Fast Reroute from Single Link and Single Node Failures for IP Multicast

Aditya Sundarrajan, Srinivasan Ramasubramanian

Department of Electrical and Computer Engineering
University of Arizona
1230 E. Speedway Blvd., Tucson, AZ 85721
adityas@email.arizona.edu, srini@ece.arizona.edu

Abstract

The rise in multicast implementations has seen with it an increased support for fast failure recovery from link and node failures. Most recovery mechanisms augment additional services to existing protocols causing excessive overhead, and these modifications are predominantly protocol-specific. In this paper, we develop a multicast failure recovery mechanism that constructs protocol independent fast reroute paths to recover from single link and single node failures. We observe that single link failure recovery in multicast networks is similar to recovering unicast traffic, and we use existing unicast recovery mechanisms for multicast traffic. We construct multicast protection trees that provide instantaneous failure recovery from single node failures. For a given node x, the multicast protection tree spans all its neighbors and does not include itself. Thus, when the node fails, the neighbors of the node are connected through the multicast protection tree instead of node x, and forward the traffic over the multicast protection tree for the duration of failure recovery. The multicast protection trees are constructed a priori, without the knowledge of the multicast traffic in the network. Based on simulations on three realistic network topologies, we observe that the multicast protection trees increase the routing table size only by 38% on average and the path length between any source-destination pair by 13% on average.

Keywords: IP multicast, fast reroute, link failure, node failure, protection trees

1. Introduction

Multicast communication model is aimed at conserving network bandwidth when simultaneously transmitting identical data from a source to multiple interested recipients over a common link. This is achieved by transmitting only a

Parts of this paper appeared in the GLOBECOM 2013 conference [1].

single copy on the link shared by multiple receivers. Multicast communication is implemented by forming multicast trees with the destination nodes as the leaves. The source may or may not be part of this tree depending on the type of multicast model being implemented. For instance, the shared tree model in protocol independent multicast - sparse mode (PIM-SM) [2] does not require the source of traffic to be part of the multicast tree as long as it does not receive traffic from that group but, the source specific multicast model [3] forms the multicast tree with the source as the root. Throughout this paper, we assume the multicast tree includes the source. Intra-domain multicast protocols like protocol independent multicast (PIM) [2, 4, 3] and core based trees (CBT) [5] form such trees in tandem with multicast group management protocols like the internet group management protocol (IGMP) [6] to connect the source and the intended recipients. IGMP helps identify interested hosts within a router's domain and protocols like PIM help connect these routers also known as designated routers (DR). Inter-domain protocols such as multicast source discovery (MSDP) [7] and multiprotocol extensions for BGP-4 (MBGP) [8] provide WAN multicast functionality.

IP multicast routing is gaining importance with the advent of applications like webcasting sessions, multi-user gaming, financial data distribution such as stock tickers that involve multiple users interested in the same content. IPTV is the most popular multicast application over the IP infrastructure. It helps service providers integrate television services with high speed internet access. IPTV, unlike traditional cable systems delivers only the requested content to the subscriber thereby saving bandwidth. This technology comes at the cost of having to maintain extremely low disruption rates that this paper addresses.

The growth of high bandwidth applications like IPTV imposes stringent delay requirements on failure recovery. Network disruption lasting a few seconds could lead to massive loss of data affecting the quality of video traffic in particular (video traffic would amount to 90% of the consumer traffic in 2017 according to the Cisco VNI forecast for 2012-2017 [9]). For instance, compressed HDTV streams are transmitted at an average rate of 8 Mbps for a single HDTV output. This, combined with web-services and voice (telephone) dictates a minimum requirement of 13 Mbps per household on the IPTV network. A network failure lasting several seconds could result in loss of large amount of data.

The intra-domain protocols mentioned above have built-in failure recovery mechanisms. The drawback of these mechanisms is that they do not provide instantaneous failure recovery [10]. Once the original tree is broken, a new tree is formed to reconnect the source and the destinations. With limited buffer space at intermediate routers, the recovery often results in loss of data. To prevent this loss, recovery must take place as soon as the failure is detected. This could be enabled by proactive mechanisms that pre-compute backup paths, thereby allowing near-immediate recovery from failures and preventing loss of data.

1.1. Failure Recovery in Multicast Networks

Fast reroute mechanisms have been extensively studied for unicast networks but they cannot be directly applied to multicast scenarios owing to significant differences in the way multicast and unicast networks operate. In unicast communication, every intermediate node on the path from a source to its destination is aware of the final destination address of a packet. This helps them reroute traffic immediately to the destination during a failure, provided an alternate path is available [11, 12]. In contrast, the destination address in multicast communication is the multicast group address, and the source or any intermediate node is not aware of the individual destinations being served. Thus, an intermediate node cannot redirect traffic to the destinations when the downstream link/node in the multicast tree fails. For instance, consider the multicast session (S, G) (directed tree) in Figure 1 involving source S and destinations $D1$ and $D2$. Intermediate nodes 0, 1 and 2 are only aware of the multicast state (S, G) and have no information about $D1$ and $D2$. When node 1 fails, node 0 cannot redirect traffic immediately to the destinations. Hence, the key is to develop a mechanism that enables rerouting multicast packets to the destination(s) for the duration a new multicast tree is being formed, such that no data is lost.

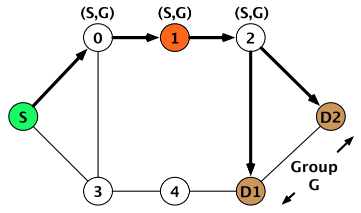


Figure 1: Example multicast tree rooted at node S and destination nodes D1 and D2.

One approach to recovering from failures is to construct multiple back up trees also called protection trees (as they bypass (protect) the failed link or node) rooted at every node in the network. Having multiple link/node disjoint trees rooted at a node helps routing around link/node failures by switching to the alternate tree when a failure is detected on the primary tree. We may construct protection trees that are specifically designed for link failures and node failures separately. For example, we can construct link independent trees and vertex independent trees rooted at a destination that would provide link-disjoint and node-disjoint paths from every node to that destination. However, in order to choose which protection tree to use, we need to classify if the failure encountered is link or node failure. As node failures are relatively rarer compared to link failures, the common approach is to assume that the failed outgoing link at a node is a simple link failure and attempt a recovery. If the packet encounters one or more subsequent link failures that share a common node, then we assume that a node has failed and attempt to recover from the node failure.

Recovery from link failures for multicast traffic is fairly straight-forward. If a multicast packet has to be forwarded to a neighbor and the link has failed, we may forward the packet to the neighboring node along an alternate (backup) path using encapsulation. The encapsulated header would have its destination as one of the alias addresses of the neighboring node. The routing for the alias address takes into account the link failure encountered by the packet.

Approaches such as not-via routing using tunnels [13, 14] and recovery from multiple link failures using colored trees [15] use such encapsulation techniques and alias addressing to recover from link failures in unicast networks. The multicast single link failure scenario is no different and fast recovery for multicast traffic from link failures is similar to recovering unicast traffic.

However, the node failure scenario is exactly what was mentioned earlier in Figure 1 where the intermediate nodes in the multicast tree cannot reroute the multicast packets directly to the destinations. In this paper, we propose a fast rerouting mechanism for recovering from node failures by constructing multicast protection trees (MPTs). We identify the failure of a node at all its neighboring nodes and reroute the packets on the MPTs. These trees require no knowledge of the final destinations, yet recover from single node failures immediately. Also, these trees can be computed in advance, independent of the multicast session it will serve unlike many recovery mechanisms that augment additional services to the multicast protocol being implemented today.

