Multicast Resilience with Quality of Service Guarantees

William Lau*, Sanjay Jha* and Suman Banerjee[†]

* School of Computer Science and Engineering

The University of New South Wales, Sydney, NSW 2052, Australia

[†] Department of Computer Sciences

University of Wisconsin-Madison, Madison, WI 53706, USA

Email: wlau,sjha@cse.unsw.edu.au, suman@cs.wisc.edu

UNSW-CSE-TR-0408

February 2004

Abstract

This paper defines new algorithms for providing bandwidth guaranteed multicast support to applications that require resilience in presence of link failures in the network. Our techniques are applicable for networks that are capable of native network-layer multicast as well as networks that have been enhanced to support infrastructure-based overlay multicast. For efficient multicast restoration, which are necessary for such a construction, we based our techniques on online computation and dynamic routing. We define new Integer Linear Programming (ILP) solutions to the problem, and from experience with the ILP solutions, we derive heuristics that are used to form a new polynomial-time algorithm.

Results from our experiments show that our proposed mechanisms can significantly improve the bandwidth efficiency (by 55%) and request acceptance rate (by a factor of 1.5) over alternative mechanisms. The benefits are comparable for infrastructure-based overlay multicast scenarios. Our results also indicate that the proposed heuristics solution performs within 12% of the ILP formulation solution with respect to different metrics.

I. Introduction

We propose algorithms that provide bandwidth guaranteed multicast path construction as well as efficient restoration for applications with quality of service (QoS) requirements. Our algorithms will allow network and application service providers to construct resilient multicast paths for large group distribution applications. The proposed techniques can be implemented entirely in the network layer or as an overlay in the provider's network. In our work we consider bandwidth requirements of multicast applications as well as efficient failover of these bandwidth guaranteed paths in presence of failures in the network.

The basic problem we are trying to solve can be explained with a simple example. Consider a wide-area video broadcast of a popular program where clients register with the content provider for high quality data streaming. Realizing that a large group of clients have registered, the content provider requests for a relatively long-lived multicast connection from the network provider. Given the requirements, the network provider sets up the appropriate multicast tree with bandwidth guarantee, and additionally provides failover guarantee, i.e. immediate failover to a backup path (tree) in presence of network failures. The problem now is to construct a primary multicast service tree for normal traffic delivery and use restoration strategies to guarantee traffic recovery over failures within the network. This involves pre-planning the routes to use and bandwidth required when failure occurs.

We present only single link failures in this paper, and the ideas presented in this paper can be extended to single node failure with minor modifications. Single failures are the most common type of failure, and majority of protection/restoration strategies [1], [2], [3] cover single failures only.

We propose new strategies for multicast tree restoration based on online computation — we assume that multicast requests arrive at the network provider in a sequence and we provision each connection in that sequence. The multicast paths chosen for the service and the backup path take into account the previously assigned connections (multicast and unicast) and their implications on residual network capacity. Each request computation does not affect previously calculated requests and is not dependent on future requests.

We first formulate the new strategies as Integer Linear Programming (ILP) problems. While ILPs can be solved optimally, they are infeasible due to the exponential computation overheads. Evaluating and analyzing the results from the ILP solutions, we derive new heuristics that aims to improve network efficiency. Using these heuristics, we propose a new polynomial-time algorithm for both network-layer multicast as well as infrastructure-based overlay multicast.

Results from our experiments show that:

1) Our proposed techniques improve bandwidth efficiency by upto 55% and request acceptance rate by a factor of 1.5 in comparison to other alternative approaches for multicast tree restoration for network-layer multicast scenarios. We observe similar benefits for infrastructure-based overlay multicast scenarios.
2) The heuristic solution achieves performance that are fairly close to the optimal solution (achieved by the ILP formulation). For example, the total bandwidth requirement is within 10% of the ILP-based solution and the acceptance rate is within 12%.

The rest of this paper is as follows. We first present the multicast service model in Section II. The algorithms used to calculate the service tree are described in section III. The new restoration strategies are introduced in section IV, and the ILP formulations of the strategies are given in section V. The experimental setup is described in section VI and the results from the ILPs are evaluated and analyzed in section VII. A new heuristic algorithm is presented in section VIII, followed by performance analysis of the algorithm with the ILP solutions and another heuristic algorithm in section IX. The heuristic restoration algorithm is extended to work for overlay-based multicast support network environments in section X and the results are presented in section XI. Related works are discussed in section XII and then the paper concludes in Section XIII.

II. Multicast Service Model

A multicast network will carry traffic for a variety of applications. In this paper, we consider multicast connections that are relatively long lived and do not have rapid changing group membership. These connections have much shorter life-time than traditional unicast leased lines but have much longer life-time than the logical TCP connections. The network provider will be required to provision some bandwidth for these relatively long lived multicast sessions using centralized or distributed server architectures. The types of applications that fit in this category are video conferencing, multimedia broadcast (e.g. cable TV. The membership and bandwidth requirements for such applications are relatively static and the life-time of the multicast connections required are relatively long (one hour or more).

Other types of multicast connections that are more dynamic or does not require bandwidth restoration guarantees are best supported using other fully distributed best-effort architectures [4], [5], [6], [7], [8]. Recently several two-tier overlay architectures have been proposed as a possible application layer solution for providing multicast services across the Internet [8], [5]. The Overlay Multicast Network Infrastructure (OMNI) [8] consists of a set of devices called Multicast service Nodes (MSNs) distributed in the network and provides efficient data distribution services to a set of end hosts. An end-host (client) subscribes with a single MSN to receive multicast data service. MSNs use a distributed algorithm to organize themselves to form a multicast delivery backbone. The data delivery path used in the overlay backbone is independent of the data delivery path used by the end-host to the MSN. Algorithms and mechanisms proposed in this work can be implemented in each network, either in network layer or as infrastructure based overlay e.g. OMNI.

The restoration model used in this work covers single link failure. The objective is to provide recovery along with bandwidth guarantees (as per original Service Level Agreement) for multicast connections. For a single domain (network provider) case, we assume appropriate traffic engineering support based on network mechanisms such as MPLS [9] and signaling protocols, such as RSVP-TE [10].

The traditional approach to restoration strategies are usually offline computation where all the requests are assumed to be available at the same time and computed together. Although offline algorithms can potentially achieve better bandwidth efficiency by taking all the requests into account, it is not suitable for on-demand multicast connections due to the high computation complexity. Therefore, we take the online computation approach.

III. Multicast Service Tree Construction Algorithms

Our basic strategy to provide quick failover of multicast connections in presence of network failures is to construct a basic multicast service tree, and appropriate backup path(s) that protects the service tree against potential single link failures in the network. Like restoration techniques proposed by other researchers in the context of bandwidth guaranteed unicast paths [11], [12], we assume that multiple simultaneous failures in a given network is fairly unlikely and hence ignored. We use separate algorithms for calculating the service multicast tree and the backup multicast tree. We first compute an appropriate service tree and subsequently compute efficient paths to restore traffic during a single link failure. An alternative approach would be to jointly compute the service and the backup paths. However, calculating the optimal least cost service tree is NP-Hard, and hence a joint computation would only increase the complexity of this problem.

In this section, we describe the algorithms used to compute the multicast service tree. We have defined an ILP to compute the optimal least cost multicast service tree , but ILP-based solution is a not viable for on-demand applications due to high computation costs. The ILP formulation also does not work for infrastructure overlay multicast networks.

Based on our experience with this formulation we propose a polynomial-time heuristic algorithm to compute the multicast service tree, and this algorithm also works for infrastructure overlay multicast networks. We call this heuristic algorithm, *Multicast Logical Tree Approximation* (MLTA) shown in Algorithm 1.

A. ILP Formulation for Multicast Service Tree

First, we introduce a new ILP formulation used to compute the least cost service multicast tree.

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

$$\sum_{j\in V} x^d_{ij} - \sum_{j\in V} x^d_{ji} = 0, \text{ for } i \in V - \{h,d\} \text{ ,}\forall d \in D \tag{2}$$

$$\sum_{j\in V} x^d_{hj} - \sum_{j\in V} x^d_{jh} = 1, \forall d \in D \tag{3}$$

$$\sum_{j\in V} x^d_{dj} - \sum_{j\in V} x^d_{jd} = -1, \forall d \in D \tag{4}$$

$$x^d_{ij} \le y_{ij} \le 1, \forall d \in D \text{ , } \forall (i,j) \in E \tag{5}$$

$$b \times y_{ij} \le A_{ij}, \forall (i,j) \in E \tag{6}$$

$$x_{ij}, y_{ij} \in \{0,1\} \tag{7}$$

$$\tag{8}$$

Consider the cost of using link (i,j) is denoted by c_{ij} (normally link weight times the bandwidth b required) and the decision variable y_{ij} determines if the link is used in the multicast tree or not. The objective function (equation 1) is thus to minimize the cost of the multicast tree that connects the source h to the set of destinations

D. A tree can be seen as a union of unicast paths that connects the source to destinations with the constraint that no loop is formed as a result of the union. Equations 2, 3, and 4 define the constraint for conservation of flows for the unicast paths between source and the destinations. Equation 5 produces the union of the x variables to form the multicast tree represented by the y variables. The capacity constraints are defined by equation 6 and the binary constraints for the decision variables defined by equation 7. The available bandwidth on the link is denoted by A_{ij} and the set of links denoted by E. If the cost for each link is non-zero, we claim this ILP formulation produces a least cost tree. We will attempt to prove this by contradiction. Lets assume an optimal solution (minimum) exists for the ILP formulation which includes a loop for variable y. Removing one of the segments that forms the loop will result in a reduction in the objective equation (non-zero cost links) while at the same time eliminates the loop. The solution will still be feasible as it satisfies all the constraints required (reachability is retained). Therefore, we have found a feasible solution that has lower cost than the optimal solution which contradicts with the initial assumption. Thus an optimal solution in this ILP formulation cannot contain any loops, and must form a tree. The tree by definition of the objective function must be the least cost solution with the constraint that the source is connected to all the destinations.

B. Heuristic Algorithm for Multicast Service Tree

There are many approximation algorithms [13], [14] for the least cost multicast tree problem, however, they do not work for infrastructure overlay multicast networks. We propose a new approximation algorithm for the least cost multicast tree problem that also works for the infrastructure overlay multicast networks.

```
Data    : Network Graph: $G(V, E)$; Request $r$;
Result  : New Network Graph: $G(V, E)$; Logical Tree $t$;
Set cost of all links $l$ that have $A_l < b$ to $\infty$;
Initialize $CON = \{h\}$;
for each $d \in D$ do
    If $d \in CON$ skip to next destination;
    Find the shortest path $p_d$ from any node in $CON$ to $d$;
    $t = p_d + t$;
    Update $G(V, E)$ for using the links in $p_d$ and bandwidth availability of the links have to be re-checked;
    Find set of connected nodes $CON_{new}$ in $p_d$ and let $CON = CON_{new} \cup CON$;
end
```

Algorithm 1: Multicast Logical Tree Approximation

Let the inputs be: the network graph $G(V, E)$, and the request that includes the required information (source, destination, bandwidth). The outputs of the algorithm be: the logical tree, and the updated resource state of the network graph. V denotes the set of vertexes and E the set of edges. The request is to build a logical tree from source node h to a set of destination nodes D. The general idea of MLTA is to create a logical multicast tree. A logical tree is a connection of logical nodes made up by the source node, the destination nodes, and the Multicast

Service Nodes (MSNs). The logical links of the tree consists of a physical path between the logical nodes. Two logical links can traverse the same physical link.

We first set all links l without enough available bandwidth A_l to have infinite cost. The requested bandwidth is denoted by b. The source node h is assumed to be a MSN and is by default the only MSN in the set of *connected* nodes (CON) initially. For each destination, if the destination is part of the tree already and is a MSN ($d \in CON$) then nothing more needs to be done for this destination node. Otherwise, we find the shortest path from each connected node to the destination. The least cost path is chosen and is included in the logical tree. Only the MSN nodes that the path traversed will be added to the set of connected nodes. The resource used by the path is updated in the network resource state. We also need to check the availability of bandwidth of these links used on the path after the resource allocation. If the available bandwidth for a link is less than b then this link cannot be used by the logical tree anymore, hence the cost of the link is set to infinity. Otherwise the cost is set to b. The logical tree is completed after all the destination nodes are connected to the tree. This algorithm tries to exploit the MSN's multicast capability to reduce bandwidth usage as much as possible. The time-complexity of this algorithm is $O(|D||M||V|log|V| + |E|)$ where M denotes the set of MSNs in the network. Polynomial-time algorithms like MLTA scales better than ILP-based solutions for large networks with large multicast sizes. Additionally our results in Section VII show that the performance of this algorithm is fairly close to the optimal solution obtained by the ILP for representative scenarios.

