A Low-Cost Algorithm for Multicast Routing

Pingyan Tan
Frederick C. Harris, Jr.
Department of Computer Science
University of Nevada
Reno, Nevada 89557
fredh@cs.unr.edu

Abstract

This paper presents a low-cost algorithm for multicast routing in computer networks. For performance evaluation, it is compared with Link-State multicast routing, which is a low-delay algorithm used by the Internet MOSPF protocol [1]. This new algorithm supports the *group* concept and the *unknown-destination* delivery. It uses a *minimum average distance* method to select the forwarding links to achieve the low-cost goal. Another important feature of this algorithm is that no delivery tree has to be maintained, which eliminates the problem of memory and CPU congestion. Moreover, its computation time complexity is only linear in the group size. Simulation study shows that this algorithm can lower delay by as much as 40 percent for the same offered load, compared with the low-delay algorithm.

Keywords: network, routing, low-cost

1 Introduction

Routing is one of the most important aspects of computer networks, and unicast routing has been the dominant scheme used in network routing over the past several decades. However, in the past few years, multicast routing has seen widespread use in those networks that support it. There are many cases that exist where an application must send the same information to more than one destination, such as:

- updating all copies of a replicated file or database.
- sending data packets to all participants in a computer-mediated conference.
- emailing to more than one receiver.
- disseminating intermediate results to a set of processors supporting a distributed computation.

Multicast routing can be defined as follows: Given a network (N, L), where N is a set of nodes and L a set of links, a source node $s \in N$, and a set of destination nodes $G \subseteq N$ (the subset G is called a *group*), if $1 < |G| < |N| - 1$, then the problem is a multicast problem. (If $|G| = 1$, it is a unicast problem and if $|G| = |N| - 1$, it is a broadcast problem).

A multicast group is a set of destination nodes to which a message is to be delivered, and each destination node is a member of the group. For each source, there can be many different destination groups. The primary thing to remember is that multicasting should be more efficient than unicasting separate copies to each destination, because multicasting should reduce the transmission overhead.

Modern networks also require the *unknown-destination* delivery function for routing. If a set of destinations can be identified by a single group address, such a group address can be used to reach one or more destinations whose individual addresses are unknown to the sender or whose address may change over time. These capabilities are some of the implications of group addressing.

Algorithms for solving the multicast routing problem can be divided into two categories: low-delay algorithms and low-cost algorithms. Low-delay routing algorithms minimize the path length from the source to every destination node. This usually requires that the message be sent to every destination along the shortest path. Low-cost routing algorithms minimize the total traffic. The total traffic is equivalent to the total length of the paths in a delivery tree. Finding a minimum-cost routing tree is equivalent to finding a Steiner Tree in a graph, which is known to be NP-complete [2]. Many traditional heuristics for Steiner Tree problem are not suitable for multicasting when the network is large (see [2], Part II, Chapter 4 for coverage of several heuristics).

A number of algorithms have been proposed for multicast routing problems. The earlier algorithms [3, 4] do not support group addressing. The optimal multicast routing algorithms [5, 6] are too costly to be practical. The algorithms described in [1] and [7] are mainly low-delay algorithms. A low-cost algorithm, such as [8], that incrementally updates a delivery tree will lose effectiveness when group changes are frequent and when the number of source-group pairs is large, for it requires information be kept at each node for each such pair.

This paper presents a low-cost multicast algorithm that requires a packet to carry only one short field in addition to the group address. It is computationally efficient and effective in reducing the cost.

The remainder of the paper is organized as follows. In Section 2 we formally describe this new algorithm. In Section 3 we present a simple example for explaining the algorithm. A simulation model is presented in Section 4. Results are presented in Section 5, and Conclusions and Future Work follow in Section 6.

2 Description of the Proposed Algorithm

In an inter-network environment, it is assumed that sub-networks have the capability of multicast routing. In our model, each router and its attached sub-network are treated as a node. We omit the details within subnetworks and represent the network with a set of nodes connected by links. In this context, we use node and router interchangeably. It is also assumed that each node maintains a routing table for all nodes, and each routing table entry records the shortest path and the first-hop node for each destination. Each node also maintains the group-member information which ensures that every node knows the local membership of every group; therefore, when a packet arrives at a node, the node can decide whether there is any member in its attached subnetwork to which the packet will be forwarded.