1.2. Contributions and Organization

The contributions of this paper are:

1. Developing a single link fast reroute scheme in multicast networks using existing unicast mechanisms.
2. Proposing MPTs that provide single node fast reroute in multicast networks.
3. Evaluating the effectiveness of these two protection mechanisms to provide protocol independent multicast failure recovery in 2-vertex connected networks.

The rest of this paper is organized as follows. Section 2 looks at other recent works that have dealt with the same problem. Section 3 sets the network model, and introduces and explains the colored trees and the MPTs that we use for link and node failure recovery respectively. Section 3 also describes the multicast failure recovery procedure from single link and single node failures. We evaluate the efficiency of this recovery scheme on three practical networks in Section 4. Finally, our conclusions are presented in Section 5.

2. Related Work

Multicast networks have gained considerable interest owing to their bandwidth conserving one-to-many communication model. Failure recovery in these networks is a natural consequence to provide a stable operation. The interest in developing fast reroute schemes to bypass failures is fueled by the growing deployment of high bandwidth applications. Several works develop failure recovery techniques that minimize the loss of data, and these techniques either construct the original multicast tree to be failure resilient such as [16, 17, 18] to name a few or develop recovery mechanisms for an arbitrarily constructed multicast tree. In this paper, we are only concerned with techniques that protect an arbitrarily constructed tree.

These recovery mechanisms can be broadly classified in to two categories. One, that looks at modifying the current multicast protocol being deployed and the other, that develops generic algorithms to overcome failures not specific to any particular multicast protocol. We look at some of the recent works in both these areas.

2.1. Protocol Dependent Recovery Techniques

PIM and its variants are the most commonly used multicast protocols in today's Internet. Many Cisco routers use PIM-SM for their multicast deployments [19]. PIM-SM has its own recovery mechanism that is triggered once the underlying unicast protocol converges after the failure is detected. This could take several seconds resulting in loss of high bandwidth data. The work in [20] proposes a cross-layer design that helps PIM converge to the new multicast tree quicker than it usually does, without waiting for the underlying interior gateway protocol (IGP) to converge over the entire network. Single link/node failures typically affect only a small section of the network, and local convergence should be sufficient for the multicast protocol above to find a new path. This is implemented using *pending joins* that modifies the way PIM reacts to the IGP reconvergence. This ensures *hitless* (i.e. without loss) transitions to the new multicast tree as soon as the local failure is detected. This work guarantees resilience under multiple link failures without link overlap during failure recovery. Link overlaps are avoided using a link weight setting algorithm developed in [17] that adaptively changes link weights in the face of a failure. The paper however does not address node failures and is based on the assumption that multiple failures are separated by enough time to allow their algorithms to converge. This work provides failure recovery only for PIM source-specific multicast (PIM-SSM) and cannot be used with other protocols. A related work [21] similar to ideas in [22], recovers from single link failures by constructing fast reroute paths that are disjoint from the original multicast tree. A node detecting an outgoing failed link encapsulates the received multicast packet and transmits it on the disjoint unicast path. This work guarantees disjointedness using the algorithm developed in [17]. The intention is to minimize traffic overlap during failure recovery for high bandwidth applications like IPTV. However, the proposed technique relies on non-identical weights in each direction on a bi-directional link.

The technique in [23] like this paper, relies on tunneling to recover multicast traffic from link or node failures in networks that deploy PIM-SM or PIM-SSM. The tunnels to route around failures are established for every multicast session after the original multicast tree is set up. This could be expensive especially in SSM deployments where there are multiple multicast sessions for the same group G. These tunnels are set up using new *PIM hello messages* that are sent between adjacent PIM nodes for link protection, and from the downstream node to the upstream node of the protected node for node failure recovery. However, the draft does not specify the number of links and nodes that can be protected using this technique because it may not always be possible to discover a backup path.

Another proposal [24] relying on modifications to PIM works by maintaining not-via paths [13] to next-next-hop (nnh) nodes in the original multicast tree. The nnh nodes are advertised in the PIM *join* messages and not-via paths to these nodes are computed. This technique would require several not-via paths to a node and several not-via addresses depending on the number of multicast trees it is a part of as each not-via tunnel is bound to the multicast session it protects. Hence, it is not clear how many not-via addresses a node can have and a large number of not-via paths could lead to excessive maintenance overhead.

2.2. Protocol Independent Recovery Techniques

Belonging to the class of algorithms that are independent of the multicast protocol, [25] uses ring topologies to ensure that every link in the original multicast tree is protected. The backup routes are formed by connecting a set of nodes that form a cycle (ring). The nodes in a cycle remain connected even after a single link failure. This protection scheme does not modify the operation of the multicast protocol but depends on the original multicast tree to form cycles. These cycles are "session" specific and links belonging to multiple multicast sessions have multiple recovery paths. It is to be noted that each link could belong to multiple cycles (recovery paths) for a single session and it is unclear which protection path would be used. However, these rings cannot be used for node failure recovery.

[26] is another proposal that provides fast reroute without modifications to the multicast protocol being deployed. This technique works for both IP and MPLS based networks. In this work, a merge point router (multicast only fast reroute enabled router) joins the multicast tree via two disjoint upstream paths, one, through the primary upstream multicast hop (UMH), and the other, through the secondary UMH to the root of the multicast tree. The merge point accepts packets from both upstream nodes and forwards only those received from the primary upstream node under a failure free scenario. When a failure occurs on the primary node, the secondary upstream node becomes the primary upstream neighbor. This ensures fast recovery. This technique incurs the overhead of redundant traffic flowing over the network affecting network bandwidth. This is pronounced if there are multiple merge points in the network. A more recent work [27] based on [26] aims to quicken the switch from the primary to the secondary UMH in the face of a failure. It also helps identify which node in the multicast tree (merge point router) should switch over to the secondary UMH to bypass the upstream failure. This work also overcomes the live-live nature of the previous scheme where the merge point router does not receive from the secondary UMH until the primary UMH has failed. However, the drawback is that failure recovery can be activated only by merge point routers and not by the nodes detecting the failure (in most cases) leading to delays in the start of failure recovery.

It is worth noting that fast rerouting algorithms that modify or depend on current protocol deployments set up back up paths only after the original multicast tree is established, and these backup paths are specific to the multicast session they protect. This results in multiple back up paths for every link/node in the network depending on the number of multicast sessions that link/node is a part of, resulting in excessive overhead due to back up path maintenance. Both of the above drawbacks are absent in the techniques proposed in this paper and every link/node has only one back up path irrespective of the multicast session it is a part of. Also, the back up paths in this paper are created independent of the individual multicast sessions.

3. Fast failure recovery in IP Multicast networks

3.1. Network Model

The networks dealt with in this paper are at least 2-vertex connected, with the ability to withstand single link/node failures without network disruption. We consider the multicast operations within an autonomous system (AS) domain, and the network employs unicast link state protocols like open shortest path first (OSPF) or intermediate system to intermediate system (IS-IS) rendering the entire network view to all the nodes. All links are bidirectional, which is an essential requirement for fast reroute mechanisms to work. The failure of any link disrupts the communication in both directions. Also, a node is assumed to have failed when at least n incident links have failed in an n-edge network ($n =$ 1, 2 ,...). The notations used throughout this paper are listed in Table 1.

