

# Independent-Tree Ad hoc Multicast Routing (ITAMAR)

S. Sajama

School of Electrical Engineering  
Cornell University  
Ithaca, NY 14853

E-mail: [sajama@ece.cornell.edu](mailto:sajama@ece.cornell.edu)

<http://www.ece.cornell.edu/~sajama>

Zygmunt J. Haas

School of Electrical Engineering  
Cornell University  
Ithaca, NY 14853

E-mail: [haas@ece.cornell.edu](mailto:haas@ece.cornell.edu)

<http://www.ee.cornell.edu/~haas/wnl.html>

*Abstract*—Multicasting is an efficient means of one to many communication and is typically implemented by creating a *multicasting tree*. Because of the severe battery power and transmission bandwidth limitations in ad hoc networks, multicast routing can significantly improve the performance of this type of networks. However, due to the frequent and hard-to-predict topological changes of ad hoc networks, maintenance of a multicasting tree to ensure its availability, could be a difficult task. We borrow from the concept of Alternate Path routing, which has been studied for providing QOS routing, effective congestion control, security, and route failure protection, to propose a scheme in which a set of multicasting trees is continuously maintained. In our scheme, a tree is used until it fails at which time it is replaced by an alternative tree in the set, so that the time between failure of a tree and resumption of multicast routing is minimal. In this paper, we introduce the scheme and present a number of heuristics to compute a set of alternate trees. The heuristics are then compared in terms of transmission cost, improvement in the average time between multicast failures and the probability of usefulness. Simulations show significant gains over a wide range of network operational conditions. In particular, we show that using alternate trees has the potential of improving mean time between interruption by 100-600% in a 50 node network (for most multicast group sizes) with small increase in the tree cost and the route discovery overhead.

## I. INTRODUCTION

An ad hoc network consists of a collection of mobile routers which are interconnected via wireless links and are free to move about arbitrarily. Multicasting is an efficient communication tool for use in multi point applications which cover a very wide spectrum including replicated database update and audio/video conferencing.

Multicasting in ad hoc networks is more challenging than in the Internet, because of the need to optimize the use of several resources simultaneously. Nodes in an ad hoc network are battery and bandwidth limited, move very fast causing links to fail rapidly and do not have a central control point.

Work on multicast routing in ad hoc networks gained momentum in the mid 90s. Some early approaches to provide multicast support in ad hoc networks consisted of adapting the existing Internet multicasting protocols (e.g. Shared Tree Wireless network Multicast [1]). Others like ODMRP [2], AMRIS [3], CAMP [4] and [5] have been designed specifically for ad hoc networks.

This work has been sponsored in part by ONR contract no. N00014-00-1-0564, AFRL contract no. F360602-97-C-0133, and NSF grant no. ANI-9980521.

One common characteristic of most of these approaches is that they react to a link failure; i.e they act **after** a link has already failed, causing a significant delay in route recovery. In our work, we have explored the possibility of using a set of precalculated alternate trees using the information (about network topology) acquired to calculate the first tree. When a link breaks, another tree, which does not include that link, can be immediately utilized. This leads to minimum possible delay, whenever a viable backup tree is available at the time of failure of the current tree. In particular, it allows communication of real-time traffic. This approach is inspired by alternate path routing, which has been used in the Internet to alleviate congestion and to improve QOS. Recently, performance gain that can be obtained from use of APR in ad hoc networks for unicast routing has been investigated [6].

## II. GOALS AND BASIC IDEAS

The goal of this work is to improve multicasting performance in ad hoc networks by efficient use of the available knowledge of the network. The basic idea is that if we are able to compute multiple backup multicast trees with minimal overlap, we could use them one after another to reduce the number of service interruptions. This would also improve the mean time between route discovery cycles for a given interruption rate and hence reduce the control overhead and the rate of data loss. At the same time, we want to keep the cost of transmission low. The mobility of ad hoc networks requires that we use very little time for tree computation and hence it is important that the algorithms be of low complexity.