IV. Multicast Restoration Strategies

Give the service tree, the only task left is to provide restoration strategies that guarantee traffic recovery during single link failures. In this paper, we use the state-dependent restoration strategies. In state dependent strategies, the route that traffic takes from source to destination is dependent on the failure state of the network. Thus for different link failures, the route taken to restore traffic from source to the set of destinations can be different.

In the state independent approach, the route used to restore traffic from source to a destination must be disjoint (link disjoint in our case) from the route used in the service tree. This is not as flexible as the state-dependent approach where the only mandatory requirement for the backup route is that it must not traverse the failed components of the failure. We choose to use state-dependent due to this flexibility.

Firstly, we define a simple restoration strategy that is used for comparison with more sophisticated restoration strategies. We call this the *Simple Restoration*. Simple Restoration uses unicast connections to restore traffic from the source to the destinations affected by the failure. Thus for each possible failure scenario, we calculate a set of unicast connections that restores traffic. Assuming single link failures means that only one failure scenario can occur at any one time, the backup connections that restore traffic in one failure scenario can use the same backup bandwidth as backup connections in other failure scenarios. The sharing of the backup bandwidth is the main motivation for Simple Restoration.

An example is used to illustrate how Simple Restoration behave. Suppose we have a service tree as shown in figure 1. Node A is the source and nodes F, G, E are the destinations. The dark line indicates the service tree

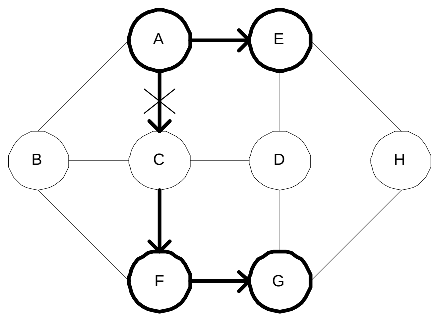

Fig. 1. Example: Service tree with link failure

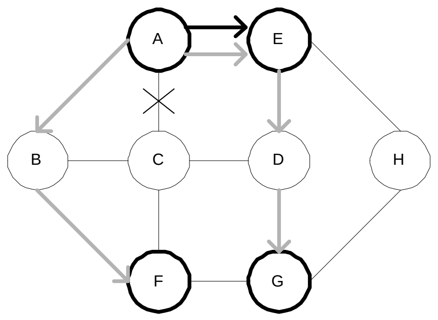

Fig. 2. Example: Simple Restoration

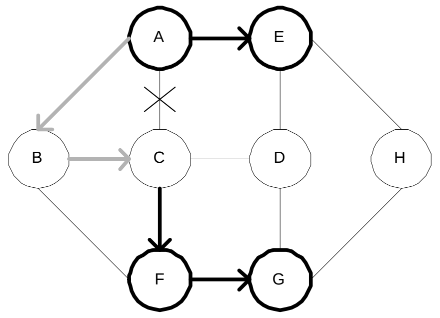

Fig. 3. Example: Line Restoration

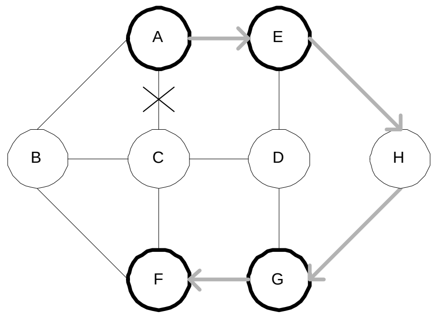

Fig. 4. Example: Whole-Tree Restoration

branches. The link between A and C fails and is indicated by the cross on the link. One possible solution using Simple Restoration is given in figure 2. Since E is not affected by the failure, it continues to be served by the original service tree, which is indicated by the black arrow lines. For each of the destinations (F and G) affected by the failure, a unicast path between the source and the destination is established to recover the traffic, which is indicated by the grey arrow lines.

Another known restoration strategy is based on *Line Restoration* [3]. Line Restoration re-connects the service tree around the failed components. For link failure, it finds another path segment that connects from the start of the failed link (preceding node) to the end of the failed link (succeeding node). The motivations of Line Restoration are:

1) Reuses the service tree's resource for restoring traffic.
2) Retains multicast efficiency when restoring traffic.

However, many related works [15], [16], [17], [18] on the restoration in the context of unicast shows that line restoration is inferior to path restoration because path restoration provide more routing flexibility for restoring traffic between the source and destination (Line restoration confines the restoration segment to be near the failed components). Results in section VII show that line restoration in multicast performs better than Simple Restoration that uses (unicast) path restoration. This is mainly due to the motivations mentioned above.

For Line Restoration, the solution for the example is illustrated in figure 3. The link between nodes A and C fails thus the solution is to connect the nodes back to restore traffic flow. A line segment from A to B to C is used

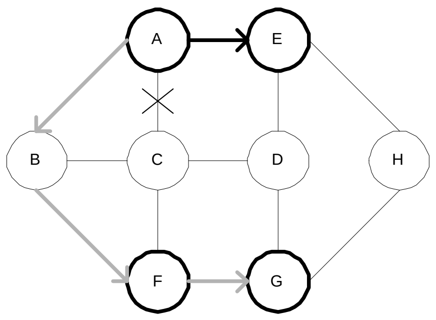

Fig. 5. Example: Sub-Tree Restoration

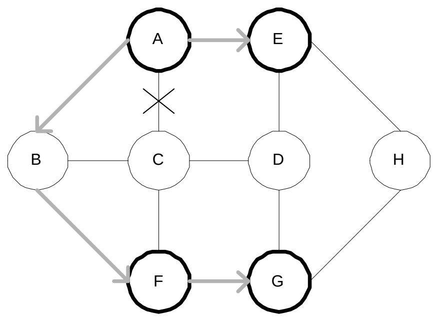

Fig. 6. Example: Skeleton Restoration

in this case, which is shown by the grey arrow lines. The rest of the service tree remains the same as indicated by the black arrow lines.

For retaining multicast efficiency but also providing flexibility to restore traffic, we propose new restoration strategies based on a new approach we call *Tree Restoration*. Tree Restoration uses a new backup multicast tree to restore traffic during failure. This differs to Line Restoration that fix the service tree during the failure. We propose three variations of the Tree Restoration approach:

1) Whole-Tree Restoration: Recompute a whole new backup tree for each failure.
2) Whole-Tree with Path Intermix Restoration: Recompute a whole new backup tree for each failure using the path intermix [19] concept.
3) Sub-Tree Restoration: Recompute a new tree for the destinations disconnected from the service tree due to the failure.
4) Skeleton-Tree Restoration: a new backup tree that reuses the unaffected portion of the service tree (the skeleton of the new tree), and connects the affected destinations to the skeleton tree.

Whole-Tree Restoration has complete flexibility in finding a new tree while retaining multicast efficiency. Whole-Tree Restoration requires traffic to be switched from the service tree to the backup tree during failure. This will cause traffic disruption for all destinations when switching. The Whole-Tree Restoration solution for the example is shown in figure 4. Although destination E is not affected by the failure, E will still be part of the backup tree and will receive traffic from this tree during the failure. The new backup tree connects to all destinations as indicated by the grey arrow lines.

A method called path intermix [19] allows the service tree's bandwidth to be re-used by the backup trees that protect this service tree during failure. By applying path intermix to the Whole-Tree Restoration, the service bandwidth can be utilized and multicast efficiency retained. Path-intermix encourages links on the service tree to be used by putting zero cost on the links when computing the backup trees. This gives freedom for the algorithms to choose the links that minimize bandwidth usage. However, path intermix can only be applied to the strategies that switch traffic off the service tree during failure. The solution is essentially the same as Whole-Tree Restoration except the cost of using the links may be different.

Sub-Tree Restoration only computes a tree (sub-tree) for the destinations disconnected from the service tree during

Strategy	**Multicast**	**Reuse Service B/W**
Simple Restoration	No	Yes
Line Restoration	Yes	Yes
Whole-Tree	Yes	No
Sub-Tree	Yes	Yes
Skeleton-Tree	Yes	Yes
Whole-Tree Path Intermix	Yes	Yes

TABLE I

MULTICAST RESTORATION STRATEGIES

the failure. Sub-Tree requires traffic to be forwarded along two trees during failure. The traffic is still forwarded along the service tree while the backup tree will also be forwarded with traffic. Multicasting efficiency is retained and the service bandwidth is utilized. However, redundancy can exist due to dual tree usage during failure. The Sub-Tree solution for the example is shown in figure 5. The unaffected destinations (E) continue to receive traffic from the service tree as indicated by the black arrow lines. The affected destinations (F and G) receive traffic from a backup sub-tree shown by the grey arrow lines.

Skeleton-Tree Restoration can be seen as restricted form of Whole-Tree Restoration with path intermix. The difference is that Skeleton-Tree Restoration forces the backup tree to use the resource of the unaffected portion of the service tree. The unaffected portion of the service tree forms the skeleton for the backup tree that is yet to be completed. The algorithm then decides the best way to connect the affected destinations to the backup tree. The affected portion (not including the failed components) of the service tree can be reused with zero cost to the backup tree. The Skeleton-Tree solution for the example is illustrated in figure 6. Destination E is not affected by the failure thus the path from the source to E becomes the skeleton for the backup tree. Destinations F and G are connected back to the skeleton tree. The skeleton tree is shown in the figure 6 by the grey arrow lines. During the failure, the traffic is switched from the service tree to the backup tree.

Using a tree for backup retains the benefits of multicasting while reusing part of the service tree also reduces backup bandwidth requirement. Sharing backup bandwidth between failure scenarios reduce the backup bandwidth even more. The motivation for this paper is to study the performance between these different strategies. A summary of the different restoration strategies are given in table I.

V. ILP FORMULATIONS FOR RESTORATION STRATEGIES

This this section, we convert the restoration strategies into new ILP problems. The ILP formulations are used to evaluate and analyze the performance and behaviour of the different restoration strategies. We use this analysis to derive new heuristics that we use to develop heuristic algorithms.

TABLE II

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 in the request.
(i,j)	link connecting i to j.	b	requested bandwidth.
A_{ij}	bandwidth available in link.	R_{ij}	backup bandwidth in link.
R_{ij}^s	backup bandwidth required in (i,j) for s.	E_m^d	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_{ij}^s	cost for using (i,j) in s.

Common notations used in the ILP formulations are given in Table II. We present the ILP formulations in the following order: Simple Restoration, Line Restoration, Whole-Tree Restoration, Whole-Tree Restoration with Path Intermix, Sub-Tree Restoration, and Skeleton-Tree Restoration.

A. Simple Restoration ILP Formulation

For each failure that affects the service tree, we find the list of affected destinations and then find the least cost unicast backup path to each of these destinations. Unicast backup paths from different failure scenarios can share backup bandwidth with each other, hence, the optimal solution can be reached by computing all the unicast backup paths together in the ILP formulation.

$$\min\left(\sum_{(i,j)\in E} z_{ij}\right) \tag{9}$$

$$\sum_j x_{ij}^{sd} - \sum_j x_{ji}^{sd} = 0, \text{ for } i \neq h, d \ , \forall d \in D_s \ , \forall s \in FS_m \tag{10}$$

$$\sum_j x_{hj}^{sd} - \sum_j x_{jh}^{sd} = 1, \forall d \in D_s \ , \forall s \in FS_m \tag{11}$$

$$\sum_j x_{dj}^{sd} - \sum_j x_{jd}^{sd} = -1, \forall d \in D_s \ , \forall s \in FS_m \tag{12}$$

$$\sum_j x_{ij}^{sd} \leq 1, \forall i \in V - \{h\} \ , \forall d \in D_s \ , \forall s \in FS_m \tag{13}$$

$$\sum_j x_{jh}^{sd} = 0, \forall d \in D_s \ , \forall s \in FS_m \tag{14}$$

$$z_{ij} \geq \left(\sum_{d\in D_s} b(x_{ij}^{sd})\right) + R_{ij}^s - R_{ij}, \forall (i,j) \in E \ , \forall d \in D_s \ , \ \forall s \in FS_m \tag{15}$$

$$z_{ij} \leq A_{ij}, \forall (i,j) \in E \tag{16}$$

$$x_{ij}^{sd} = 0, \forall (i,j) \in E_s \ , \forall d \in D_s \ , \forall s \in FS_m \tag{17}$$

$$x_{ij}^{sd} \in \{0,1\} \tag{18}$$

$$z_{ij} \geq 0 \tag{19}$$

The objective function is to minimize the additional backup bandwidth requirement. Let z_{ij} represents the additional backup bandwidth required on link (i,j). For each failure scenario affecting the service tree ($s \in FS_m$), the set of affected destinations is defined by D_s. We use a unicast path to restore the connection back from the source h to the destinations $d \in D_s$. Let x_{ij}^{sd} represent that decision variable (equation 18) for whether link (i,j) should be used for failure scenario s to recover traffic to destination d. Equations 10, 11, and 12 are the flow conservation constraints for the unicast path. Equations 13 and 14 are for preventing loops in the unicast path.