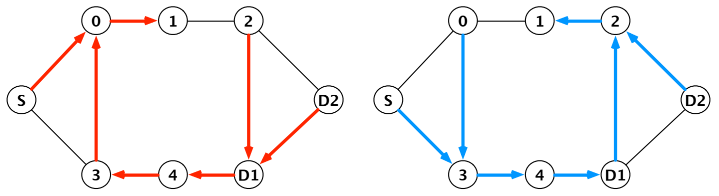
Table 1: Notations used in this paper.

Notation	**Comment**
$G(V, E) = G$	Graph G with vertex set V and edge set E
$\|V\|$	Total number of vertices in G
$\|E\|$	Total number of edges in G
V_X	Vertices in any set X
$T(V_T, E_T) = T$	Tree T with vertex set V_T and edge set E_T
$Ne(z) = ne_z^1,\ ne_z^2,\ \ldots,\ ne_z^{N_z}$	N_z neighbors of node z, $N_z = 1, 2, \ldots$
PA_v	Protection address of node v
MPT_v	Multicast protection tree of node v

As indicated in Section 1, the failure recovery procedure always begins as link failure recovery followed by node failure recovery when at least two incident links on the common node have failed. The protection trees for failure recovery are used until a new multicast tree is constructed. Link failures in multicast networks can be recovered from using existing unicast techniques and we illustrate the use of colored trees to bypass the failed link. Node failures are recovered from using MPTs that we develop in this paper. In the following sections, we briefly explain about colored trees and focus more on the MPTs that form the major contribution towards a protocol independent failure recovery mechanism in IP multicast networks.

3.2. Colored Trees

Colored trees [15, 28] provide an efficient mechanism to develop link/node disjoint paths from every node to every other node in the network. This is achieved with minimum routing table overhead and lesser lookup time. Two trees namely red and blue trees rooted at a destination are constructed such that the every node can reach the destination via two disjoint routes on these trees. Such trees are proactively created for every node in the network. It is worth noting that it is necessary and sufficient for a network to be 2-edge (2-vertex) connected to compute the colored trees such that every node has two link (node) disjoint paths to the root on these trees. In the following figure, the red and blue trees rooted at node 1 are shown in Figure 2.



(a) Red tree at node 1 (b) Blue tree at node 1

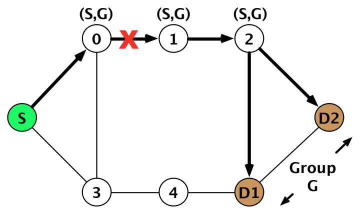
Figure 2: Red and blue trees rooted at node 1.

The colored trees approach provides two forwarding links (red and blue) at every node for a destination via two routing table entries, one for each tree. The forwarding link chosen by the node is indicated using a color bit (CB) which is carried in the header of every packet. CB can be any of the unused bits in IP header. A value of 0 indicates a blue forwarding link and 1 indicates a red forwarding link. Every downstream node in the forwarding path reads the CB to decide which next hop node towards the destination should be used.

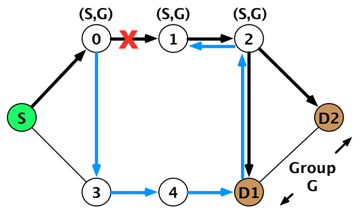
Colored trees can be used for tunneling traffic directly to the destination during a failure. When there are no failures in the network, the forwarding decisions are based on only the shortest path that leads to the destination. But if a failure occurs, the forwarding decision should be based on the destination address and the type of failed outgoing link. If the failure occurred on an outgoing red (blue) link at a node then the blue (red) outgoing links towards the destination have to be used for the reminder of path as long as another failure does not occur. However, if the failed outgoing link was neither red nor blue, then the node could choose either the red or blue neighbor as the next hop node to forward traffic to the destination.

The tunneling mechanism mentioned above, makes these colored trees well suited for link failure recovery in multicast networks where the destination that has to be reached is the node on the other side of the failed link. For instance, in Figure 3, if the link $0 - 1$ on the original multicast tree fails, node 0 uses the blue tree rooted at node 1 to reach node 1 for the duration of the failure recovery, during which the new multicast tree will be formed. The blue forwarding neighbor was used since the red outgoing link was the failed link.

This illustrates that link failure recovery in multicast networks is no different from that in unicast networks and existing unicast fast reroute mechanisms could be used for multicast networks too. This has the added advantage of requiring



(a) Failure of link $0-1$



(b) Fast reroute using blue path from nodes 0 to 1

Figure 3: Single link failure recovery using colored trees.

no new mechanisms to be installed for multicast link failure recovery. This is an important observation as one common link reroute mechanism can be installed for different communication paradigms. The algorithm to compute these colored trees can be found in [15].

3.3. Multicast Protection Trees

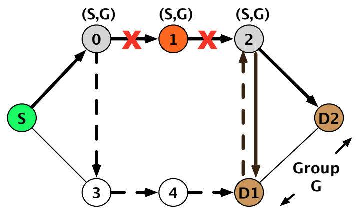
The MPTs are Steiner trees [29] that span the neighbors of a node. Such trees help the node (neighbor) detecting the failed next-hop node reroute around the failure requiring no knowledge of the individual destinations in the original multicast tree, or in fact the multicast tree itself. For a node v, its MPT denoted by MPT_v, is the tree spanning $Ne(v)$ and not containing v. This tree facilitates the communication between the neighbors of node v without having node v involved. MPT_v connects all the downstream neighbors of v with the upstream neighbor. This guarantees node failure recovery of the original multicast session as one or more downstream neighbors would be part of the original multicast tree.

Every node v is assigned a multicast protection address, referred to as PA_v. Node v shares its multicast protection address with its neighbors $Ne(v)$. If a packet is destined to PA_v, then the intended destination nodes are neighbors of v. Note that only packets with destination address PA_v need to be routed on MPT_v. Thus, we may proactively construct the MPT for every node and populate the routing table entries at the nodes in MPT_v for PA_v.

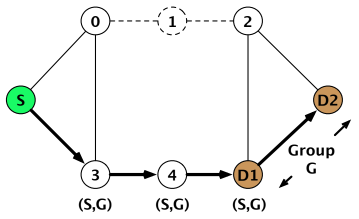
Consider the example multicast tree shown in Figure 1. Let PIM-SSM be the multicast protocol being deployed. Figure 4(a) shows the multicast protection tree for node 1 (MPT_1): 0–3–4–D1–2. As node 1 has only two neighbors, the tree is simply a path. Once the failure of node 1 is detected, node 0 sends out the multicast packets that needs to be forwarded to node 1 to all the neighbors on MPT_1 using encapsulation, with destination address set to the multicast protection address of node 1. Nodes that are neighbors of node 1 are the intended destinations of this encapsulated packet. Upon receiving this encapsulated packet, the neighbors of node 1, decapsulate the packet and forward the inner packet based on the normal forwarding rules.

Upon detecting a failure, PIM-SSM kick starts its recovery procedure by sending out new *join* messages towards the source to form a new multicast tree.

Once the new tree is established, the MPT computed before the failure may be discarded. Multicast protocols can activate make-before-break mechanisms by adjusting their failure reaction timers to ensure lossless data transmission on the MPT during failure recovery. This requires no enhancements to current multicast protocol operations and gives complete freedom to future protocol designs by making failure recovery an independent operation.



(a) MPT for node 1



(b) New multicast tree

Figure 4: Example multicast tree rooted at node S and destination nodes D1 and D2.