This method of using one tree after another will be effective if the trees to be used as backup last for a significant amount of time after the previous trees fail. This means that the failure times of the trees should be independent of one another. If we assume that nodes move independently of one another, having no common nodes (and hence no common edges) would make the trees fail independently of one another. The **dependence** of a pair of trees is defined as the correlation of the failure times of the two trees. Dependence of a pair of trees is a complicated function of the mobility pattern of the nodes. Hence a practical way to compute independent enough trees would be to discourage common edges and nodes among the trees.

### III. SCHEMES FOR COMPUTING MAXIMALLY INDEPENDENT TREES

Two ways of the using a backup tree set upon link failure are : 1) replace the whole tree being used currently by a backup tree, if available or 2) replace or augment paths to only those nodes which cannot be reached because of this link failure. This section details several algorithms to compute backup trees depending on how they are going to be used. The network connections are represented by a graph G and the trees to be found are referred to as T1, T2 and so on. T1 is intended to be used at the start and the others are to be used as backup. The set of all edges in the graph along with a quantity called cost of each edge is called the *Cost Function of the graph*

#### A. Computing Backup Trees

##### A.1 Matroid Intersection Heuristic

The Matroid Intersection algorithm (found in books on combinatorial optimization [9]) can be used to find two maximally independent spanning trees on any given graph (i.e, spanning trees with minimum possible number of common edges) such that total cost of the two spanning trees is minimized. The two spanning trees obtained are called J1 and J2. Given a sender (call it the source node) and a set of receivers, two multicasting trees T1 and T2 are obtained on graphs J1 and J2 respectively using Dijkstra algorithm. When it is not possible to have 2 completely edge-disjoint spanning trees, the above algorithm will give 2 trees edge disjoint trees with maximal cardinality (These might not be spanning trees as adding any more edges might require overlap between the 2 trees). Hence now to complete each tree, we arrange the links in the other tree in ascending order of their costs and keep adding links to the first tree (unless they form circuits) until the first tree is complete and vice versa. Multicast trees generated in this way may not have minimum possible number of common edges, though the spanning trees do have this property. Also, this scheme can be used only to obtain one backup tree, because the problem of finding intersection of 3 Matroids is NP-Hard.

##### A.2 Shortest Path Heuristic

In this scheme, the first tree, T1, is the shortest path tree from the source to the set of receivers. The Cost function of the graph is modified after computing the first tree in the following manner - cost associated with edges used in T1 is increased by the parameter Link Weight and the cost associated with edges which share a common node with T1 is increased by the parameter Node Weight. T2 is computed using the original incidence matrix of the graph and this new cost function. Since Dijkstra algorithm tries to use edges of the lowest cost, this way of modifying the Cost function discourages use of the edges and nodes already used in T1 (the extent of the discouragement depends on the values of the parameters Link Weight and Node Weight). Computation of subsequent backup trees is carried out by discouraging the use of links and nodes already used in previous trees by *further* modification of Cost Function.

#### Shortest Path Heuristic Algorithm

```

T1=Dijkstra_Algorithm(G,Cost,Source,Receivers)
Initialize Cost1 to equal Cost for all edges in G
For each edge i in T1
    { Cost1i = Costi + LinkWeight }
For each node in T1
    {
        For each link in G incident on this node in T1
            { Cost1i = Costi + NodeWeight }
    }
T2=Dijkstra_Algorithm(G,Cost1,Source,Receivers)

```

#### A.3 Low Cost Heuristic

Low Cost Heuristic algorithm is designed to reduce the the total number of transmissions in the multicast trees. In this method each of the trees are constructed path by path. **Computation of a tree** given an initial Cost function is done in the following way - path to a node is computed, cost function is modified and then path to next node is computed and added to the partial tree already constructed and so on. The modification of Cost Function in between computing paths to each receiver is done in such a way as to encourage use of minimum number of additional transmissions; i.e. if a link already carries the multicast data , its transmission cost is decreased to a very small value. The cost function taken at the beginning of computation of second tree is a modified version of the original cost function of the tree done in order to discourage use of links and nodes already used in prior trees (for details of modification look at description in Shortest Path Heuristic). Computation of subsequent backup trees is carried out by discouraging use of links and nodes already used in previous trees by modification of Cost Function.