Each failure scenario requires certain backup bandwidth on each link to restore traffic. The backup bandwidth allocation on each link is the maximum of all failure scenarios' backup bandwidth requirement on the link. For each failure scenario that affects the service tree, unicast paths are used to retire traffic and these paths require backup bandwidth on the links traversed. The sum of the bandwidth on link (i,j) used by these unicast paths for failure scenario s plus the current bandwidth requirement of failure s has on the link, minus the current backup bandwidth allocated on the link, gives the additional backup bandwidth requirement as a result of failure s. There are many different failure scenarios affecting the service tree so each of them might require different additional backup bandwidth on the same link. Since the failures are independent of each other, the additional backup bandwidth that needs to be allocated on the link is the maximum additional backup bandwidth out of all the failure scenarios. This is expressed by the constraint in equation 15 and equation 19 prevents negative additional backup bandwidth requirements. Equation 16 ensures that the additional bandwidth requirement must be within the bandwidth available on the link. Equation 17 prevents any of the failed links (E_s) from being used.

B. Line Restoration ILP Formulation

Line restoration aims to reconnect the service tree around the failure. For the optimal solution, the ILP formulation computes all the line segments protecting each possible failure affecting the service tree together.

$$\min \left(\sum_{(i,j)\in E} z_{ij} \right) \tag{20}$$

$$\sum_j x^s_{ij} - \sum_j x^s_{ji} = 0, \text{ for } i \neq k, n \ , \forall s \in FS_m \tag{21}$$

$$\sum_j x^s_{kj} - \sum_j x^s_{jk} = 1, \forall s \in FS_m \tag{22}$$

$$\sum_j x^s_{nj} - \sum_j x^s_{jn} = -1, \forall s \in FS_m \tag{23}$$

$$\sum_j x^s_{ij} \leq 1, \forall i \in V - \{k\} \ , \forall s \in FS_m \tag{24}$$

$$\sum_j x^s_{jk} = 0, \forall s \in FS_m \tag{25}$$

$$z_{ij} \geq \tau^s_{ij}(x^s_{ij}), \forall (i,j) \in E \ , \forall s \in FS_m \tag{26}$$

$$z_{ij} \leq A_{ij}, \forall (i,j) \in E \tag{27}$$

$$x^s_{ij} \in \{0,1\} \tag{28}$$

$$z_{ij} \geq 0 \tag{29}$$

This IP formulation is similar to Simple Restoration except the path is not from source to affected destinations, but from the node preceding the failure to the nodes succeeding the failure. For single link failure, suppose the failed link on the service tree is (k, n) thus the preceding node is k and succeeding node is n. The objective function (equation 20) minimizes the additional backup bandwidth requirement in the network. Equations 21, 22, and 23, are for flow conservation for the path segment that restores the traffic. Equations 24 and 25 are for preventing loops in the path segment.

The additional backup bandwidth required in each link is the maximum of the additional backup bandwidth caused by the failure scenarios, this is expressed in equation 26. The cost of using a link in failure scenario s is given by τ^s_{ij}:

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

The first case occurs when a link can be used with no additional cost because there is enough backup bandwidth on the link to accommodate the backup bandwidth already required by this failure scenario with addition of the current requested bandwidth (b). The second case occurs when the backup bandwidth on the link cannot cover all of the requested bandwidth, thus additional bandwidth needs to be allocated to the link. This additional bandwidth required becomes the cost for using this link. 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. Equation 29 ensures that the additional backup bandwidth requirement cannot be negative and equation 27 ensures the additional bandwidth required is within the link capacity limits.

C. Whole-Tree ILP Formulation

We define a new ILP formulation for the Whole-Tree strategy proposed in this paper. The logic is to compute a whole new backup tree that connects to all destinations for every failure that affects the service tree. The optimal solution would be reached by computing these trees together in the same ILP formulation.

$$\min \left(\sum_{(i,j) \in E} z_{ij} \right) \tag{30}$$

$$\sum_j x_{ij}^{sd} - \sum_j x_{ji}^{sd} = 0, \text{ for } i \neq h, d \ , \forall d \in D \ , \forall s \in FS_m \tag{31}$$

$$\sum_j x_{hj}^{sd} - \sum_j x_{jh}^{sd} = 1, \forall d \in D \ , \forall s \in FS_m \tag{32}$$

$$\sum_j x_{dj}^{sd} - \sum_j x_{jd}^{sd} = -1, \forall d \in D \ , \forall s \in FS_m \tag{33}$$

$$x_{ij}^{sd} \leq y_{ij}^{s} \leq 1, \forall (i,j) \in E \ , \forall d \in D \ , \ \forall s \in FS_m \tag{34}$$

$$\sum_j y_{ji}^{s} \leq 1, \forall i \in V - \{h\} \ , \forall s \in FS_m \tag{35}$$

$$\sum_j y_{jh}^{s} = 0, \forall s \in FS_m \tag{36}$$

$$z_{ij} \geq \tau_{ij}^{s}(y_{ij}^{s}), \forall (i,j) \in E \ , \ \forall s \in FS_m \tag{37}$$

$$z_{ij} \leq A_{ij}, \forall (i,j) \in E \tag{38}$$

$$x_{ij}^{sd}, y_{ij}^{s} \in \{0,1\} \tag{39}$$

$$z_{ij} \geq 0 \tag{40}$$

The objective is to find the minimum additional backup bandwidth required for the set of backup trees. For each failure affecting the service tree, we try to find a multicast tree that connects to all the destinations. A multicast tree can be as a union of individual unicast paths that connect from source to the destinations. The flow conservation constraints for these unicast paths are expressed in equations 31, 32, and 33. The union of the unicast path is expressed in equation 34 and the constraints for ensuring a tree structure is expressed in equations 35 and 36.

The cost for using link (i, j) in failure scenario s is defined by τ_{ij}^{s}, this is the same equation as the one defined in section V-B. Backup trees for different failure scenarios can share backup bandwidth with each other thus equation 37 allows the backup bandwidth to overlap. The additional backup bandwidth required on each link is thus the maximum of the additional backup bandwidth required by the backup trees being calculated.

The ILP formulation is still not completed as there is a small probability of dangling branches on the backup trees. Dangling branches refer to those links in the tree that do not connect to any receiver. This is possible due to equation 37 where the cost of using any link can be overlapped between the backup trees in calculation. Therefore, if including the dangling branches do not increase the objective function then the backup trees with the dangling branches are one of the optimal solutions of the problem. A post-processing is required to prune the dangling branches or additional methods can be used to avoid dangling branches. One method for removing dangling branches is by putting in additional constraints that do not allow the backup trees to have a branch if neither of the sender-receiver flows use that branch:

$$\sum_{d \in D} x_{ij}^{sd} \geq y_{ij}^{s}, \forall (i,j) \in E \text{ and } \forall s \in FS_m$$

This requires $|FS_m||E|$ additional constraints. The dangling branch issue applies to all the tree-based ILP formulations described in this paper.

D. Whole-Tree with Path Intermix ILP Formulation

The Whole-Tree strategy assumes that during failure, the traffic is switched from the service tree to the backup tree. Thus the bandwidth of the service tree is not used during the failure. We can allow the backup tree protecting the service tree to exploit the service bandwidth allocated to this service tree. This is the concept of path intermix [19] where the bandwidth used by the backup tree can be of two types. The first type is the service bandwidth belonging to the service tree and this is only for the links used by this service tree. The second type is the backup bandwidth on the link and this is only for the links not used by the service tree.

Using path intermix with complete recalculation of backup tree is the most flexible approach where the backup tree may choose to use the links on the service tree for with zero cost if it is a good decision to do so. It does not force the algorithm to reuse any part of the service tree.

For path intermix, the Whole-Tree Restoration ILP formulation remains the same but a new case is added to τ_{ij}^{s} as shown below. The first case ensures that the cost for reusing service tree's bandwidth is zero. The edges used by the service tree is denoted by E_m and the edges failed denoted by E_s.

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

E. Sub-Tree ILP Formulation

For each failure scenario affecting the service tree, Sub-Tree uses a backup tree to restore traffic to the affected destinations only. The unaffected destinations continue receiving traffic from the service tree. The optimal solution to this strategy can be reached by computing all the backup trees in the same ILP formulation.

$$\min \left(\sum_{(i,j) \in E} z_{ij} \right) \tag{41}$$

$$\sum_j x_{ij}^{sd} - \sum_j x_{ji}^{sd} = 0, \text{ for } i \neq h, d \; , \forall d \in D_s \; , \forall s \in FS_m \tag{42}$$

$$\sum_j x_{hj}^{sd} - \sum_j x_{jh}^{sd} = 1, \forall d \in D_s \; , \forall s \in FS_m \tag{43}$$

$$\sum_j x_{dj}^{sd} - \sum_j x_{jd}^{sd} = -1, \forall d \in D_s \; , \forall s \in FS_m \tag{44}$$

$$x_{ij}^{sd} \leq y_{ij}^{s} \leq 1, \forall d \in D_s \; , \forall (i,j) \in E \; , \forall s \in FS_m \tag{45}$$

$$\sum_j y_{ji}^{s} \leq 1, \forall i \in V - \{h\} \; , \forall s \in FS_m \tag{46}$$

$$\sum_j y_{jh}^{s} = 0, \forall s \in FS_m \tag{47}$$

$$z_{ij} \geq \tau_{ij}^{s}(y_{ij}^{s}), \forall (i,j) \in E \; , \forall s \in FS_m \tag{48}$$

$$z_{ij} \leq A_{ij}, \forall (i,j) \in E \tag{49}$$

$$x_{ij}^{sd}, y_{ij}^{s} \in \{0,1\} \tag{50}$$

$$z_{ij} \geq 0 \tag{51}$$

This IP formulation is similar to that of Whole-Tree's except the set of destinations the backup tree connects with only the affected destinations (D_s) of the failure rather than every the destinations.

F. Skeleton-Tree ILP Formulation

Skeleton-Tree Restoration is similar to Whole-Tree Restoration with path-intermix, except the unaffected portion of the service tree forms the skeleton of the backup tree. The affected destinations of the failure is connected back to this skeleton using the least cost paths.

$$\min\left(\sum_{(i,j)\in E} z_{ij}\right) \tag{52}$$

$$\sum_j x_{ij}^{sd} - \sum_j x_{ji}^{sd} = 0,\ \text{for } i \neq h, d\ ,\forall d \in D_s\ ,\forall s \in FS_m \tag{53}$$

$$\sum_j x_{hj}^{sd} - \sum_j x_{jh}^{sd} = 1, \forall d \in D_s\ ,\forall s \in FS_m \tag{54}$$

$$\sum_j x_{dj}^{sd} - \sum_j x_{jd}^{sd} = -1, \forall d \in D_s\ ,\forall s \in FS_m \tag{55}$$

$$x_{ij}^{sd} \leq y_{ij}^{s} \leq 1, \forall (i,j) \in E\ ,\forall d \in D\ ,\ \forall s \in FS_m \tag{56}$$

$$\sum_j y_{ji}^{s} \leq 1, \forall i \in V - \{h\}\ ,\forall s \in FS_m \tag{57}$$

$$\sum_j y_{jh}^{s} = 0, \forall s \in FS_m \tag{58}$$

$$x_{ij}^{sd} = 1,\ \text{If } (i,j) \in E_m^d\ ,(i,j) \notin E_s\ ,\forall d \in D - D_s \forall s \in FS_m \tag{59}$$

$$x_{ij}^{sd} = 0,\ \text{If } (i,j) \notin E_m^d\ ,(i,j) \notin E_s\ ,\forall d \in D - D_s \forall s \in FS_m \tag{60}$$

$$z_{ij} \geq \tau_{ij}^{s}(y_{ij}^{s}), \forall (i,j) \in E\ ,\ \forall s \in FS_m \tag{61}$$

$$z_{ij} \leq A_{ij}, \forall (i,j) \in E \tag{62}$$

$$x_{ij}^{sd}, y_{ij}^{s} \in \{0,1\} \tag{63}$$

$$z_{ij} \geq 0 \tag{64}$$

Additional constraints to the Whole-Tree Restoration formulation is required to form the skeleton. The constraints are defined in equations 59 and 60. E_m^d denotes the set of edges on service tree that routes from the source to destination d. The link cost equation τ_{ij}^{s} is the same as that defined when using path intermix (Section V-D).