3.3.1. Computing Multicast Protection Trees

The MPT of node v is a Steiner tree with the neighbors of v as the set of leaf nodes. As packets sent to the protection address are broadcast in the multicast protection tree (as it has to reach all leaf nodes), we need to reduce the number of links in the tree in order to minimize resource consumption. Thus, the optimal MPT is the minimum Steiner tree, which is known to be an NP-Hard [30] problem.

In the following sub sections, we illustrate two centralized algorithms that use link state protocols like OSPF to obtain the network topology, using which the MPTs are constructed. We evaluate the two sub-optimal algorithms on a given graph $G(V, E)$ that offer different trade-offs. One based on computing shortest paths from a node $w \in Ne(v)$ to all other nodes in $(Ne(v) - w)$ and the other based on appending shortest paths from the existing tree $T(V_T, E_T)$ to vertices in $(Ne(v) - V_T)$. Prior to the construction of these trees, the node whose MPT is to be calculated, sends a list of all its neighbors to an arbitrary neighbor which initiates the construction. The node in question along with its incident edges are not considered. We next describe these two algorithms.

Single neighbor shortest path trees:

A neighbor ne_v^1 of a node v is chosen as the root of the MPT and shortest paths are computed from that root to every other neighbor of v. This is shown in Algorithm 1 and has a time complexity of $\mathcal{O}(|Ne(v)| \times (|E| + |V| \log |V|))$, where $\mathcal{O}(|E| + |V| \log |V|)$ is the running time for Dijkstra's(x,y), i.e. Dijkstra's shortest path algorithm between nodes x and y in G, using Fibonacci heaps [31]. This algorithm provides a quick sub-optimal solution but could lead to much larger MPTs as the MPT is optimized from the perspective of the root rather than the entire tree. For this reason, we do not use this algorithm to construct

MPTs in this paper. But it could be used in scenarios where bandwidth is not a constraint and quick computations of MPT are required after the new multicast tree is formed such as for VoIP services.

Algorithm 1 Single neighbor SPTs

```
Input: Graph $G(V, E)$, any node $v \in V$.
Output: $MPT_v$.
1: $i = 2$. For node $v$,
2: while $i \leq N_v$ do
3:     $P_i$ = Dijkstra's$(ne_v^1, ne_v^i)$, $P = P \cup Pi$
4:     $i = i + 1$
5: end while
6: $MPT_v = P$
```

Augmented Steiner trees:

Another approximation to the Steiner tree problem based on [32] for computing MPT$_v$ relies on adding shortest paths from the existing tree, $T_i(V_{T_i}, E_{T_i})$ where, $V_{T_i} \subseteq V$, to a node in $(Ne(v) - V_{T_i})$ (neighbors of v that have not been visited yet). This procedure is performed iteratively until all nodes in $Ne(v)$ are visited. The procedure is illustrated in Algorithm 2.

This algorithms provides near-optimal solutions since the cost of the entire tree is minimized. It has the same time complexity as Algorithm 1 as each new path $P(T_{i-1}, ne_v^i)$ in the i_{th} iteration is computed in $\mathcal{O}(|E| + |V| \log |V|)$ time. However, it is interesting to note that the number of paths that have to be examined at each iteration i is $|V_{T_{i-1}}|$. This is because step 6 always chooses the shortest among all possible paths.

Algorithm 2 Augmented Steiner trees

```
Input: Graph $G(V, E)$, any node $v \in V$.
Output: $MPT_v$.
1: $T_i(V_{T_i}, E_{T_i})$ = Tree formed at $i^{th}$ iteration
2: $P(T_{i-1}, ne_v^i)$ = Shortest path from $V_{T_{i-1}}$ to $ne_v^i \in (Ne(v) - V_{T_{i-1}})$
3: $T_1 = (ne_v^1, \phi)$
4: $i = 2$
5: while $i \leq N_v$ do
6:     $T_i = T_{i-1} \cup P(T_{i-1}, ne_v^i)$
7:     $i = i + 1$
8: end while
9: $MPT_v = T_i$
```

On comparing the average number of edges in the MPTs constructed using the above algorithms for the NSFNET topology, we observe that the average number of edges is 5.571 using single neighbor shortest path tree and 4.857 using Algorithm 2. The latter is in fact the optimal solution in this case. For the remainder of this paper, we employ the augmented Steiner trees approach to construct the multicast protection trees.

3.4. Multicast Failure Recovery

The multicast failure recovery procedure uses tunneling (generic routing encapsulation (GRE) [33], IP-in-IP [34]) to reroute packets around the failed link/node instantaneously. The node detecting the outgoing link/node failure tunnels the packet over the appropriate protection trees. Every destination on the tunneled path decapsulates the received encapsulated packet to obtain the originally transmitted multicast traffic, and continues forwarding it along the original multicast tree.

Colored trees and MPTs facilitate every node in the network to autonomously switch to the failure recovery path rather than requesting an upstream node to activate the recovery procedure. Every packet carries with it two bits, referred to as *failure code* (FC). FC bits help identify if a packet has encountered a failure or not. A value of 0 indicates that the packet has not seen any failures. A value of 1 indicates a link failure, while a value of 2 indicates a node failure. These two bits may be used from any reserved bits in the packet header, or alternatively, assigned to the least two significant bits of the destination address. Nodes switching to a link/node protection tree can transmit these decisions via FC bits allowing the downstream nodes to participate in the failure recovery procedure if required. To facilitate failure recovery, every node has four addresses. One address is viewed as primary address (or normal address (NA)) to be used when the network has no failures, while the others are used as alias addresses under certain failure conditions. We refer to the address corresponding to failure code of 1 as the not-via address (NVA) [13] and that corresponding to failure code of 2 as the PA.

The recovery procedure always begins as a link failure recovery procedure and switches to the node failure recovery mechanism when needed. The transition from link failure recovery to node failure recovery is enabled using *link failure* messages. These messages are generated by the node detecting a link failure and are transmitted on the MPT of the downstream node. *Link failure* messages are sent only the first time the corresponding link failure is detected. For instance, when node w detects a failure of the link connected to node v, it generates a *link failure* message destined to PA_v and sends it to node v's neighbors on MPT_v. If a second neighbor of v, say x, observes the failure of link x–v and receives a failure message from w, then it is an indication of multiple link failures around node v. In this work, we assume that if more than one link failure is seen around a node, then the node has failed. A neighbor of v therefore marks node v as failed if:

1. It receives two failure messages from two different neighbors of v or
2. It receives one message from another neighbor of v and has detected a failure of its link to v.