#### Low Cost Heuristic Algorithm

```

Initialize Cost' to Cost
For each receiver j
    {
        Pj=Dijkstra_Algorithm(G,Cost',Source,j)
        For each edge in P1 ⊕ ... ⊕ Pj
            { Cost'i = 0 }
        For each node in P1 ⊕ ... ⊕ Pj
            {
                For each link in G incident on this node
                in P1 ⊕ ... ⊕ Pj
                    { Cost'i = ε }
            }
    }
T1 = P1 ⊕ P2 ⊕ ... ⊕ PN
Initialize Cost1 to Cost
For each edge i in T1
    { Cost1i = Costi + LinkWeight }
For each node in T1
    {
        For each link in G which is incident on this
        node in T1
            { Cost1i = Costi + NodeWeight }
    }
Initialize Cost' to Cost1
For each receiver j
    {

```

```

P'_j = Dijkstra_Algorithm(G, Cost', Source, j)
For each edge in P'_1 ⊕ ... ⊕ P'_j
  { Cost1_i = 0 }
For each node in P'_1 ⊕ ... ⊕ P'_j
  {
  For each link in G which is incident on
  this node in P'_1 ⊕ ... ⊕ P'_j
    { Cost1_i = ε }
  }
}
T2 = P'_1 ⊕ P'_2 ⊕ ... ⊕ P'_N

```

### B. Computing Backup Paths

Independent Path Algorithm computes trees such that paths to each receiver in these trees are disjoint while allowing paths to different receivers to overlap across trees. The disadvantage of using trees as backup is that even if just one link in the tree fails, we need to replace the whole tree by another, when most of the first tree may be intact. Instead, in Independent Path Algorithm, we start off with a tree and then for each receiver, have a set of backup paths which are maximally disjoint from one another and from the path to the receiver in the first tree. The first tree can be computed using either Dijkstra Algorithm or using the Low Cost Heuristic. For each receiver, a path independent of the original path to the node in the first tree is computed by modifying the Cost Function (as in the Shortest Path Heuristic) in order to discourage use of already used nodes and edges.

#### Independent Path Algorithm

```

T1 = Dijkstra_Algorithm(G, Cost, Source, Receivers)
Initialize Cost1 to equal Cost
For receiver node k
  {
  For link i in path (in T1) node k
    { Cost1_i = Cost_i + LinkWeight }
  For each node in path (in T1) node k
    {
    For each link in G incident on this node in T1
      { Cost1_i = Cost_i + NodeWeight }
    }
  Backup Path to k = Dijkstra(G, Cost1, Source, k)
  }

```

This method differs from the backup tree methods not only in that it replaces only the damaged part of the tree (local repair) but also in that the backup path to any given receiver can overlap with the rest of the first tree (apart from what is being used to transmit data to that receiver). It is more likely to find paths independent from a given path rather than one independent from a given tree. As in the previous two methods, computation of subsequent backup trees is carried out (similar to the second tree) by discouraging use of links and nodes already used in previous trees by modification of Cost Function.

## IV. CRITERIA USED FOR PERFORMANCE COMPARISON

### A. Cost

A number, cost  $c_i$ , is associated with each link  $i$  in the graph. As in traditional networks, it could be chosen to be inversely

proportional to the link capacity, proportional to the current load, etc. The cost of a tree is defined as the sum of costs of all the links in the tree. The cost of a set of trees is defined as the sum of costs of all the trees in the set. For a given multicast group size, the Average Cost of a scheme is the weighted average of the cost of all the trees being computed, weighted by the average amount of time each of the trees is being used; i.e., it is

$$\text{Average Cost} = \frac{\sum_{i=1}^n C_{treei} * T_{treei}}{\sum_{i=1}^n T_{treei}}$$

### B. Dcost

The idea behind defining Dcost is that in a single channel wireless network, the MAC layer is naturally of broadcast type. In other words, when a node transmits, all its neighbors are able to listen to it. Hence the cost of transmission of information to all neighbors from one node is the same as the cost of transmission to the most "expensive" neighbor. The Dcost of a tree is the total number of transmissions required for the data to reach all receivers. The Average Dcost of a scheme is the weighted average of the Dcost of all the trees being computed, weighted by the average amount of time each of the trees is being used; i.e., it is

$$\text{Average Dcost} = \frac{\sum_{i=1}^n Dcost_{treei} * T_{treei}}{\sum_{i=1}^n T_{treei}}$$