For a general multicast routing algorithm, some basic requirements should be considered:

1) The algorithm should be computationally efficient: an algorithm is not useful if it requires more than linear time to determine to which link a packet is to be sent. A constant-time complexity is preferred.

2) The algorithm should be scalable: it should not be heavily dependent on network size, group size, etc.

3) The algorithm should be compatible with unicast routing: it should not require excessive information when compared with unicast routing and should not change dramatically from the unicast environment.

4) The algorithm should seek a minimization of delay and traffic: these two goals may or may not conflict with each other. A minimum-cost tree may require packets to reach some member host along a path that is not the shortest path. But when the minimum-cost tree is utilized, the entire traffic is reduced, which may help reduce the delay.

Another important issue is that of low-delay and low-cost routing. Low-delay routing is designed to minimize the path length from the source to every destination node. It usually requires the message to be sent to every destination along the shortest path or its equivalent as in unicast routing. Each link is assigned a length which can be 1, or the recent traffic rate, or queue length on the link. An interesting thing to remember when designing algorithms is that the link length from two opposite directions can be different.

The goal of low-cost routing is to minimize the total traffic, which is equivalent to the total length of the paths in a delivery tree. In graph-theoretic terms, a minimum-delay routing corresponds to a shortest-path tree, while a minimum-cost routing corresponds to a Steiner Tree in a graph. Finding a Steiner Tree is an NP-complete problem. There exist some heuristic algorithms for computing a low-cost multicast tree, as discussed in Section 1, but they either suffer from high computational cost or are not suitable for the datagram environment.

Our algorithm is aimed at low-cost routing and supports *group* addressing. The simulation results we present later strongly show that our new algorithm has better performance than a compared low-delay algorithm, in both cost and delay.

The main idea behind this algorithm is that we use the *minimum average distance* method to select the forwarding links. When a packet arrives at a node, the algorithm computes the average distance for each neighbor of the current node, where the *average distance* is the average distance from the neighbor node to all nodes in its *responsible destination set.* Then we select the neighbor node which has the minimum average distance to forward the packet. This procedure continues until all destination members are processed.

To describe the algorithm, we first define the following variables:

G	The destination set, that is, a group.
G_i	Node i's responsible destination set. It is the subset of G to which node i is responsible for delivering the message.
N_i	Node i's open neighborhood set, i.e., all of node i's neighbor nodes except for node i itself.
D_j	Neighbor j's responsible destination set.
L_j	Average distance from neighbor node j to j's responsible destination set.
A_i	The assigned neighbor set, i.e., the subset of G_i that has been assigned to a link.
d_{ij}	The shortest path distance from node i to node j.

In addition to the routing table of its own, each node also keeps the routing table of its neighbors, which is the common practice in the distributed Bellman-Ford algorithms [9]. As usual, the routing table at each node is assumed to contain the following entries that are related to our work:

- destination
- shortest distance
- next hop

where "next hop" tells which link to take to reach the destination on the shortest path from the current node. We do not use the tabled next hop directly, but compute the next hop based on our algorithm. In the algorithm, each packet p is required to carry a small integer $p.radius$ that is used in conjunction with the next-hop entries to compute the routing path.

Suppose packet p arrives at node i via a neighbor node k or is generated at node i. The description of the algorithm is as follows:

1. If i is the source node,

   $R = \infty; \quad A_i = \emptyset;$

   else

$R = p.radius; \quad A_i = \{k\};$

If $R = 0$, stop.

2. Compute node i's responsible set

$$G_i = \{g \mid g \in G,\ d_{ig} \le R \text{ and } g \ne i\}$$

3. For each $j \in N_i$ and $j \notin A_i$, compute j's destination set

$$D_j = \{g | g \in G_i \text{ and } d_{jg} < d_{ig}\} \tag{1}$$

and the average length

$$L_j = (d_{ij} + \sum_{g \in D_j} d_{jg} - \hat{L}_j)/|D_j| \tag{2}$$

where $\hat{L}_j$ is the sum of the overlapping next-hop links, defined as

$$\hat{L}_j = \sum_{n_{jk} > 1} (n_{jk} - 1) l_{jk}$$

where n_{jk} is the number of members in D_j that will be sent via link jk from node j to node k, and l_{jk} is the length of the link. If (all $D_j = \emptyset$) or ($N_i = A_i$), stop.

4. Let J be the subscript such that

$$L_J = \min_{j \in N_i, j \notin A_i} L_j$$

and define

$$R_J = \max_{g \in D_J} d_{Jg} \tag{3}$$

Send a packet with radius R_J to node J.