The core of the multicast failure recovery procedure lies in identifying a link/node failure and switching to the corresponding recovery paths seamlessly. Every node in the network selects a forwarding path based on the incoming FC value, the destination address and the outgoing active interface. The steps to forward a packet at a node are shown in Figure 5. Steps 1 and 2 extract the FC and the destination address from the packet and compute the forwarding entry corresponding to these two fields. Steps 3 through 10 show the forwarding procedure when the incoming packet sees a link failure and steps 11 to 30 show the forwarding procedure when a node failure has occurred. At any stage, if no failure is detected on the outgoing links, the FC remains the same and the forwarding path is chosen based on the corresponding destination address.

```
Input: FC and destination address from received packet.
Output: New destination address, outgoing link and FC for outgoing packet.
 1: Extract FC and destination address from received packet
 2: Compute the new destination address, outgoing link and FC based on the following
 3: if FC = 0 then
 4:   if Outgoing link has failed then
 5:     Single link failure scenario
 6:     Send link failure message to all neighbors of downstream node adjacent to
        failed link
 7:     Encapsulate packet into unicast packet with the NVA of the adjacent node
        as the destination address and set FC to 1
 8:     Forward packet along colored trees for single link failure recovery
 9:   end if
10: end if
11: if FC = 1 then
12:   if Outgoing link incident on destination node has failed then
13:     Single node failure scenario
14:     Send link failure message to all neighbors of failed node
15:     Decapsulate received packet
16:     if Original packet is unicast then
17:       Forward packet directly to destination using path not involving failed node
18:     else
19:       if Original packet is multicast then
20:         Forward packet according to routing table entries (original multicast
            packet)
21:         Encapsulate packet with the protection address of the failed node as the
            destination address and set FC to 2
22:         Forward packet according to routing table entries (PA of failed node)
23:       end if
24:     end if
25:   end if
26: end if
27: if FC = 2 then
28:   Node failure has occurred
29:   Forward packet according to routing table entries (PA of failed node)
30: end if
```

Figure 5: Steps to forward a packet at a node.

Once the information about link or node failures, based on the received *link failure* messages are recorded at a node, the node takes into account the number of messages received, in addition to the received FC value to decide which forwarding path to use. For instance, if a node already knows that the downstream node on the original multicast tree has failed, it does not have to send another *link failure* message, and can immediately use the corresponding MPT to forward traffic for the duration of failure recovery. The same applies for link failure recovery too.

Steps 16 and 17 illustrate the actions to be performed when the original packet was a unicast packet. This shows the possibility of developing a single failure recovery procedure for unicast and multicast networks. However, in this paper, we only deal with multicast traffic.

The entire multicast failure recovery mechanism is explained in the following subsection using a particular multicast session in the NSFNET topology.

3.5. Example

Consider the graph in Figure 6(a) which is a revised version of the NSFNET topology for easier analysis. Node 1 is the source (designated source router to which the source host is attached) and nodes 6 and 8 are the destinations (designated destination routers). The directed tree in black in Figure 6(b) is the original multicast tree from the source to the destinations. Under a failure-free scenario, the NA of a node is used and the FCs in all transmitted packets are set to 0. When 3 detects a failed outgoing link to 4 (as in Figure 6(c)), it treats it as a link failure scenario and sends a *link failure* message to nodes 5 and 6 over MPT_4 (the tree connecting nodes 3, 5 and 6). At this point, nodes 5 and 6 know that one incident link on node 3 has failed. They send individual link failure messages over MPT_4 if their incident links on node 3 fail too. Node 3 then encapsulates the received multicast packet into a unicast packet, sets FC to 1 and sends it to the NVA of node 4 along the blue outgoing edge to node 4 as the red outgoing edge was the failed link.

Under a single link failure scenario, node 4 would decapsulate the received not-via packet, recognize the received multicast packet and process it accordingly. The single link failure scenario is no different from that in unicast networks and as illustrated here, the same techniques could be used.

For a single node failure scenario, node 5 detects a failure in the outgoing link on the blue path to node 4 (Figure 6(d)) and sends a link failure message to nodes 6 and 3 over MPT_4. At this point, the neighbors 3, 5 and 6 know that node 4 has failed. Node 5 then decapsulates the received unicast packet, stores a copy of the original multicast packet, re-encapsulates the packet with the destination address set to PA_4 and FC set to 2, and sends the packet across MPT_4. If the original packet was a unicast packet instead of a multicast packet, node 5 would reroute the packet directly to the destination on a path not involving node 4.

At this point node 3 is aware of node 4's failure, and sends future multicast packets directly to PA_4 and FC set to 2 for the duration of the node failure recovery.

4. Performance Evaluation

Any multicast protocol in today's Internet establishes its own multicast tree to route traffic from the source to the destinations. During a link/node outage, the fast reroute procedure is instantaneously activated by the node detecting this failure. Fast reroute mechanisms guarantee lossless transmissions at the cost of maintaining extra information at participating nodes. The following subsections look at these overheads to determine the feasibility of the multicast failure recovery approach proposed in this paper. They are computed over commonly used topologies like the NSFNET (Figure 7(a)), NJLATA [35] (Figure 7(b)), Modified NJLATA (NJLATA with a few links removed, Figure 7(c)) and ARPANET (Figure 7(d)). Modified NJLATA has more nodes with degree 2 when compared to the regular NJLATA network. This modified topology is used to better understand the overheads incurred while using MPTs during node failure recovery. All these networks are 2-vertex connected.

The entire multicast failure recovery procedure is a combination of the single link and single node failure recovery algorithms. We measure the overheads incurred by the colored trees for single link failure recovery and the MPTs for single node failure recovery. Apart from these overheads, we also measure the average path lengths between any source and destination before, during and after the failure recovery procedure. The path lengths measured during the failure recovery procedure indicate the amount of additional network resources that the colored trees or the MPTs require during failure recovery. We denote this as the network utilization in both cases. And finally, we also evaluate the ability for these fast reroute schemes to stick to the original idea of multicast, i.e. transmit only a single copy of data over a shared link.

4.1. Single Link Failure Recovery

Colored trees rooted at a node are used to recover from the failure of a link incident on that node. This paper helps understand that no new techniques are required to recover from link failures in multicast networks, and that established fast reroute techniques for unicast networks can be used here. Though we illustrate only single link failure recovery, double link failures could be recovered from too using unicast mechanisms, provided the network is at least 3-edge connected. We examine the amount of additional information that has to be maintained at each node, and the network utilization during single link failure recovery.

4.1.1. Overhead due to Colored Trees

The red and blue forwarding paths at a node require only two additional routing table entries per destination node. As the multicast link failure recovery procedure reuses colored trees that are developed for unicast networks, no new information is required for multicast single link failure recovery. It is also interesting to note that a node needs the colored tree information only for its neighbors as it only recovers from failed links incident on its neighbors.

Table 2: Network utilization - single link failure recovery.

Topology	**NSFNET**	**NJLATA**	**Modified NJLATA**	**ARPANET**
Average path length before single link failure	2.14	1.75	2.04	2.79
Average path length during single link failure recovery	6.05	3.39	4.42	7.02
Average path length after single link failure recovery	3.54	2.46	3.18	4.33

4.1.2. Network Utilization

During single link failures, the recovery path from any source to a destination on the multicast tree is the combination of the paths from the source to the node detecting the failure, the fast reroute path between the upstream and downstream nodes of the failed link and the path from the downstream node to the destination. We measure the average path length of all possible combinations of recovery paths to understand the amount of extra links that would be used by the colored trees during failure recovery.

Let $\mathcal{P}_{u,v}$ indicate the shortest path from any node u to any node v and let $\mathcal{P}^{NV}_{u,v}$ indicate the not-via fast reroute path from any node u to any node v. In the original path from source s to destination t on any multicast tree, $s \rightarrow x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow x_4 \rightarrow t$, if link $x_2 \rightarrow x_3$ fails, the recovery path length is computed as

$$|\mathcal{P}^R_{s,t}| = |\mathcal{P}_{s,x_2}| + |\mathcal{P}^{NV}_{x_2,x_3}| + |\mathcal{P}_{x_3,t}| \tag{1}$$

.

To calculate the network utilization during failure recovery, every intermediate link on the path from s to t is put down and the recovery path length is calculated. The calculated recovery path length, averaged over all combinations of (s,t) gives the network utilization as in Table 2.