### C. Time of Failure or Mean Time Between Interruptions

Time of failure of a tree is the minimum time by which at least one of the links of the tree fails and the time of failure of the system is the minimum time at which all paths, (in the first and the backup trees) to at least one of the multicast receivers fail. We use the terms - system time and the mean time between interruption, interchangeably since an interruption occurs whenever there is a failure of all the trees triggering re-computation of trees.

### D. Probability of usefulness

The probability that the backup set computed by any of the above schemes will be used is defined as the probability of usefulness. It is that fraction of the total number of trials for which failure time of the system is greater than failure time of the first tree.

## V. SIMULATION RESULTS AND DISCUSSION

### A. The Simulation Environment

50 Nodes are uniformly distributed over a square area of size 700 meters by 700 meters. Each node can exchange information with any other node within 140 meters of themselves. At time 0, with probability 0.5 they pick a destination point (which is also uniformly distributed in the area) and start moving in that direction with velocity 40 [m/s] and with Probability 0.5 they wait in their positions for a random amount of time (uniformly distributed over 0-5 seconds) before choosing a destination. After reaching their destination point, they stop with Probability 0.5 in their positions for a random amount of time (uniformly distributed over 0-5 seconds) and choose another destination point and start moving in the new direction with probability 0.5. The nodes were allowed to move according to the above mobility model until the multicast tree and

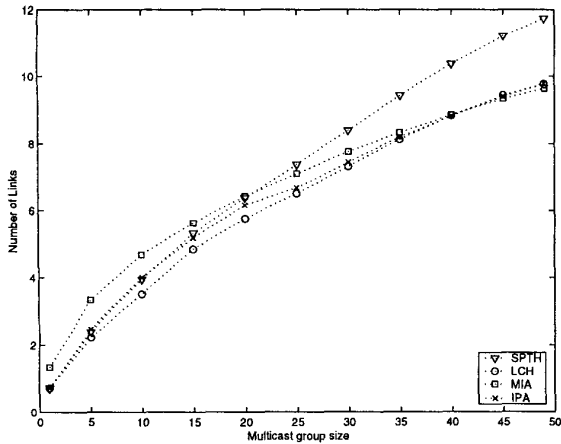


Fig. 1. Average Cost (One backup Tree)

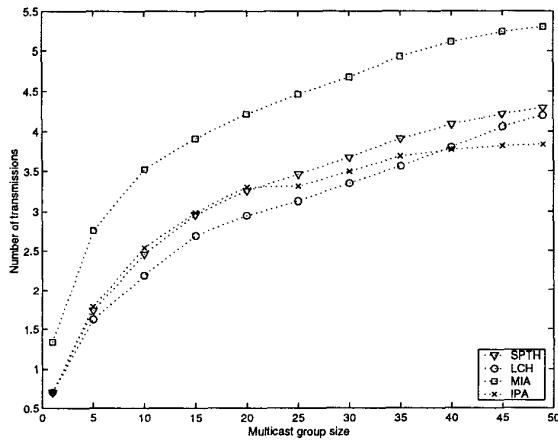


Fig. 2. Average Dcost (One backup Tree)

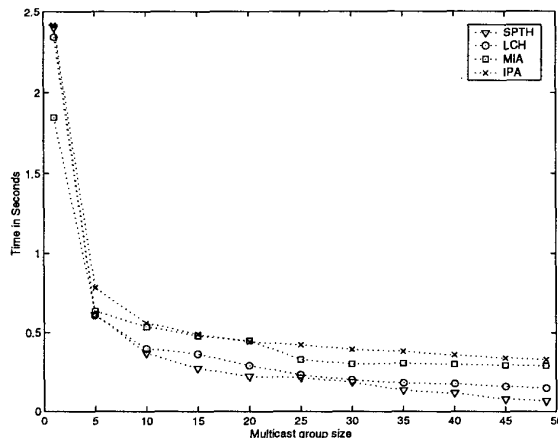


Fig. 3. Time of Failure of the System (One backup Tree)