VI. Simulation Results

For the simulations, four network topologies are used and are shown in Figures 7, 8, 9, and 10. Network1 and Network2 [17] are larger network topologies that are taken from actual topologies of provider networks. Note that in practice, networks of most providers are relatively small in size (i.e. not more than 100 routers). Application service providers requesting multicast support from these networks will specify a *relatively small* list of destination nodes in the provider's network which need to be connected by appropriate multicast paths. Typically these would be the border routers of the provider's network. Therefore it would not be uncommon to have these network provisioned multicast groups (connecting a subset of border routers) to be fairly small in size, e.g. 10 or 12 routers. In most of our simulations we limit the multicast group size to these numbers for two reasons: (1) we believe multicast connections on a provider's network will typically connect a group of routers of sizes in these ranges, and (2) from a practical simulation standpoint, we wanted to compare our results to the proposed ILP, and the computation

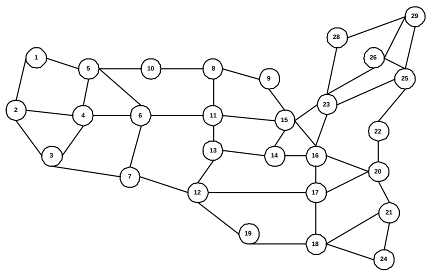

Fig. 7. Network1: US Long-Distance

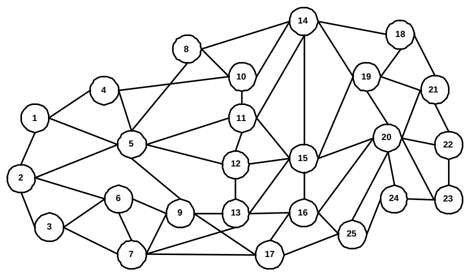

Fig. 8. Network2: Toronto Metropolitan

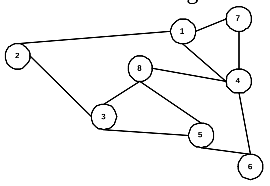

Fig. 9. Network3: Random Sparse Network

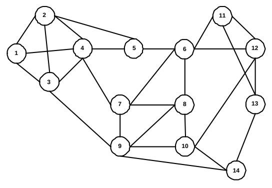

Fig. 10. Network4: Random Dense Network

overheads of the ILP is too high for larger group sizes. Our proposed polynomial-time algorithms clearly generalize to larger group sizes.

Network3 and Network4 are smaller network topologies that are randomly created.

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.

For large networks: Network1 and Network2, 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 solution. Thus we limit the number of destinations to 8 for large networks. For smaller networks, the number of destinations is limited by the number of nodes in the network. For Network3, the number of nodes is 8 thus the maximum number of destinations could only be 7 as the source node is not counted as a destination. For consistency with other topologies' simulations, we use only even number of destinations.

There are two types of experiments, one for the unlimited link capacity scenario, and the other is the limited link capacity scenario. For the limited link capacity scenario all the links in the network will have the same capacity for simplicity. The link capacity is chosen in the range where blocked requests occur for majority of the algorithms. The number of destinations are 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.

1) **Network1**: The link capacity varies from 200 to 400 units with incremental steps of 50 units.
2) **Network2**: The link capacity varies from 50 to 200 units with incremental steps of 50 units.

3) **Network3**: The link capacity varies from 500 to 800 units with incremental steps of 100 units.
4) **Network4**: The link capacity varies from 200 to 400 units with incremental steps of 50 units.

The following metrics are used for analysis of results:

1) Total bandwidth refers to the sum of the bandwidth required by all the service tree and all the backup paths/trees.
2) Backup bandwidth refers to the bandwidth used by all the backup trees.
3) Average backup structure size is the sum of the links used for restoring traffic over all failure scenarios affecting the service tree, divided by the number of failure scenarios. The size is measured by the number of links used.
4) 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. The acceptance rate is just the number of requests accepted divided by the number of requests considered.

The ILP formulations in the experiments are solved using CPLEX 8.1 [20] and the heuristic algorithms are implemented using C++. The experiments are executed on a PIII 1Ghz Linux Server with 1GB of RAM.

VII. Simulation Results Part I

The first set of simulation results assumes full multicast support within the networks, and compares between seven ILP-based solutions:

1) **Unicast Service**: uses shortest unicast service paths and ILP in section V-A for calculating the unicast backup connections.
2) **Simple Restoration**: uses service tree ILP (section III-A), and Simple Restoration ILP (section V-A).
3) **Line Restoration**: uses service tree ILP (section III-A), and Line Restoration ILP (section V-B).
4) **Whole-Tree Restoration**: uses service tree ILP (section III-A), and Whole-Tree Restoration ILP (section V-C).
5) **Whole-Tree Restoration with Path-Intermix**: uses service tree ILP (section III-A), and Whole-Tree with Path-Intermix Restoration ILP (section V-D).
6) **Sub-Tree Restoration**: uses service tree ILP (section III-A), and Sub-Tree Restoration ILP (section V-E).
7) **Skeleton-Tree Restoration**: uses service tree ILP (section III-A), and Skeleton-Tree Restoration ILP (section V-F).

The reason for including Unicast Service is to show the how much more efficient multicast-based solutions are compared with solutions without multicast support. Simple Restoration uses multicast for the service tree but not for backup. To restore traffic, Simple Restoration uses unicast backup connections. Comparing multicast backup tree solutions with Simple Restoration show the advantage of using multicast for backup. Line Restoration uses multicast service tree and routes the service tree around the failure. This is the alternative strategy compared with restoration solutions that use a new backup tree to restore traffic.

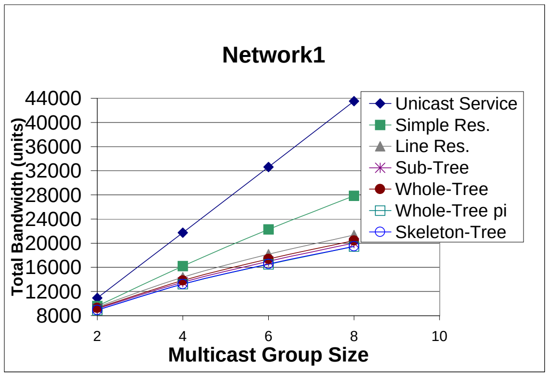

Fig. 11. Total Bandwidth Requirements for Network1

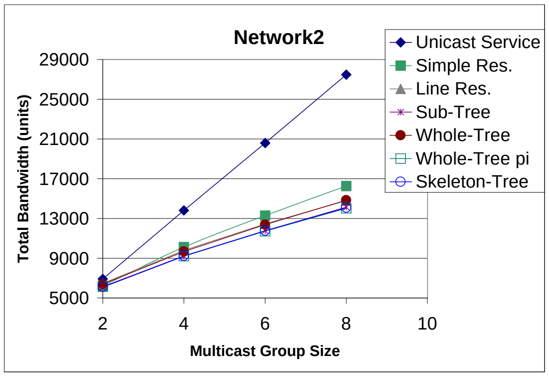

Fig. 12. Total Bandwidth Requirements for Network2

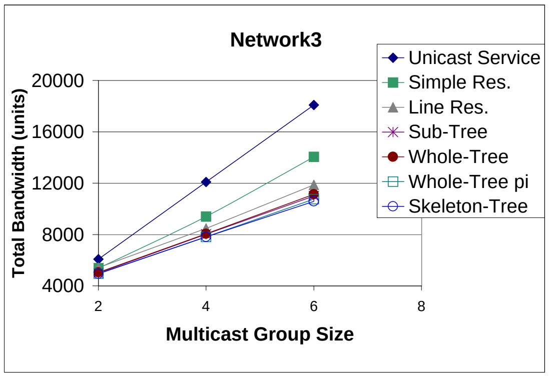

Fig. 13. Total Bandwidth Requirements for Network3

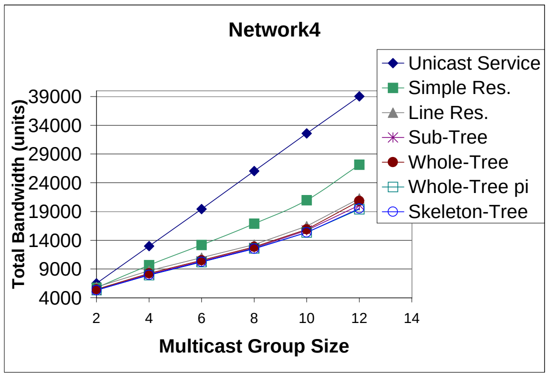

Fig. 14. Total Bandwidth Requirements for Network4

The results are grouped into simulations for unlimited link capacity scenario and for limited link capacity scenario.

A. *Unlimited Link Capacity Scenario*

The simulation results for the total bandwidth requirement for all the strategies in different network topologies are shown in figure 11, 12, 13, and 14. The main purpose of these graphs is to show the importance of using multicasting for both primary service communication and backup communication.

Focusing on the results for multicast group of size 6, the Unicast Service requires 32–50% more total bandwidth than the other ILP solutions in Network1, 35–43% more in Network2, 23–42% more in Network3, and 33–48% more in Network4. This highlights the benefit of using multicast for primary and backup communications. Simple Restoration performs better than Unicast Service mainly due to the bandwidth efficiency of the service tree (they both use unicast connections for backup).

We now focus only on the solutions that use service tree as it is the central focus of this paper. All the solutions use the same ILP-based algorithm for the service tree, thus the service bandwidth requirements are about the same. The main difference between the solutions come from the restoration strategies that have direct impact on the backup bandwidth requirement.

The backup bandwidth requirement results are shown in figures 15, 16, 17, and 18 for the different network topologies.

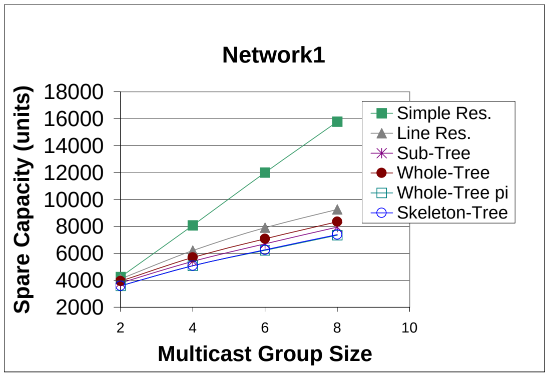

Fig. 15. Backup Bandwidth Requirements for Network1

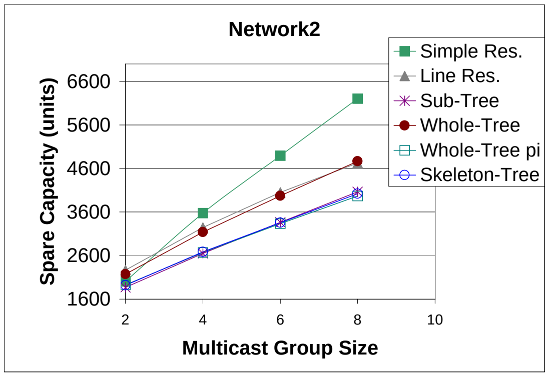

Fig. 16. Backup Bandwidth Requirements for Network2

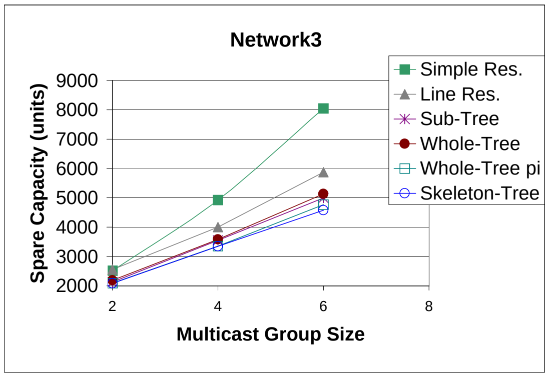

Fig. 17. Backup Bandwidth Requirements for Network3

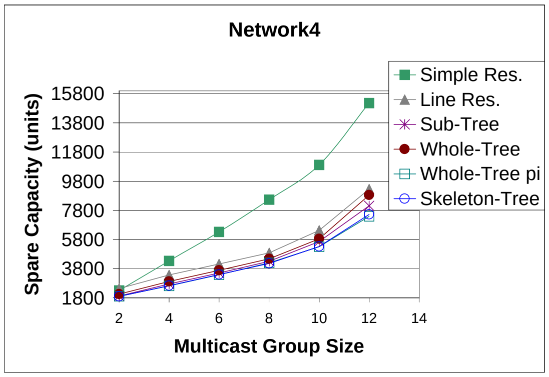

Fig. 18. Backup Bandwidth Requirements for Network4

We take a snapshot look at the results for multicast group of size 6. This is for analyzing the performance difference between the restoration strategies.