The average path lengths during single link failure recovery using colored trees are approximately 1.5 times longer than the shortest path that would be calculated between any two nodes s and t after failure recovery. We could obtain shorter path lengths by constructing link protection trees specifically for multicast failure recovery. For instance, if the red forwarding path at u is the link that is directly incident on the neighbor v, then the blue forwarding path from u would be the shortest path from u to v obtained on graph $G(V,E); E = E \setminus (u,v)_{red}$, where $(u,v)_{red}$ is the red forwarding path from nodes u to v. In the example figures, Figure 8(a) and Figure 8(b) below, if the red link $3 \rightarrow 0$ fails, the ideal fast reroute path would be $3 \rightarrow 4 \rightarrow 0$ but instead the single link failure recovery algorithm chooses the blue path $3 \rightarrow 7 \rightarrow 6 \rightarrow 2 \rightarrow 1 \rightarrow 0$.

The goal is to use fast reroute paths already established for unicast networks, hence the longer paths. Reusing existing protection trees facilitates quicker deployment of multicast failure recovery from link failures.

4.2. Single Node Failure Recovery

The MPTs provide fast reroute paths during a single node failure in multicast networks at the cost of maintaining extra information. The overheads incurred are due to,

1. Maintaining the MPT infrastructure and
2. Flooding fast reroute packets to unwanted downstream neighbors that are not part of the original multicast tree.

These two will be discussed in the following sections. As before, we also measure the network utilization under a single node failure scenario.

4.2.1. Overhead due to the MPT Infrastructure

In order to route over the MPTs, every node has additional routing entries. These entries correspond to the MPTs that every node is a part of. In Table 3 we calculate the maximum and average number of additional routing entries that have to be maintained at each node in all four topologies. In NSFNET and ARPANET, the degrees of the nodes are less than 5 (for a total of 14 and 20 nodes respectively) and hence each node would be part of multiple MPTs. However, owing to large degrees at nodes 4 and 7 in NJLATA, the number of additional routing entries is smaller. Given that many real ISP networks are approximately three connected [36], this overhead will be tolerable as indicated in Table 3.

4.2.2. Flooding Overhead due to Non-Participating Downstream Neighbor Recipients

In the original multicast tree, all the downstream neighbors of the failed node may not be part of the downstream portion of the ongoing multicast session (such as node 4's neighbor node 5 in Figure 6(b)). This leads to unnecessary overhead while routing to non-participating downstream nodes that belong to the MPT of the failed node. The anonymity of the participating downstream neighbors to the upstream neighbor of the failed node (the node detecting the failure) causes this overhead. We evaluate the extent of this overhead on the four topologies in Figure 7. The overhead measures the ratio of the total number of links used (total number of edges in the MPT) to the number of links required to connect the participating neighbors on all MPTs in the network. This ratio is calculated for the entire topology by considering all combinations of destinations with a single source over all MPTs in the network. The sum of required number of edges over all combinations of a single MPT would be less than or equal to

Table 3: MPT overhead during single node failure recovery.

Topology		NSFNET	NJLATA	Modified NJLATA	ARPANET
(Number of vertices, number of edges)		$(14, 21)$	$(11, 23)$	$(11, 17)$	$(20, 32)$
Additional routing table entries	**Maximum**	10	8	7	10
	Average	5.86	4.18	4.18	5.85
Overhead due to unnecessary flooding		1.39	1.61	1.59	1.40

the sum of number of edges in the MPT for all combinations (Note: the total number of edges in the MPT will be the same for all combinations). The ratio computed in this section is a conservative estimate of the flooding overhead as we do not disregard non-participating neighbors of the failed node that could be on the path connecting the participating neighbors. Hence, the ratios computed in this section indicate the worst possible overhead that can occur.

From Table 3, for the NSFNET topology where the degree of any node is not more than 4, the flooding overhead is only 39% on average. This is representative of networks that do not have a small number of *high* degree nodes as seen in the NJLATA topology. Node 4 for instance has degree 8 (for a total of 11 nodes in the network). This leads to larger overheads when the MPT_4 is used to connect only a few intended recipients (neighbors of node 4 in the downstream section of the original multicast tree). Highly connected networks would not have this issue owing to smaller MPTs but networks with few high degree nodes such as NJLATA would bear the brunt. This overhead is caused only while the new multicast tree is being formed and should not affect the network performance. Optimum solutions could be obtained by analyzing the membership pattern of the network. Modified NJLATA, on account of slightly lesser connectivity at nodes 4 and 7 benefits by about 2% on average over NJLATA. The ARPANET topology has similar characteristics as the NSFNET network owing to similar neighborhood sizes at each node.

4.2.3. Network Utilization

We measure the impact of MPTs on the network resource utilization by comparing the average path lengths between any (source, destination) pair during three scenarios, (1) on the original multicast tree when no failure occurs, (2) during the node failure recovery when the MPTs are used and (3) after the

Table 4: Network utilization - single node failure recovery.

Topology	**NSFNET**	**NJLATA**	**Modified NJLATA**	**ARPANET**
Average path length before single node failure	2.49	2.28	2.51	3.16
Average path length during single node failure recovery - NV and MPT	8.24	4.24	5.69	9.71
Average path length during single node failure recovery - only MPT	4.36	2.91	3.56	5.62
Average path length after single node failure recovery	3.57	2.68	3.43	4.64

node failure recovery when a new multicast tree has been established. This is evaluated on the topologies in Figure 7.

Let $\mathcal{P}_{u,v}$ indicate the shortest path from any node u to any node v and let $\mathcal{P}_{u,v}^{MPT_w}$ indicate the fast reroute path from any node u to any node v on MPT$_w$. In the original path from source s to destination t on a multicast tree, $s \to x_1 \to x_{f_1} \to x_f \to x_{f_2} \to x_2 \to t$, if node x_f fails, the recovery path length is computed as:

$$|\mathcal{P}_{s,t}^R| = |\mathcal{P}_{s,x_{f_1}}| + |\mathcal{P}_{x_{f_1},x_{f_2}}^{MPT_{x_f}}| + |\mathcal{P}_{x_{f_2},t}| \tag{2}$$

It is to be noted that:

$$|\mathcal{P}_{s,t}^R| = |\mathcal{P}_{s,x_{f_1}}| + |\mathcal{P}_{x_{f_1},x_{f_3}}^{NV}| + |\mathcal{P}_{x_{f_3},x_{f_2}}^{MPT_{x_f}}| + |\mathcal{P}_{x_{f_2},t}| \tag{3}$$

is the length traversed by the first few packets during node failure recovery, where x_{f_3} is another neighbor of x_f that detects the second link failure. Once x_{f_1} recognizes x_f's failure, the following packets go over the MPT alone without going through the link failure recovery procedure first.

To calculate the network utilization during failure recovery, every intermediate node on the path from s to t is put down and the recovery path length is calculated. The calculated recovery path length, averaged over all combinations of (s,t) gives the network utilization.

Table 4 lists the average path length for single node failure scenarios before, during and after node failure recovery. For this analysis we only consider paths whose lengths are greater than or equal to 2. From Table 4, we see that the average path lengths during node failure recovery using MPTs alone are quite close to those obtained after the failure recovery procedure. This is particularly evident in NJLATA and Modified NJLATA that have nodes with high connectivity. This is because, in most cases, the recovery path from the upstream to the downstream node of the failed on the MPT corresponds to their second shortest path. In networks like NJLATA where the average path length between any two nodes is as small as 2.28, the difference in path lengths during and after failure recovery would be minimal.