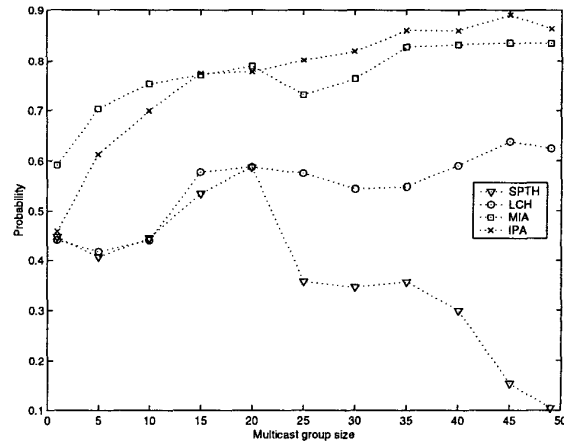


Fig. 4. Probability of usefulness (One backup Tree)

all backups had failed and the failure times of various schemes were recorded. The data presented here are averaged over 500 different trees (each under 25 realizations of the mobility pattern) for each multicast group size. In all the graphs in this Section, SPTH refers to Shortest Path Tree Heuristic, LCH refers to Low Cost Heuristic, IPA refers to Independent Path Algorithm and MIA refers to Matroid Intersection Algorithm.

#### B. Results for one backup tree

Figure 1 shows that the average Cost of trees used is not very different for the various schemes. However, from Figure 2 we see that the average number of transmissions required per packet (the Dcost) is significantly higher for Matroid intersection algorithm while the Dcost curves for other three schemes are relatively bunched together. This is because of the fact that the Matroid Intersection Algorithm, in the process of making the two spanning trees edge disjoint, causes links incident on any given node to be distributed among the two spanning trees. Hence, there are less number of links incident on any given node in each of these spanning trees when compared to the whole network. Because of this, each multicast tree (which is computed on these spanning trees as described in section 4.1.1) has less number of outgoing links to choose from at each node and hence has greater number of transmissions.

If we were using just one tree, we would expect that the Mean Time Between Interruptions be reduced with an increase in multicast group size. This is so since increase in multicast group size increases the size of the tree and hence increases the probability that at least one of the links fails by any given time. However, while using backups, the total time for which the system lasts may increase with increase in multicast group size due to increase in probability of usefulness. This is because even though the first tree fails faster, the backup trees are available more often, hence increasing the total time (on an average) for which the the set of trees lasts. The effect of these two factors can be seen in Figure 3. From this figure, we see that in terms of the mean time between interruptions we can rank the schemes in the following order IPA, MIA, LCH and

Number of backup trees	SPTH	LCH	IPA
1	0.100s	0.142s	0.297s
2	0.058s	0.081s	0.105s
3	0.056s	0.079s	0.132s

TABLE I

INCREASE IN MEAN TIME BETWEEN INTERRUPTIONS DUE TO  $n^{th}$  BACKUP TREE

Number of backup trees	SPTH	LCH	IPA
1	0.37	0.54	0.77
2	0.07	0.12	0.12
3	0.03	0.05	0.03

TABLE II

INCREASE IN PROBABILITY OF USEFULNESS DUE TO  $n^{th}$  BACKUP TREE

SPTH with the IPA performing best. The two trees in LCH are expected to have greater independence than the SPTH, because by encouraging several links from one node to be included in the first tree, we make the tree occupy a smaller “area”, hence leaving greater space for the other tree to be formed without having to overlap with the first one. MIA ensures that the trees are almost edge disjoint, by computing the two trees simultaneously while the SPTH and LCH compute the first tree before the second one hence losing out on the possibility of combined optimization. IPA lasts much longer than other schemes, especially for larger multicast groups because of the fact that it includes local repair.

