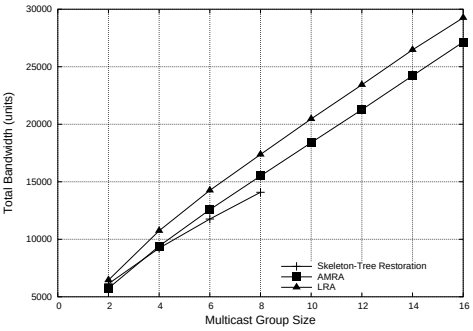

Fig. 8. Total Bandwidth Requirement for Network2

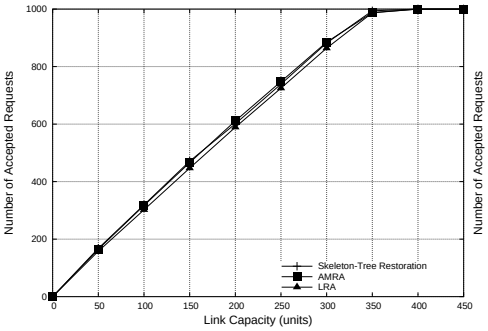

Fig. 9. Number of Request Accepted for Network1

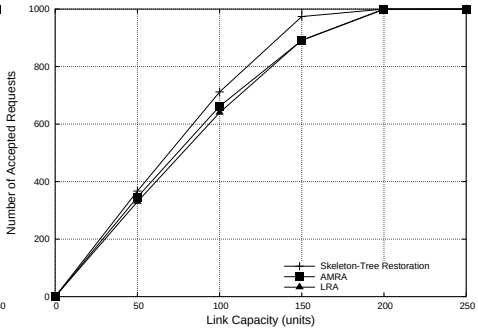

Fig. 10. Number of Request Accepted for Network2

The second set of simulations is for limited capacity experiments where the multicast group size is fixed to 6 destinations while the link capacity varies. The results are shown in Figures 9-10. The results show that the three solutions perform very similarly in terms of the acceptance rate. AMRA performs about the same as Skeleton-Tree Restoration for Network1 (within 1%). For Network2, the acceptance rate for AMRA reaches within 9% of Skeleton-Tree Restoration. Compared with LRA, AMRA has up to 4% higher acceptance rate in Network1 and up to 5% higher in Network2. The main cause (nearly all) of request blocks for the algorithms is from failing to find a feasible service tree. Skeleton-Tree Restoration uses an ILP algorithm that will always find a feasible service tree if one exists. Thus they have slightly more acceptance rate than the heuristic algorithms, but the hot-spots cause all the solutions to block most requests at network saturation point.

8 Conclusion

In this paper, we proposed a full ILP-based solutions (Skeleton-Tree Restoration) and a complete heuristics solution (AMRA) based on a new restoration strategy. Simulation results show that using any of these solutions will improve bandwidth efficiency substantially over other existing solutions (Optimal Path-Pair and LRA). Bandwidth efficiency for AMRA is within 10% of the full ILP-based solution and the request acceptance rate is within 12%. The heuristics solution has polynomial-time complexity and is more scalable to larger networks and multicast group sizes than the ILP-based solution.

References

1. FastWeb: TV Over FastWeb. http://www.fastweb.it/ (2004)
2. now.com.hk: Pay TV Over Broadband. http://www.now.com.hk/ (2004)
3. TV, S.P.: Terrestial Broadcast. http://www.skyperfectv.co.jp/skycom/e/ (2004)
4. Moy, J.: Multicast Extensions to OSPF. IETF RFC 1584 (1994)
5. Ballardie, A.: Core Based Trees (CBT). IETF RFC 2201 (1997)
6. Chu, Y., Rao, S., Seshan, S., Zhang, H.: Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture. In: Proceedings of ACM SIGCOMM, San Diego, USA (2001)
7. Banerjee, S., Kommareddy, C., Kar, K., Bhattacharjee, B., Khuller, S.: Construction of an Efficient Overlay Multicast Infrastructure for Real-Time Applications. In: Proceedings of IEEE INFOCOM, San Francisco, USA (2003)
8. Rosen, E., Viswanathan, A., Callon, R.: Multiprotocol Label Switching Architecture. IETF RFC 3031 (2001)
9. Berger, L.: Generalized Multi-Protocol Label Switching (GMPLS) Signaling Functional Description. IETF RFC 3471 (2003)
10. Yasukawa, S.: Requirements for Point to Multipoint extension to RSVP-TE. IETF Internet Draft draft-ietf-mpls-p2mp-requirement-01.txt (2004)
11. Markopoulou, A., Iannaccone, G., Bhattacharyya, S., Chuah, C., Diot, C.: Characterization of Failures in an IP Backbone. In: Proceedings of IEEE INFOCOM, Hong Kong, China (2004)
12. Kodialam, M., Lakshman, T.: Dynamic Routing of Bandwidth Guaranteed Multicasts with Failure Backup. In: Proceedings of IEEE ICNP, Paris, France (2002)
13. Medard, M., Finn, S., Barry, R., Gallenger, R.: Redundant Trees for Preplanned Recovery in Arbitary Vertex-Redundant or Edge-Redundant Graphs. IEEE/ACM Transactions on Networking **7(5)** (1999)
14. Singhal, N., Sahasrabuddhe, L., Mukherjee, B.: Provisioning of Survivable Multicast Sessions Against Single Link Failures in Optical WDM Mesh Networks. IEEE Journal of Lightwave Technology **21(11)** (2003)
15. Lau, W., Jha, S.: Multicast Resilient Connections with Quality of Service Guarantees. UNSW Technical Report 0408 (2004)
16. Lau, W., Jha, S.: Failure-Oriented Path Restoration Algorithm for Survivable Networks. eTransactions on Network Service and Management (IEEE Communications Society) **1(1)** (2004)
17. Xiong, Y., Mason, L.: Restoration Strategies and Spare Capacity Requirements in Self-Healing ATM Networks. IEEE/ACM Transactions on Networking **7(1)** (1999)