5. $G_i = G_i - D_J \qquad A_i = A_i \cup \{J\}$

If $G_i = \emptyset$

stop

else

go to step 3. □

When a packet arrives at a node i and is addressed to group G, we give a condition to limit the size of the destination set – that is, to decide to which members of group G should be sent the packet from the current node i. We use a variable $p.radius$ as the condition. The value of $p.radius$ is the path distance between the current node and the most remote destination member. That is, the current node i sends packets only to those members (in G_i) whose shortest-path distance from node i is less than or equal to $p.radius$ (Step 2). If node i is the source node, then we set $p.radius = \infty$. This means that at the beginning, we consider all members of the group G as the destination set. If node i is not the source node, the packet must have come from its parent node; the packet then carries a radius with it which was computed at the parent node. A_i at the source node is $\emptyset$. When a packet arrives at node i from its parent, A_i initially contains the parent node. A_i will be increased by adding node i's neighbors one by one until it contains all of the neighbors or other terminating conditions are satisfied (Step 3).

For each neighbor $j \in N_i$ and $j \notin A_i$, we compute j's responsible destination set D_j by (1) and each j's average length L_j by (2). We then send the packet together with a radius to the selected neighbor node which has the minimum average path length. The radius is computed by (3). It is the path length between the selected neighbor node and the most remote member from G_J. G_i is updated by taking out the destination set of the selected neighbor node. A_i is updated by adding in the selected neighbor node.

The algorithm will terminate at a node in three cases:

1) if all $D_j = \emptyset$
2) if $N_i = A_i$
3) if $G_i = \emptyset$

If none of the above conditions are satisfied, then it will go back to Step 3 to process other neighbor nodes until one of the three conditions is satisfied.

Each node executes the same algorithm whenever a packet arrives. The algorithm execution will stop when the radius for every packet is zero.

3 A Simple Example

We use the network shown in Figure 1 to illustrate our algorithm. Assume that the source node is S, and the destination set $G = \{E, F, M\}$. Each node in the graph has a letter which is the name of the node and numbers which represent an ordering of their neighbors. The result shows that the cost generated by our algorithm is 11, but with shortest path algorithm, the total cost is 13.

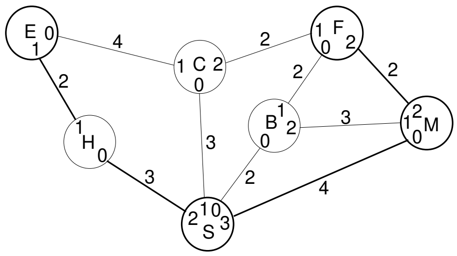

Figure 1: Example Network

At node S

1. The packet is generated, $R = \infty$, and $A_S = \emptyset$.

2. $G_S = \{E, F, M\}$

3. $N_S = \{H, C, B, M\}$, $A_S = \emptyset$
   $D_H = \{E\}$ $\quad L_H = 3 + 2 = 5$
   $D_C = \{E, F\}$ $\quad L_C = (3 + 4 + 2)/2 = 4.5$
   $D_B = \{F, M\}$ $\quad L_B = (2 + 2 + 3)/2 = 3.5$
   $D_M = \{M, F\}$ $\quad L_M = (4 + 0 + 2)/2 = 3$

4. $J = M$, $R_J = 2$
   —**send packet** p **to** M **with** $p.radius = 2$

5. $G_S = \{E\}$, $A_S = \{M\}$ — go to step 3

   3. $N_S - A_S = \{H, C, B\}$
      $D_H = \{E\}$ $\quad L_H = 5$
      $D_C = \{E\}$ $\quad L_C = 7$
      $D_B = \emptyset$ $\quad L_B = \infty$

   4. $J = H$, $R_J = 2$
      —**send packet** p **to** H **with** $p.radius = 2$

   5. $G_S = \emptyset$ — stop

At node M

1. $G = \{E, F, M\}$, $R = 2$, $A_M = \{S\}$

2. $G_M = \{M, F\}$

3. $N_M = \{B, S, F\}$, $A_M = \{S\}$, $N_M - A_M = \{B, F\}$
   $D_B = \emptyset$ $\quad L_B = \infty$
   $D_F = \{F\}$ $\quad L_F = 2/1 = 2$