Probability of usefulness decreases with increase in dependence between the two trees. For the tree based algorithms, dependence between the two trees increases with increase in multicast group size because each tree occupies more “area”. On the other hand, as size of a tree increases, its failure time decreases. For this reason, given that first tree fails, it is very likely that the rest of the network is intact and hence the second tree is intact with higher probability. These opposing factors can be seen at play in Figure 4 especially for SPTH curve. On the other hand, the LCH curve does not change much, because the two factors balance each other out. In the case of MIA (IPA) the two trees (paths) are almost edge disjoint, irrespective of the size of the group and hence only the second factor dominates.

**Results for Two and Three Backup Trees** follow the same trends for various parameters as the one backup case with greater improvements in terms of probability and mean time between interruptions and probability of usefulness and higher cost and Dcost of trees.

### C. Improvements as a function of number of backups

This section presents performance of the algorithms as a function of number of backup trees computed. Table I contains the increase in mean time between interruptions (aver-

aged over multicast group sizes) as a function of number of backups for various schemes. We see that the IPA results in greater increase in mean time between interruptions than the other two schemes. Surprisingly, we also observe that, for IPA, the improvement in time due to third backup tree is greater than the improvement due to second backup tree. This is also true for LCH and SPTH for low multicast group sizes where dependence between two trees is still low. From Table II we see that increase in probability of usefulness decreases with increase in number of backup trees. This is an expected result because with increase in n, the probability that at least one first n-1 backup trees is available along with the nth backup tree at the time of failure of first tree increases.

## VI. CONCLUSIONS

Several heuristic schemes for constructing multiple “independent” trees were developed, simulated and their performance was compared in various contexts. We found that the Independent Path Algorithm gives much better performance than the others with very small increase in transmission cost of the multicast trees. We have shown through simulations that in a typical ad hoc network it is possible to have working backup infrastructure with high probability without much extra expense in terms of cost of the trees or computation or data collection. The probability of backup being useful is 0.9 for just 2 backup trees computed with no extra control overhead and mean time between interruptions is increased by 100%-600% (for most multicast group sizes) by use of 3 backup trees in a 50 Node network. The simulation results also indicate that, contrary to intuition, the improvement obtained due to additional trees does not always decrease with increase in number of backup trees.

## REFERENCES

- [1] C. Chiang, M. Gerla, and L. Zhang, “Shared tree wireless network multicast”, *IEEE International Conference on Computer Communications and Networks (ICCCN’97)*, September 1997.
- [2] S.-J. Lee, M. Gerla and C.-C. Chiang, “On-Demand Multicast Routing Protocol”, *Proc. IEEE WCNC’99*, New Orleans, LA, Sept 1999, pp. 1298-1304.
- [3] C. W. Wu and Y. C. Tay, “AMRIS: A Multicast Protocol for Ad hoc Wireless Networks,” *Proceedings of IEEE MILCOM ’99*, Atlantic City, NJ, Nov. 1999.
- [4] J.J. Garcia-Luna-Aceves, and E.L.Madruga, “The Core-assisted mesh protocol,” *IEEE Journal on Selected Areas in Communications*, Special Issue on Ad-Hoc Networks, Vol. 17, No. 8, Aug. 1998.
- [5] J. E. Wieselthier, G. D. Nguyen, and A. Ephremides, “Algorithms for Energy-Efficient Multicasting in Ad Hoc Wireless Networks”, *Proceedings of the IEEE Military Communications Conference (MILCOM)*, 1999, pp. 1414-1418.
- [6] M. Pearlman and Z. Haas, *On the impact of Alternate Path Routing for Load Balancing in Mobile Ad Hoc Networks*, MobiHOC’2000, Boston, MA, Sept. 1999, pp 3-10.
- [7] Katia Obraczka, Gene Tsudik, “Multicast Routing Issues in Ad Hoc Networks,” *IEEE International Conference on Universal Personal Communication (ICUPC’98)*, Oct. 1998.
- [8] M. Pearlman and Z. Haas, *Improving the Performance of Query-Based Routing Protocols Through ‘Diversity Injection’*, WCNC’99, New Orleans, LA, Sept. 1999.
- [9] W.J. Cook, W.H. Cunningham, W.R. Pulleyblank, and A. Schrijver, *Combinatorial Optimization*, John Wiley and Sons, 1998.