4.2.4. Benefits of the proposed technique

In comparison to the existing mechanisms, the use of MPTs provide the following advantages: (1) The MPTs are constructed a priori, independent of the original multicast tree, unlike most existing schemes that construct backup paths only after the original multicast tree is set up. This enables protocol-independent node failure recovery. (2) Every node has only one protection tree that can used by any multicast session unlike existing schemes that construct multiple session-specific back-up paths for the same node. Therefore, using MPTs significantly reduces the backup path maintenance cost incurred in other approaches. (3) Every node can independently initiate node failure recovery without requiring deployment of special nodes to help recover multicast traffic. All these factors result in guaranteeing instant node failure recovery for multicast traffic.

4.3. Closeness to Required Multicast Behavior

In this section, we evaluate a new metric that measures the extent to which the recovery techniques divert from the original multicast behavior. The multicast communication model guarantees a single transmission of identical data over a link that is common to multiple recipients. However, during failure recovery, the colored trees and the MPTs divert the original multicast packets over alternate paths that could result in multiple transmissions of identical data over a common link. For this bidirectional network, transmission in either direction on a link are considered non-identical.

Property 1. *The number of times that identical data is transmitted simultaneously over a single link is limited to* 2 *in the local area of failure.*

Proof. The original multicast tree T_M and the fast reroute paths (during single link/node failure recovery) which are also trees denoted by T_{FRR}, together constitute two trees that connect a subset of nodes N_M and N_{FRR} respectively. For every node pair (u, v): $u, v \in N_M \cap N_{FRR}$, the directed edge between nodes u and v on both trees would either be in the same direction or in opposite directions. The maximum overlap on a link occurs when the edges in the two trees between the same pair of adjacent nodes are in the same direction which is 2. □

In most cases, the overlap occurs only on one or two links for the duration of failure recovery, and standards such as DOCSIS 3.0 should be able to support multiple high-bandwidth transmissions over a single link in the backbone networks [19].

5. Conclusion

In this paper, a fast rerouting mechanism to recover from single link and single node failures is developed for multicast networks. We make the following observations and conclusions about the proposed failure recovery scheme.

1. Single link failure recovery in multicast networks is the same as that in unicast networks and no new infrastructure is required. The single link fast reroute scheme makes use of the colored trees that are set up for unicast networks, to bypass the failed link. For networks with higher edge connectivity, double link failures can be recovered from in a similar manner.
2. We propose MPTs that span the neighbors of a node. The MPTs provide a fast reroute path around the failed node by connecting the upstream and downstream nodes. Two centralized algorithms to compute these MPTs are discussed.
3. Both the colored trees and the MPTs require no knowledge of the original multicast tree and can be computed in advance. They are independent of the multicast protocol being implemented.
4. The node detecting the link/node failure can activate the appropriate protection trees without depending on specific upstream or downstream nodes in the network. FC bits are used to indicate the type of failure detected.
5. The MPTs require about 38% more routing table entries at a node on average. This value is obtained over four different topologies. The average path lengths between any (source, destination) pair during single node failure recovery is only 13% more than that which is computed after the multicast protocol has re-converged
6. We show that the maximum number of edge overlaps that can occur during single link and single node fast reroute is 2.
7. The MPTs with certain optimizations could be used for single node failure recovery in unicast networks too. Both the colored trees and the MPTs set the base for protocol independent single link and single node fast reroute in multicast and unicast networks.

Acknowledgment

The research developed in this paper is funded by the National Science Foundation under grant CNS-1117274.

References

[1] A. Sundarrajan, S. Ramasubramanian, Fast rerouting for IP multicast under single node failures, in: IEEE Global Communications Conference, 2013, pp. 2076–2081.

[2] B. Fenner, M. Handley, H. Holbrook, I. Kouvelas, Protocol independent multicast - sparse mode (PIM-SM), RFC 4601 (Aug. 2006).

[3] S. Bhattacharya, An overview of source-specific multicast (SSM), RFC 3569 (Jul. 2003).

[4] A. Adams, J. Nicholas, W. Siadak, Protocol independent multicast - dense mode (PIM-DM), RFC 3973 (Jan. 2005).

[5] A. Ballardie, Core based trees (CBT) multicast routing architecture, RFC 2201 (Sept. 1997).

[6] B. Cain, S. Derring, I. Kouvelas, B. Fenner, A. Thyagarajan, Internet group management protocol, version 3, RFC 3376 (Oct. 2002).

[7] B. Fenner, D. Meyer, Multicast source discovery protocol (MSDP), RFC 3618 (Oct. 2003).

[8] T. Bates, R. Chandra, D. Katz, Y. Rekhter, Multiprotocol extensions for BGP-4, RFC 4760 (Jan. 2007).

[9] Cisco visual networking index : Forecast and methodology, 2012 - 2017 (white paper).

[10] X. Wang, C. Yu, H. Schulzrinne, P. A. Stirpe, W. Wu, IP multicast fault recovery in PIM over OSPF, in: ICNP, 2000, pp. 116–125.

[11] M. Shand, S. Bryant, IP fast reroute framework, RFC 5714 (Jan. 2010).

[12] S. Kini, S. Ramasubramanian, A. Kvalbein, A. Hansen, Fast recovery from dual-link or single-node failures in IP networks using tunneling, IEEE/ACM Transactions on Networking 18 (6) (2010) 1988 –1999.

[13] S. Bryant, S. Previdi, M. Shand, A framework for IP and MPLS fast reroute using not-via addresses, Internet-Draft draft-ietf-rtgwg-ipfrr-notvia-addresses-10 (Dec. 2012).

[14] M. Gjoka, V. Ram, X. Yang, Evaluation of IP fast reroute proposals, in: 2nd International Conference on Communication Systems Software and Middleware, 2007, pp. 1–8.

[15] G. Jayavelu, S. Ramasubramanian, O. Younis, Maintaining colored trees for disjoint multipath routing under node failures, IEEE/ACM Transactions on Networking 17 (1) (2009) 346–359.

[16] M. Médard, S. G. Finn, R. A. Barry, Redundant trees for preplanned recovery in arbitrary vertex-redundant or edge-redundant graphs, IEEE/ACM Transactions on Networking 7 (5) (1999) 641–652.

[17] R. Doverspike, G. Li, K. Oikonomou, K. Ramakrishnan, D. Wang, IP backbone design for multimedia distribution: Architecture and performance, in: IEEE INFOCOM, 2007.

[18] J. Wu, K. Shin, SMRP: Fast restoration of multicast sessions from persistent failures, in: International Conference on Dependable Systems and Networks, 2005, 2005, pp. 150–159.

[19] J. Godas, B. Field, A. Bernstein, S. Desai, T. Eckert, H. Parandekar, Ip multicast in cable networks (white paper).

[20] M. Yuksel, K. K. Ramakrishnan, R. D. Doverspike, R. K. Sinha, G. Li, K. N. Oikonomou, D. Wang, Cross-layer failure restoration of IP multicast with applications to IPTV, Computer Networks, Elsevier Science 55 (2011) 2329–2351.

[21] R. Luebben, G. Li, D. Wang, R. Doverspike, X. Fu, Fast rerouting for IP multicast in managed IPTV networks, in: 17th International Workshop on Quality of Service, 2009, 2009, pp. 1–5.