Simple Restoration clearly performs the worst with 35% more backup bandwidth requirement than others in Network1, 18% more in Network2, 27% more in Network3, and 35% more in Network4. The trend is aggravated when the multicast group size increases. All the other restoration strategies retain some form of multicasting during backup, therefore, *retaining multicasting during failure is an essential element for reducing backup bandwidth*.

The next worst performer is Line Restoration. Compared with Skeleton-Tree Restoration for group size of 6, Line Restoration uses 21% more backup bandwidth in Network1, 17% more in Network2, 22% more in Network3, and 18% more in Network4. Although, Line Restoration retains multicast efficiency by restoring the service tree and maximize reuses of the service bandwidth, it does not give the restoration algorithm enough freedom to spread out the backup bandwidth. Skeleton-Tree Restoration gives the algorithm complete freedom to choose the way the affected destinations join back the skeleton tree. This observation is similar to that of Line Restoration compared with Path Restoration in the context of unicast [16], [15], [17]. Therefore, *restoration that exploits the routing flexibility to reach further away from the failure can better reduce backup bandwidth*.

One of the solutions that perform better than Line Restoration is the Whole-Tree Restoration. Compared with Skeleton-Tree Restoration for group size of 6, Whole-Tree requires 12% more backup bandwidth in Network1, 16% more in Network2, 11% more in Network3, and 8% more in Network4. Whole-Tree retains multicast efficiency and provides complete flexibility on choosing the routes for connecting to the destinations during failure. However, it does not reuse any of the service bandwidth. Whole-Tree Restoration sacrifices the possibility of reusing service bandwidth for a higher degree of freedom to spread out the backup bandwidth. Skeleton-Tree Restoration and Sub-Tree Restoration both forces service bandwidth from the unaffected portion of the service tree to be re-used, but allow flexibility on choosing the routes taken to restore traffic back to the affected destinations. Results show that they both perform better than Whole-Tree Restoration. From this, we can deduce that *reusing the unaffected portion of the service tree is as important as routing flexibility*. A good algorithm provides a trade-off between the two.

Sub-Tree Restoration differs from Skeleton-Tree Restoration in the way the affected destinations of a failure are connected back to the source. Sub-Tree creates a new multicast tree that connects only the affected destinations. Using a two multicast tree can cause redundancy and the results between the two shows the consequence of this redundancy. Compared with Skeleton-Tree Restoration, Sub-Tree Restoration requires 7% more backup bandwidth in Network1, near equivalent in Network2, 8% more in Network3, and 3% more in Network4. The difference in performance is smaller in dense networks (Network2 and Network4) and performance difference is larger for smaller size networks (Network3 and Network4). For dense network, there is more chance of spreading the backup bandwidth out therefore reducing the impact of redundancy. The level of redundancy is largely increased in smaller size networks thus the observed difference. Hence, *using more than one tree during failure causes redundancy which increases the backup bandwidth requirement*.

As mentioned earlier, Whole-Tree Restoration does not reuse the service bandwidth thus requires much more backup bandwidth than Skeleton-Tree Restoration. By using path-intermix with the Whole-Tree Restoration (Whole-Tree-pi), we can choose to use the service bandwidth with zero cost and still retain the full freedom of choice on how to connect the source back to the destinations. Compared with Whole-Tree Restoration by itself, Whole-Tree-pi reduces backup bandwidth by 8% in Network1, 16% in Network2, 7% in Network3, and 8% in Network4. Path intermix removes the trade-off between reusing service bandwidth and freedom of choice as mentioned earlier. Reusing service path is now part of the freedom of choice for the algorithm. Compared with Skeleton-Tree, Whole-Tree-pi performs about the same for Network1, Network2 and Network4, while requiring 4% more Network3. Therefore, *forcing the unaffected portion of the service tree to be re-used is observed to perform just as well as encouraging reuse.*

An interesting observation to make is the backup bandwidth requirement between the dense networks: Network2 and Network4. Network2 is a larger network requires roughly similar amounts of backup bandwidth in the network to support 1000 requests. To explain this phenomenon, we use some additional information about the network state called the *link backup bandwidth frequency range*. That is, we group the links into equally spaced intervals (width of 20 units) of backup bandwidth requirement and count the frequency in each interval. We focus on the Skeleton-Tree results with group size of 6. The results are shown in figure 19.

Comparing the results between Network2 and Network4 shows that in Network2, the backup bandwidth are more equally distributed between the links at the lower range while in Network4, there are significantly larger amount

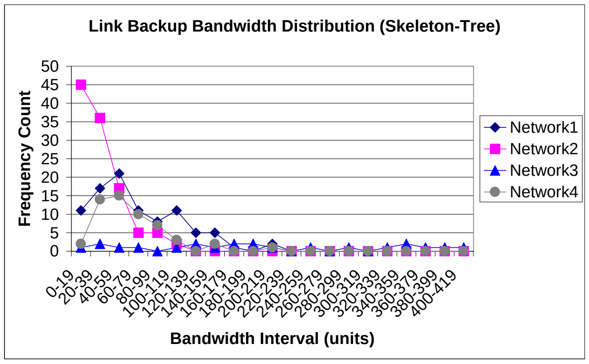

Fig. 19. Backup Bandwidth Distribution (Group Size 6)

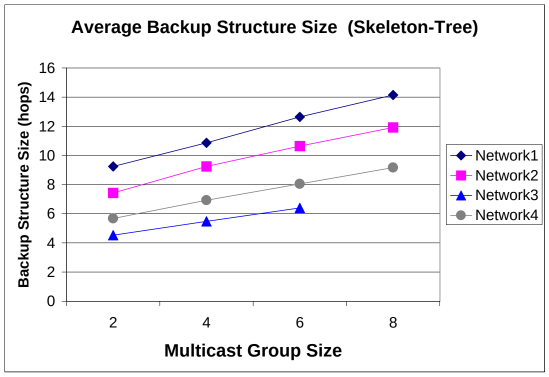

Fig. 20. Average Backup Tree Size (Group Size 6)

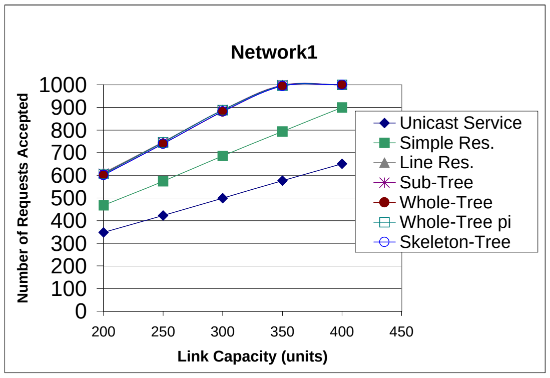

Fig. 21. Number of Accepted Requests for Network1

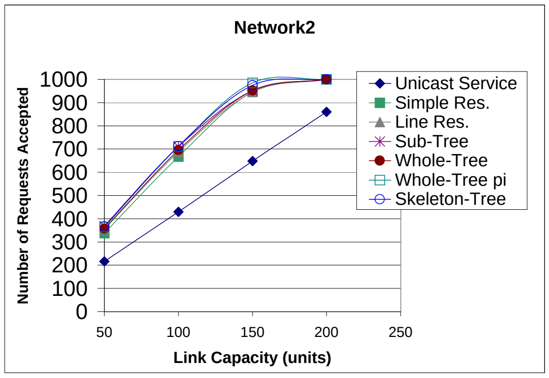

Fig. 22. Number of Accepted Requests for Network2

of links in the high range that contribute much more to the network backup bandwidth requirement. The larger size of Network2 provides more links for the algorithm to spread the backup bandwidth. Figure 20 shows the average size of the backup trees (the number of links used) for the Skeleton-Tree solutions in the different network topologies. The graph shows that the backup tree size are larger in Network2 than in Network4. This indicates that the algorithm reaches further out to find opportunities (shared bandwidth) that the larger network gives. Similar trend can be seen between Network1 and Network3.

Comparing results between Network1 and Network2 shows that dense network requires much less backup bandwidth than sparse networks. Figure 19 shows that the dense networks spread backup bandwidth more equally between links than in sparse network that have significantly more links having a large backup bandwidth allocation. This indicates that there are many hot-spots where the backup trees are forced to use. Dense networks provide more links thus more alternative route that the backup tree can traverse and find opportunities.

The trend line for Network4 indicates that backup bandwidth will start to grow exponentially as group size increases beyond 8. This coincides with the exponential growth in processing time for Network1 and Network2 when the group size increases over 8.

B. *Limited Link Capacity Scenario*

For the limited link capacity scenarios, the results are shown in figures 21, 22, 23, and 24. Unicast Service expectingly has the lowest acceptance rate,and that is followed next by Simple Restoration, while all the other

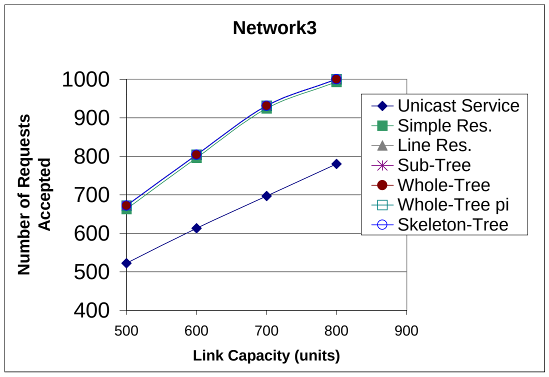

Fig. 23. Number of Accepted Requests for Network3

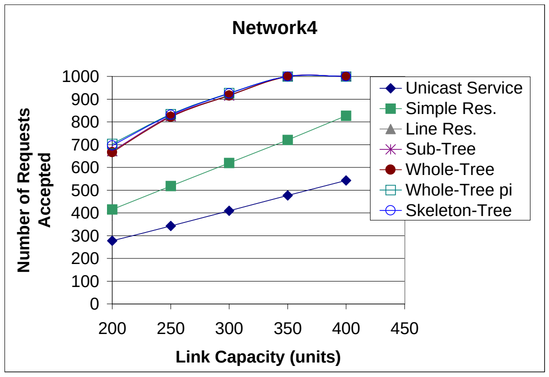

Fig. 24. Number of Accepted Requests for Network4

solutions perform very closely with each other (within 5%). The main reason for the blockings is from failing to find a feasible service tree. Although the backup bandwidth is minimized by the strategies, the service multicast 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 solutions. Note that the chance of blocking is much higher in multicast then in unicast as the number of links required to set up a multicast tree is significantly higher.

We find that the multicast service tree algorithms impose a higher impact on network efficiency than the backup restoration strategies because the service bandwidth contributes to most of the total bandwidth requirement. In networks where the backup bandwidth contributes to a higher percentage of the total bandwidth requirement, the backup restoration strategies impose a larger impact (Network1 and Network4 in our experiments).