[22] R. Doverspike, G. Li, K. Oikonomou, K. Ramakrishnan, R. Sinha, D. Wang, C. Chase, Designing a reliable IPTV network, Internet Computing, IEEE 13 (3) (2009) 15–22.

[23] L. Wei, A. Karan, N. Shen, Y. Cai, M. Napierala, Tunnel based multicast fast reroute (TMFRR) extensions to PIM, Internet-Draft draft-1wei-pim-tmfrr-00 (Oct. 2009).

[24] N. Wang, B. Dong, Fast failure recovery for reliable multicast-based content delivery, in: International Conference on Network and Service Management (CNSM), 2010, pp. 505 –510.

[25] A.-W. Tam, K. Xi, H. Chao, A fast reroute scheme for IP multicast, in: IEEE GLOBECOM, 2009, pp. 1–7.

[26] A. Karan, S. Filsfils, D. Farinacci, B. Decraene, N. Leymann, U. Joorde, W. Henderickx, Multicast only fast re-route, Internet-Draft draft-karan-mofrr-02 (Mar. 2012).

[27] I. Wijnands, A. Csaszar, J. Tantsura, Tree notification to improve multicast fast reroute, Internet-Draft draft-wijnands-rtgwg-mcast-frr-tn-00 (Oct. 2012).

[28] S. Ramasubramanian, M. Harkara, M. Krunz, Distributed linear time construction of colored trees for disjoint multipath routing, in: Proceedings of IFIP Networking, 2006, pp. 1026–1038.

[29] F. K. Hwang, D. S. Richards, P. Winter, The Steiner Tree Problem (Annals of Discrete Mathematics), North-Holland, 1992.

[30] R. M. Karp, Reducibility among combinatorial problems, in: Complexity of Computer Computations, 1972, pp. 85–103.

[31] M. L. Fredman, R. E. Tarjan, Fibonacci heaps and their uses in improved network optimization algorithms, Journal of the ACM 34 (3) (1987) 596–615.

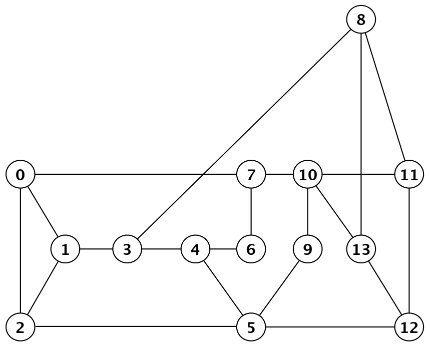
[32] H. Takahashi, A. Matsuyama, An approximate solution for the Steiner tree problem in graphs, Math. Japonica 24 (6) (1980) 573–577.

[33] S. Hanks, D. Farinacci, P. Traina, Generic routing encapsulation (GRE), RFC 1701 (Oct. 1994).

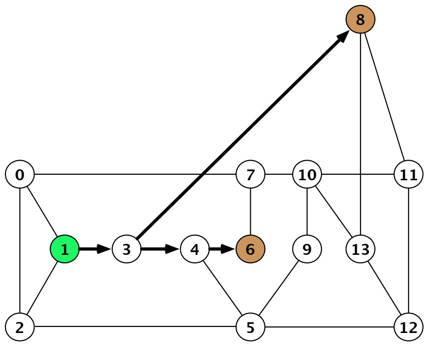
[34] C. Perkins, IP encapsulation within IP, RFC 2003 (Oct. 1996).

[35] P. Singh, A. K. Sharma, S. Rani, Minimum connection count wavelength assignment strategy for WDM optical networks, Optical Fiber Technology 14 (2) (2008) 154–159.

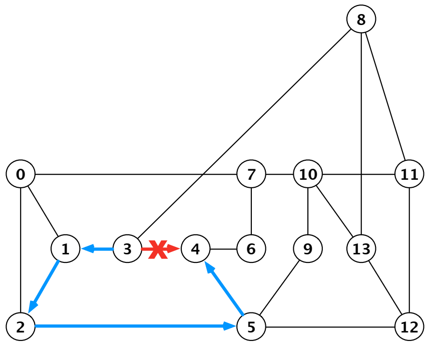
[36] N. Spring, R. Mahajan, D. Wetherall, T. Anderson, Measuring ISP topologies with rocketfuel, IEEE/ACM Transactions on Networking 12 (1) (2004) 2–16.



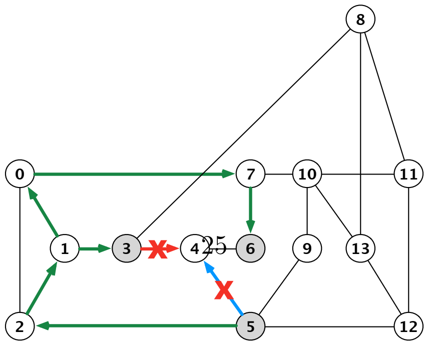
(a) Modified NSFNET topology



(b) Example multicast tree

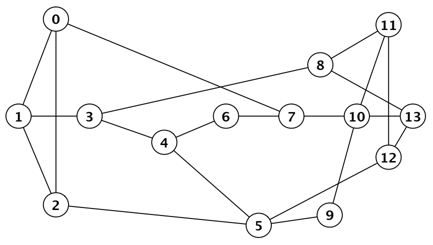


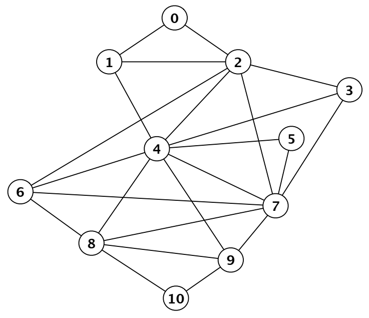
(c) Link 3-4 fail

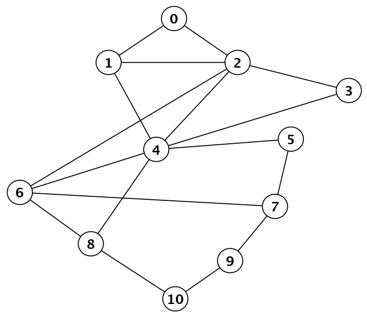


(d) Link 5-4 fail

Figure 6: Multicast failure recovery example.


(a) NSFNET


(b) NJLATA


(c) Modified NJLATA

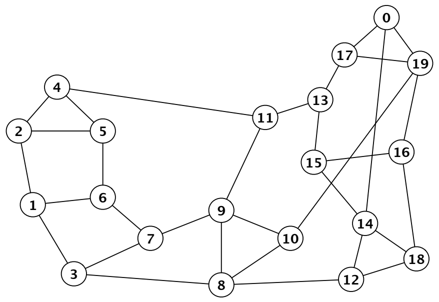
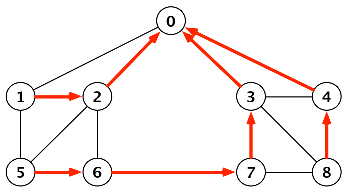

(d) ARPANET

Figure 7: Network topologies used for performance evaluation.


(a) Red Tree at Node 0

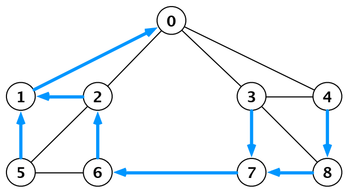

(b) Blue Tree at Node 0

Figure 8: Longer fast reroute paths during multicast single link failure recovery.