VIII. New Heuristics Restoration Algorithm

Integer Linear Programming based solutions are computationally expensive and are known [21] to have exponential worst case times. The Skeleton-Tree Restoration solution 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. From the observation of the results on the ILP solutions in section VII, we draw up some heuristics that are observed to improve bandwidth efficiency:

1) Retain multicasting during failure recovery.
2) Force reuse of the unaffected portion of the service tree. This retains the general structure of the original multicast tree.
3) Use only one tree during failure (switch traffic from the service tree to the backup tree).

We propose a new Approximate Multicast Restoration Algorithm (AMRA) described in Algorithm 2 using these heuristics. The inputs to the algorithm are: the network resource state $G(V, E)$, the list of failure scenarios FS, and the service tree t. The expected outputs of the algorithm are: 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 scenario, we remove the failed links (E_s) and then we calculate a backup tree to restore traffic to all destinations.

```
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 $BT = \emptyset$;
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 $\tau_l^s$;
    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 $\infty$;
    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 2: Approximate Multicast Restoration Algorithm

The key idea of this algorithm is as follows. We first form the skeleton of the backup tree which is 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.

```
Data      : Network Graph: $G(V,E)$; Skeleton tree: $t$; destination: $d$;
Result    : Least cost path: $p$;
$path_{min} = 0$;
for each $v \in V_t$ 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 3: Shortest Path to Skeleton Tree

To find the shortest path from the skeleton-tree to a destination, we use the algorithm described in Algorithm 3. This is essentially just a loop that goes through all the node in the skeleton tree (V_t) and find 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:

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

Let R_l denotes the backup bandwidth allocated on the link l, and R_l^s is the backup bandwidth requirement that failure scenario s has on the link. 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 belongs to the service tree that can be used for path intermix. This link must not be part of the current skeleton tree and is not one of the failed links. Path intermix allows this link to be used without additional cost. The third case is when there is enough backup bandwidth to cover the request (b) in addition to what has been used by this failure scenario already. 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 (A_l) on the link to cover the additional bandwidth requirement.

The time-complexity of AMRA is polynomial and in order of $O(|FS_t|(|D||V|^2 \log |V| + |E|))$. The worst case for a failure is when all the destinations are affected ($|D|$) thus the backup tree needs to find connections to all the destinations. The worst case of founding a connection from the skeleton tree to a destination node is when the skeleton tree includes all the nodes in the network ($|V|$) thus for each node, we have to calculate the shortest path

$(O(|V| \log |V|))$.

It is possible for the algorithm to produce some inefficiencies from paths that intersect at an intermediate node. Although the result is still correct, we can increase efficiency by using the multicast ability at the intermediate node and remove the redundant components. Redundancy components can exist because links can be used with zero additional cost. A easy work around is to use a very small value in place of zero cost. This can prevent redundant components from forming without any extra processing cost, however, the value has to be chosen such that the resulting path will always be with the real least cost. An alternative is to do some extra processing to remove loops when calculating the least cost path that connects the destination to the skeleton tree.

AMRA is extended in section X to work for infrastructure overlay multicast networks.

IX. Simulation Results Part II

The experiment setup described in section VI is used for the simulation results presented here. The objective of the simulations is to show the performance difference between the heuristic-based solution and the ILP-based solution for the full multicast support networks.

Skeleton-Tree Restoration was shown in section VII to perform better than the other ILP-based solutions and is used in this experiment to compare against the heuristic-based solutions. MLTA-AMRA is our proposed heuristic solution which uses heuristic algorithms for both service tree computation and backup tree computation. We also combine MLTA with another heuristic restoration algorithm based on line restoration. This algorithm this called: *Line Restoration Algorithm* (LRA). The algorithm is derived using the greedy heuristic approach to approximate the ILP solution proposed in section V-B. Detail description of the algorithm is available in the Appendix.

1) **Skeleton-Tree Restoration**: this is a full ILP-based solution that uses service tree ILP (section III-A), and Skeleton-Tree Restoration ILP (section V-F).
2) **MLTA-AMRA**: this is a full heuristic-based solution that uses the MLTA service tree algorithm (section III-B), and AMRA algorithm for restoration (section VIII).
3) **MLTA-LRA**: this is another full heuristic-based solution that uses the MLTA service tree algorithm (section III-B), and LRA: a heuristic algorithm based on line restoration.

The results are grouped into simulations for unlimited link capacity scenario and for limited link capacity scenario.

A. Unlimited Link Capacity Scenario

The total bandwidth requirements are presented in Figures 25, 26, 27, and 28 corresponding to Network1, Network2, Network3, and Network4 respectively.

From the results, Skeleton-Tree Restoration has the lowest bandwidth requirement followed next by MLTA-AMRA, and then MLTA-LRA. Results show that MLTA-AMRA requires upto 4% more bandwidth than Skeleton-Tree Restoration in Network1, upto 10% more in Network2, 5% more in Network3, and 9% more in Network4. There are two reasons why Skeleton-Tree Restoration performs better: the first reason is due to the ILP that calculates the optimal least cost service tree. MLTA-AMRA uses an approximation algorithm (MLTA) that produces a sub-optimal tree. The consequence is that the service tree that MLTA-AMRA uses contains more links. This increases the

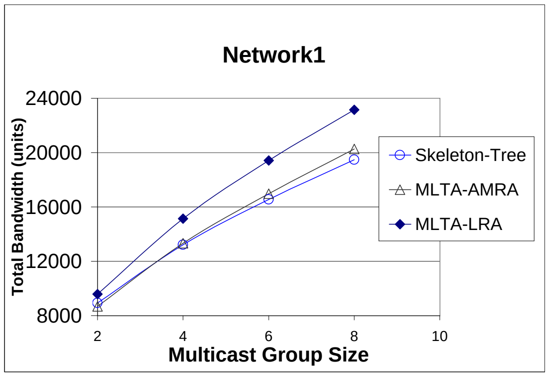

Fig. 25. Total Bandwidth Requirements for Network1

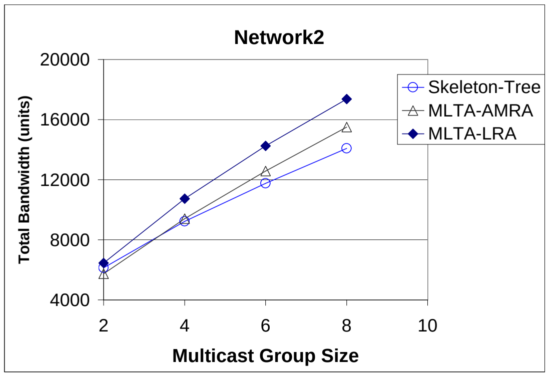

Fig. 26. Total Bandwidth Requirements for Network2

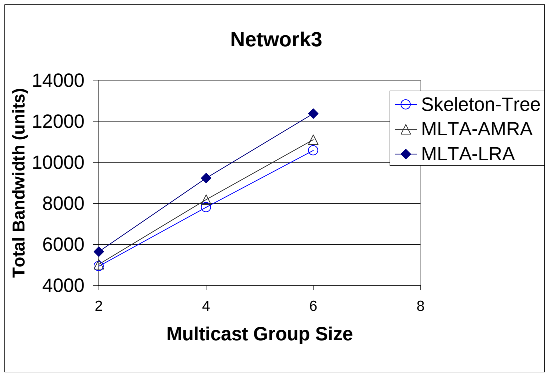

Fig. 27. Total Bandwidth Requirements for Network3

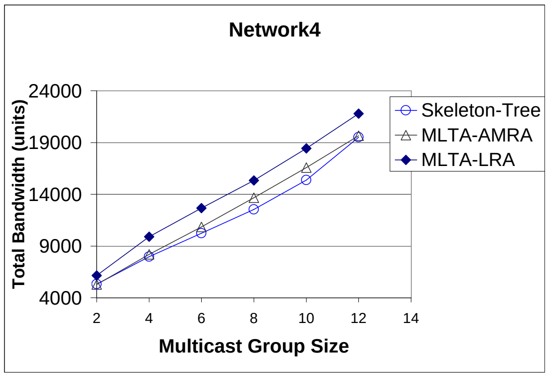

Fig. 28. Total Bandwidth Requirements for Network4

service bandwidth required and indirectly increases the backup bandwidth required. Backup bandwidth increases since more links on the service tree translates to more backup path segments. The second reason is because Skeleton-Tree Restoration uses the more efficient ILP formulation to calculate the backup trees compared to the greedy approach used in MLTA-AMRA.

MLTA-LRA requires upto 15% more bandwidth than MLTA-ARMA in Network1, upto 13% more in Network2, upto 13% more in Network3, and upto 21% more in Network4. MLTA-LRA and MLTA-AMRA use the same service tree algorithm, thus the difference in performance is from the difference in backup algorithms used. LRA is based on the line restoration strategy and AMRA is based on the skeleton-tree restoration strategy.

The backup bandwidth requirements are shown in figures 29, 30, 31, and 32 for Network1, Network2, Network3, and Network4 respectively.

In terms of backup bandwidth requirement, MLTA-LRA consumes upto 40% more than MLTA-AMRA in Network1, upto 52% more in Network2, upto 30% more in Network3, and upto 68% more in Network4. This show the true extent of the difference between the line restoration strategy (LRA) and the skeleton-tree restoration strategy (AMRA). AMRA is much more bandwidth efficient.

MLTA-AMRA takes about the same amount of backup bandwidth as Skeleton-Tree Restoration in Network1, upto 10% more in Network2, upto 6% more in Network3, and upto 12% more in Network4.

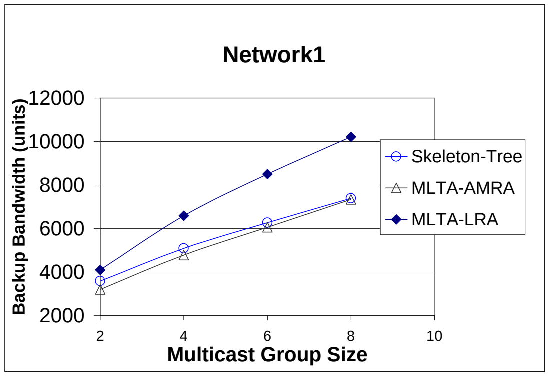

Fig. 29. Backup Bandwidth Requirements for Network1

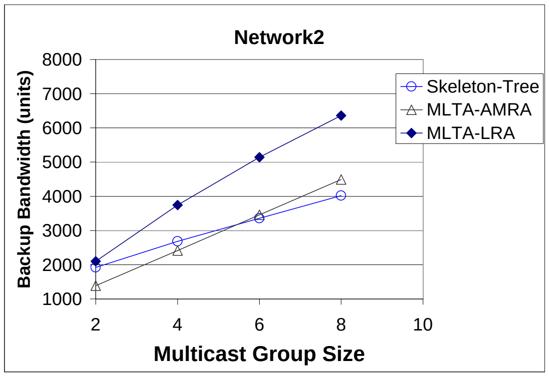

Fig. 30. Backup Bandwidth Requirements for Network2

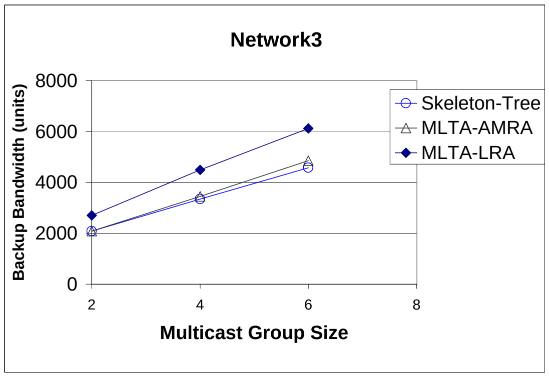

Fig. 31. Backup Bandwidth Requirements for Network3

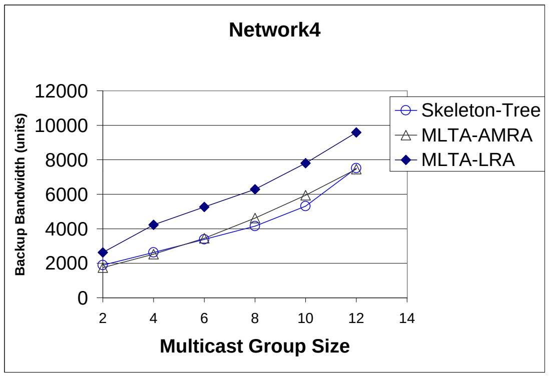

Fig. 32. Backup Bandwidth Requirements for Network4

B. Limited Link Capacity Scenario

For the limited link capacity scenarios, the results are shown in figures 33, 34, 35, and 36.

The results show that the three solutions perform very similarly in terms of the acceptance rate. MLTA-AMRA performs about the same as Skeleton-Tree Restoration for Network1 and Network3. For Network2, the acceptance rate for MLTA-AMRA reaches within 9% of Skeleton-Tree Restoration. MLTA-AMRA reaches within 12% of Skeleton-Tree Restoration in Network4. Compared with MLTA-LRA, MLTA-AMRA has upto 4% higher acceptance rate in Network1, upto 5% higher in Network2, upto 2% higher in Network3, but upto 3% lower in Network4.

The reason why all three solution performs so similar is due to the expensive cost setup for the service trees. Establishing the multicast service tree is very expensive and as the network becomes saturated, the probability of finding a feasible service tree is very low for all three solutions. Compare to unicast, multicast needs to connect to all destinations (in the simulations we use 6 destinations) , which means that the probability of blocking is substantially higher. We find that the main cause (nearly all) of request blocks for the solutions is from failing to find a feasible service tree. Although the backup bandwidth is minimized by the restoration algorithms, the service multicast 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 solutions.

Skeleton-Tree Restoration has better bandwidth efficiency and ILP will always find a feasible solution for the

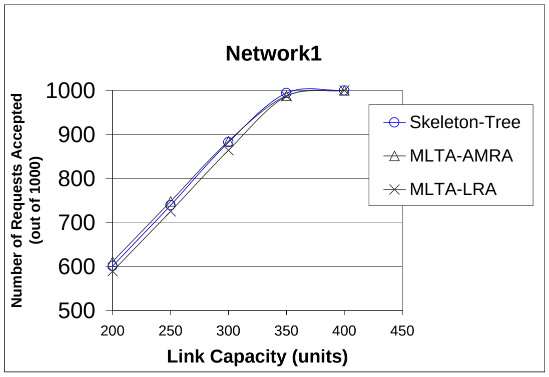

Fig. 33. Number of Requests Accepted vs Link Capacity

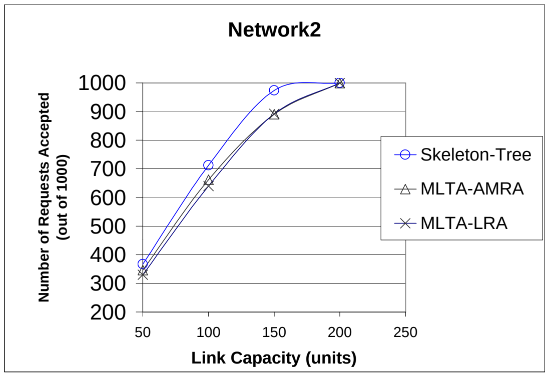

Fig. 34. Number of Requests Accepted vs Link Capacity

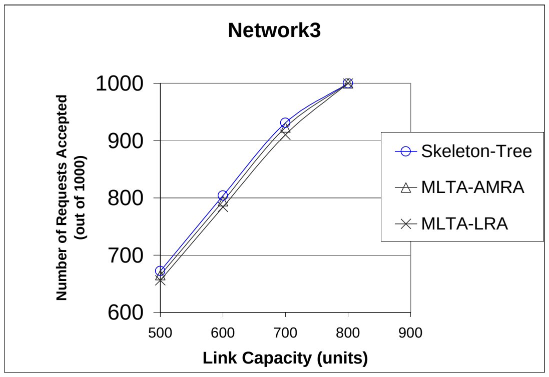

Fig. 35. Number of Requests Accepted vs Link Capacity

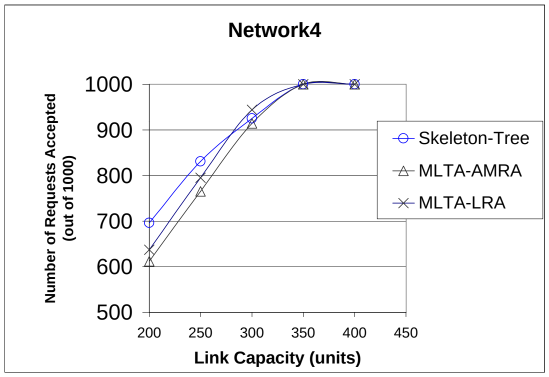

Fig. 36. Number of Requests Accepted vs Link Capacity

service tree if one exists. Thus it has slightly more acceptance rate than the other solutions, but the hot-spots cause all the solutions to block most requests at network saturation point.

Interestingly, a higher percentage of MLTA-AMRA's blocking results from failing to find feasible backup trees than the other solutions. This is especially significant in Network4 where MLTA-AMRA performs better than MLTA-LRA. AMRA achieves good backup bandwidth efficiency, however, this can sometimes adversely impact the network when the network becomes loaded. For example, LRA causes the links in the hot-spots to have higher backup bandwidth allocation than AMRA. As these links at congested, the higher backup bandwidth allocated on these links allow path segments to exploit the overlapping property (described in section IV) and thus the path segments will be able to find routes through the hot-spots, even if the links are full. However, AMRA will have lower backup bandwidth allocated on these links hence the backup tree will find it harder to find a feasible route through the hot-spots. It is these hot-spots that cause most of the blockings for MLTA-AMRA. This shows that there are trade-offs between bandwidth efficiency and load balancing within the network. Load balance is not only between links, but also balance between the service bandwidth and the backup bandwidth within the same link. Future work can be directed into investigating techniques in load balancing to increase the acceptance rate in limited capacity networks.

X. Supporting restoration in Infrastructure based overlay multicast

In infrastructure based overlay multicast networks such as OMNI, when a network provider initially deploys multicast support, it is likely that it will be done incrementally where only a limited number of Multicast Service Nodes are enabled. Therefore, we modify AMRA to operate in this environment such that it can exploit the MSNs whenever possible. We call this the *xAMRA* and the algorithm is described in Algorithm 4.

```
Data   : Network Graph: $G(V,E)$; Failure scenario set: $FS$; Logical service tree: $t$;
Result : New Network Graph: $G(V,E)$; Logical backup tree set: $BT$; Failure scenario set: $FS$;
Initialize $BT = \emptyset$;
Initialize $CON = \{h\}$;
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 $tree_{skeleton}$;
    Find set of connected nodes $CON_{new}$ in $tree_{skeleton}$ and let $CON = CON_{new} \cup CON$;
    Recompute the network links cost $\tau_l^s$;
    Find the list of affected destinations $D_s$;
    for each $d \in D_s$ do
        If $d \in CON$ then skip to next destination;
        Find the shortest path $p_d$ from any node in $CON$ to $d$;
        $tree_{skeleton} = p_d \cup tree_{skeleton}$;
        Update $G(V,E)$ for using the links in $p_d$ for backup in $s$;
        Find set of connected nodes $CON_{new}$ in $p_d$ and let $CON = CON_{new} \cup CON$;
    end
    Add $tree_{skeleton}$ to $BT$;
    Insert failed links back into $G(V,E)$;
end
```

Algorithm 4: Extended Approximate Multicast Restoration Algorithm

The general idea of xAMRA is to create a logical backup-tree for every failure scenario affecting the logical service tree. The physical links used in the logical tree do not need to be unique. Due to this non-uniqueness, we restrict the use of path-intermix in xAMRA. We force the logical backup tree to use the unaffected part of the logical service tree as the skeleton and use the service bandwidth allocated on the links in the skeleton. The skeleton is the only component that can reuse the service bandwidth.

First, we find the list of failure scenarios affecting the logical service tree, and then loop through each failure scenario to find a logical backup tree that restores traffic to the affected set of destinations. The list of links failed is removed from the network. To create a logical backup tree, we use the skeleton of the logical service tree that is made up of the links that connect to the unaffected set of destinations. The MSNs used in the skeleton are added into a set of *connected* nodes. We assume that the sender transmitting to the source node can do the multicasting thus the source node can be considered a MSN in this request. Thus the set of connected nodes will include the source node h by default. However, we do not assume the destination nodes to be MSNs. If the destination node is not a MSN then it cannot deliver a incoming packet out of the network and forward a copy of the packet to the downstream node at the same time.

For each affected destination, the objective is to find the least cost path that connects from any of the connected nodes to the destination. The least cost path is included as a skeleton, the network is updated to use this path, and any new MSNs traversed is added to the connected set. If a feasible path cannot be found to connect to the destination, then the request cannot be satisfied and the previous network updates for this request will be reversed. However, if the destination node is already in the connected set then it is already in the skeleton tree and it is also a MSN (otherwise it will not be in the connected set). Thus there is no need to find a least cost path to deliver the traffic to this destination node.

Once all the affected destinations are connected to the skeleton, we have a logical backup tree for the failure scenario. When calculating the least cost path, the cost of using the links are calculated using the following equation:

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

Let the set of all MSNs be denoted by M. The time complexity of xAMRA is in the order of $O(|FS_t|(|D||M||V|log|V|+|E|))$. Similar to AMRA (see section VIII), it is possible for the algorithm to produce some inefficiencies from paths that intersect at an intermediate node. An easy work around is to use a very small value in place of zero cost.

XI. Simulation Results Part III

The second set of simulation results compares between three solutions on infrastructure based overlay multicast networks:

1) **Unicast Service**: uses shortest unicast service paths and ILP in section V-A for calculating the unicast backup connections.
2) **Simple Restoration**: uses service tree ILP (section III-A), and Simple Restoration ILP (section V-A).
3) **xAMRA**: uses the heuristic algorithm MLTA (section III-B) for calculating the logical service tree and the heuristic algorithm xAMRA (section X) for calculating the logical backup trees.

Note that the ILP service tree solution computes the optimal service tree for the network-layer multicast case only, and hence we do not include results for that approach in this section.

For the infrastructure based overlay multicast scenario, we use a list of MSNs as shown in Table III and the multicast group size is fixed to 6 destinations.

TABLE III

LIST OF MSNS USED

Network Topology	Multicast Service Nodes
Network 1	3, 5, 6, 8, 11, 12, 18, 20, 23
Network 2	2, 4, 7, 10, 17, 18, 22, 24
Network 3	4, 8
Network 4	3, 5, 7, 8, 9, 11, 14

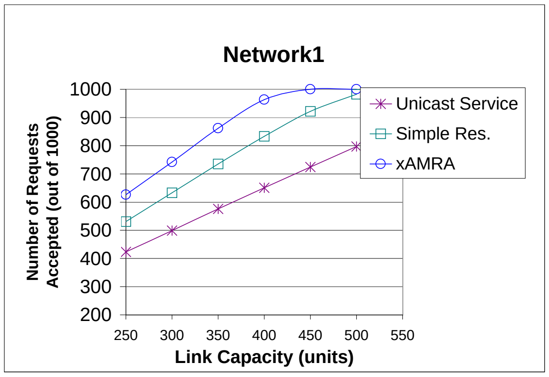

Fig. 37. Number of Requests Accepted vs Link Capacity

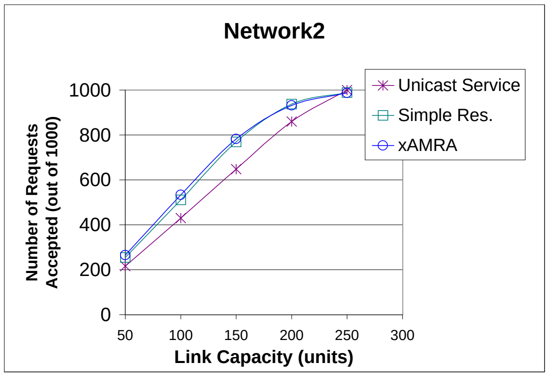

Fig. 38. Number of Requests Accepted vs Link Capacity

The performance of xAMRA and Simple Restoration is highly sensitive to the position of the Multicast Service Nodes (MSN) and the number of nodes. Thus we have chosen the placement of MSNs strategically to improve on performance, as Network Providers would do for their network topologies. The reason for the sensitivity is in the way the logical tree algorithms work. Although the MSNs can help improve bandwidth efficiency by building logical trees, the MSNs attract too much traffic to the area since the algorithms attempt to use these MSNs. If a MSN is placed in network hot-spots (the area where the network is most likely be loaded), then the MSN will draw in more traffic thus aggravating the congestion problem in the area. Knowing this behaviour, we can use the MSNs to draw traffic to other under-utilized area (load balance) thus improving the utilizing of the network. Systemic strategies for placing MSNs in the network is beyond the scope of this paper and is left as an interesting research topic for future work.

The results of the simulations are shown in figures 37, 38, 39, and 40 for Network1, Network2, Network3, and Network4 respectively.

Results show that significant gains can be achieved by using infrastructure based overlay multicast in the networks, as compared with no multicast support. For Network1 and Network4, xAMRA gains upto 50% and upto 84% higher acceptance rate than Unicast Service respectively. This corresponds with the results in section VII where MLTA-AMRA performs significantly better than Unicast Service in Network1 and Network4. For Network2, xAMRA has upto 24% higher acceptance rate than Unicast Serivce, and upto 12% higher in Network3. This shows the potential for using infrastructure based overlay multicast

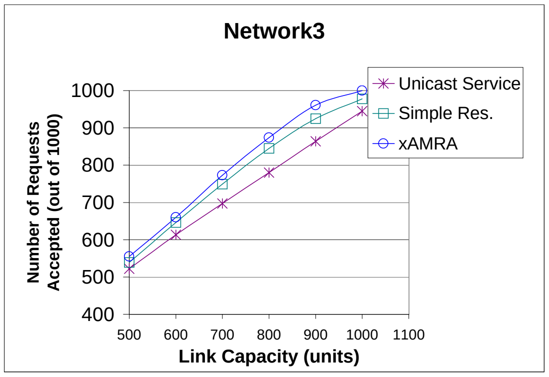

Fig. 39. Number of Requests Accepted vs Link Capacity

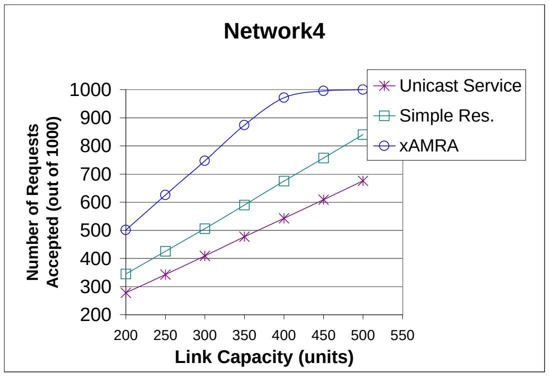

Fig. 40. Number of Requests Accepted vs Link Capacity

Significant acceptance rate improvements (upto 28%) by Simple Restoration over Unicast Service show the advantage of using logical multicast service tree. Comparing between Simple Restoration and xAMRA, xAMRA obtains significantly higher acceptance rate in Network1 (upto 18%) and Network4 (upto 48%), while while still providing small improvements in Network2 (upto 5%) and Network3 (upto 4%). This shows the benefit of using logical multicast tree for restoration.

XII. Related Work

Kodialam et al. [22] proposed a heuristic algorithm using the nearest neighbour first approach for the multicast least cost tree problem. The general idea is to connect the destination node to the multicast tree using the least cost path. However, this algorithm does not work for infrastructure based overlay multicast where only a limited number of network nodes can perform multicasting. In this paper, we propose a new algorithm that uses the nearest neighbour first approach to build logical multicast tree in infrastructure based overlay multicast networks.

Fei et al. [2] propose a dual-tree structure for multicast resilience. The idea is to build a secondary tree that is bi-directional and connects all the leaf nodes together. The possibility of bandwidth sharing between links used for the secondary tree was not investigated. In this paper we focus on minimizing bandwidth allocation for backup requirements.

Medard et al [1] proposed an algorithm to compute a backup tree for edge/node redundant graphs. The idea is to build a secondary tree such that when a link fails, all the destination nodes will be at least connected to one of the trees. The primary and secondary tree do not need to be completely node disjoint. The focus of their work is on finding a feasible secondary tree and not minimizing the backup bandwidth.

Banerjee et al [23] propose a probabilistic approach in multicast resilience. Using some random probability for redundant packets forwarding from leaf to leaf, they improve the chance of packet delivery during network failures. However, their objective was not focused on network efficiency and does guarantee recovery in deterministic failure scenarios.

Kodialam et al. [3] propose a heuristic algorithm for the multicast line restoration strategy for network-layer multicast. They called the algorithm: Multicast Complete Information (MCI). MCI was based on single node failures only.

In this paper we focus on single link failures and propose a new Integer Linear Programming (ILP) formulation for the multicast line restoration strategy. We use greedy heuristics to derive an algorithm (in Appendix) that approximates the ILP solution. Results show that the AMRA has better bandwidth efficiency than the line restoration heuristics and performs slightly better in limited capacity networks. We also extended our heuristic algorithms to work for infrastructure-based overlay multicast.

XIII. Conclusion

In this paper, we proposed new restoration strategies for multicast resilience with bandwidth guarantees. Results from ILP formulations of these strategies show that the Skeleton-Tree approach is more network efficient. Using the observations from the ILP results, we derive new heuristics that form the basis of our new heuristic algorithm (AMRA). Results show that AMRA forms close to the ILP-based solution and is more bandwidth efficient then the heuristic algorithm (LRA) based on line restoration. By extending our heuristic algorithms (MLTA and AMRA) to work in infrastructure overlay networks, we show that significant benefit can be derived from deploying limited multicast support compared with no multicast support. Our heuristics solution is expected to scale better than the ILP-based solutions for large network sizes and large multicast group sizes.

We believe that the techniques proposed in this paper can provide significant improvement in bandwidth guarantees for relatively long-lived multicast flows. Our techniques can be implemented in both network-layer only multicast as well as infrastructure-based overlay multicast. We believe that the benefits quantified in this paper will serve as an incentive for network providers to enable such services (even if via overlay-based mechanisms) for large group distribution applications.

References

[1] M. Medard, S. Finn, R. Barry, and R. Gallenger, "Redundant Trees for Preplanned Recovery in Arbitary Vertex-Redundant or Edge-Redundant Graphs," *IEEE/ACM Transactions on Networking*, vol. 7(5), October 1999.

[2] A. Fei, J. Cui, M. Gerla, and D. Cavendish, "A Dual-Tree Scheme for Fault-Tolerant Multicast," in *Proceedings of IEEE ICC*, Helsinki, Finland, June 2001.

[3] M. Kodialam and T. Lakshman, "Dynamic Routing of Bandwidth Guaranteed Multicasts with Failure Backup," in *Proceedings of IEEE ICNP*, Paris, France, November 2002.

[4] A. Ballardie, "Core Based Trees (CBT) Multicast Routing Architecture," *IETF RFC 2201*, 1997.

[5] S. Shi, J. Tuner, and M. Waldvogel, "Dimensioning server access bandwidth and multicast routing in overlay networks," in *Proceedings of NOSSDAV*, New York, USA, June 2001.

[6] Y. Chu, S. Rao, S. Seshan, and H. Zhang, "Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture," in *Proceedings of ACM SIGCOMM*, San Diego, USA, August 2001.

[7] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *Proceedings of ACM SIGCOMM*, Pittsburgh, USA, August 2002.

[8] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller, "Construction of an Efficient Overlay Multicast Infrastructure for Real-Time Applications," in *Proceedings of IEEE INFOCOM*, San Francisco, USA, March 2003.

[9] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," *IETF RFC 3031*, 2001.

[10] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels," *IETF RFC 3209*, 2001.

[11] M. Kodialam and T.V. Lakshman, “Dynamic routing of restorable bandwidth-guaranteed tunnels using aggregated network resource usage information,” *IEEE/ACM Transactions on Networking*, vol. 11(3), June 2003.

[12] G. Li, D. Wang, C. Kalmanek, and R. Doverspike, “Efficient Distributed Restoration Path Selection for Shared Mesh Restoration,” *IEEE/ACM Transactions on Networking*, vol. 11(5), October 2003.

[13] B. Waxman, “Routing of Multipoint Connections,” *IEEE Journal on Selected Areas of Communication*, vol. 6, 1989.

[14] P. Winter, “Steiner Tree problem in networks: A Survey,” *Networks*, vol. 17, 1987.

[15] M. Herzberg, S. Bye, and A. Utano, “The Hop-Limit Approach for Spare-Capacity Assignment in Survivable Networks,” *IEEE/ACM Transactions on Networking*, vol. 3(6), December 1995.

[16] R. Iraschko, M. MacGregor, and W. Grover, “Optimal Capacity Placement for Path Restoration in Mesh Survivable Networks,” in *Proceedings of IEEE ICC*, Dallas, USA, June 1996.

[17] Y. Xiong and L. Mason, “Restoration Strategies and Spare Capacity Requirements in Self-Healing ATM Networks,” *IEEE/ACM Transactions on Networking*, vol. 7(1), February 1999.

[18] H. Sakauchi, Y. Nishimura, and S. Hasegawa, “A Self-Healing Network with an Economical Spare-Channel Assignment,” in *Proceedings of IEEE GLOBECOM*, San Diego, USA, December 1990.

[19] W. Lau and S. Jha, “Failure-Oriented Path Restoration Algorithm for Survivable Networks,” in *Proceedings of IEEE/IFIP NOMS*, Seoul, Korea, April 2004.

[20] ILOG, “CPLEX 8.1,” *http://www.ilog.com*, 2003.

[21] Hillier and Lieberman, “Introduction To Operations Research, Seventh Edition,” *McGraw-Hill*, 2001.

[22] M. Kodialam, T. Lakshman, and S. Sengupta, “Online Multicast Routing With Bandwidth Guarantees: A New Approach Using Multicast Network Flows,” in *Proceedings of ACM SIGMETRICS*, Santa Clara, USA, June 2000.

[23] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan, “Resilient Multicast using Overlays,” in *Proceedings of ACM SIGMETRICS*, San Diego, USA, June 2003.

APPENDIX

A description of a greedy heuristic algorithm based on the line restoration strategy is provided in Algorithm 5. We call this algorithm: Line Restoration Algorithm (LRA),

```
Data   : Network Graph: $G(V, E)$; Failure scenario set: $FS$; Service tree: $m$;
Result : New Network Graph: $G(V, E)$; Backup path segment set: $BP$; Failure scenario set: $FS$;
Initialize $BP = \emptyset$;
Find the set of failures $FS_m$ affecting the service tree $m$;
for each $s \in FS_m$ do
    Remove failed link $l$ from $G(V, E)$;
    Recompute the network links cost $\tau_l^s$;
    If $l$ connects from node $k$ to node $n$ then find a shortest path from $k$ to $n$;
    Add the shortest path to $BP$;
    Update $G(V, E)$ for using the new path;
    Insert failed link $l$ back into $G(V, E)$;
end
```

Algorithm 5: Line Restoration Algorithm

The inputs to the algorithm are: the network resource state $G(V, E)$, the list of failure scenarios FS, and the service tree m. The expected output of the algorithm is the set of backup path segments BP, and the updated network resource state. First, we find the list of failure scenarios that affects the service tree FS_m. For each of these failure scenario, we find the backup path segment that restores traffic around the failed link. To find the path segment, we remove the failed link from the network graph, then we recompute the network link costs τ_l^s:

$$\tau_l^s = \begin{cases} 0, & \text{if } l \neq f \text{ and } R_l^s + b \leq R_l \\ b - R_l + R_l^s, & \text{if } l \neq f \text{ and } R_l^s + b > R_l \\ & \text{ and } A_l \geq b - R_l + R_l^s \\ \infty, & \text{otherwise.} \end{cases}$$

Let f denotes the failed link, and A_l denotes the available bandwidth on link l. R_l denotes the backup bandwidth allocated on the link l, and R_l^s is the backup bandwidth requirement that failure scenario s has on the link.

The first case occurs when a link can be used with no additional cost because there is enough backup bandwidth on the link to accommodate the backup bandwidth already required by this failure scenario with addition of the current requested bandwidth (b). The second case occurs when the backup bandwidth on the link cannot cover all of the requested bandwidth, thus additional bandwidth needs to be allocated to the link. This additional bandwidth required becomes the cost for using this link. 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.

Once we have the new network link costs, we find the least cost path segment that restores traffic around the failed link. This least cost path segment is then added to the set of path segments BP and the network resource state is updated for using this new path segment. The failed link is inserted back into the network graph, and we continue to find the next path segment for the next failure scenario. The algorithm ends when all path segments, which protect the service tree from the single link failures, are found.

The time-complexity of ALRA is $O(|FS_m|(|V|\log|V|+|E|))$, where the Dijsktra's shortest path algorithm used runs in $O(|V|\log|V|)$.