4. $J = F$, $R_J = 0$
   —**send packet** p **to** F **with** $p.radius = 0$

5. $G_M = \{M\}$, $A_M = \{S, F\}$ — go to step 3

   3. $N_M - A_M = \{B\}$
      $D_B = \emptyset$ — stop

At node H

1. $G = \{E, F, M\}$, $R = 2$, $A_H = \{S\}$

2. $G_H = \{E\}$

3. $N_H = \{E, S\}$, $N_H - A_H = \{E\}$
   $D_E = \{E\}$ $\qquad L_E = 2$

4. $J = E$, $R_J = 0$
   —**send packet** p **to E with** $p.radius = 0$

5. $G_H = \emptyset$ — stop

In the end, all packets at all involved nodes have $R = 0$, so the algorithm execution is terminated.

4 Simulation Studies

Simulation studies were performed on our algorithm and the Link-State multicast algorithm described in [7], which is a low-delay algorithm. The Link-State multicast algorithm was chosen because it assumes roughly the same network information as does our algorithm and because it is the multicast version of the Internet standard protocol OSPF and is used in the experimental MOSPF protocol [1]. Also, by comparing with a low-delay algorithm, we were able to study the impact on delay by traffic reductions.

The simulation studies measured two different relationships: one was delay versus offered load, the other was utilization versus offered load. Delay is defined to be the interval between the time a packet is generated and the time the packet arrives at the destination. In a multicast environment, the delay is the average delay for all destinations. Offered load is defined to be the rate of packet generation. Utilization is defined to be the average utilization of all links.

A queueing model was used for the simulation study, where the link from node i to node j was modeled as a queue. Unicast packets and multicast packets with random destinations were generated at each node with exponentially distributed intervals. Groups of random sizes appeared and disappeared also with exponentially distributed intervals. The group population was maintained at a stable level by equal birth and death rates.

The parameters of interest in our algorithm are defined in Table 1. Note that *ms* in the table should be interpreted as a relative time unit. The initial group population is also the average number of active groups, with the group birth and death rates being equal.

Symbol	Definition
t_b	transmission rate, bits/ms
t_p	packet transmission time, ms
λ_u	unicast packet arrival rate, packet/ms/node
λ_m	multicast packet arrival rate, packet/ms/node
λ_g	birth and death rate of groups, group/ms
G	initial group population
$G_{\max}$	maximum size of a group
$G_{\min}$	minimum size of a group

Table 1: Network Parameters

```
1.  On receipt of a link-state update reporting a change in g
        for each cache record c with c.grp = g,
            discard c
2.  On receipt of a multicast packet from
    source s to group g via node k,
        if no cache record c exists such that
        c.src = s and c.grp = g
            create c; c.src = s; c.grp = g
            compute c.parent and c.child
        if c.parent = k,
            for each child node j,
                send copy of packet to node j
```

Figure 2: *Link-State multicast* Algorithm

The Link-State multicast algorithm [7] maintains a delivery tree for each active source-group pair. A simplified version of the Link-State multicast algorithm is given in Figure 2. In the algorithm, a cache record c is created for each active source-group pair to be maintained in the delivery tree. A cache record at a node has the fields $c.src$, $c.grp$, $c.parent$, and $c.child$ to keep the source, group, parent and child nodes in the delivery tree for that source-group pair. Whenever a group g appears or disappears on a node k, node k will flood a link-state update to report this change.

5 Results

We randomly generated four networks with 40, 75, 97, and 123 nodes respectively. In each network, the number of groups fluctuated around an average, determined by a birth/death process, and the size of each group was determined by a random number uniformly distributed in a certain range. In the simulation, the multicast packet generation rate, λ_m, was set to be the same as the unicast rate, λ_u. The rates were assigned the values for which no severe congestion was observed and reasonable utilizations were obtained. Table 2 gives the ranges for the values of other parameters for the four networks (refer to Table 1 for parameter definitions). In the table, the value $1/1K$ for λ_g means that the group birth and death rate is 1/1024 of the unicast packet generation rate.

network size	G	$G_{\min}$	$G_{\max}$	av. # links	hit ratio	λ_g	t_p	link length
40	3	2	39	2.95	91%	$1/1K$	0.1	1
75	5	2	37	2.59	91%	$1/1K$	0.1	1
97	4	2	48	3.22	92%	$1/1K$	0.1	1
123	3	2	40	3.25	86-95%	$1/1K$	0.1	1

Table 2: Parameter values

The hit ratio is the ratio for not reconstructing the delivery tree for the Link-State multicast algorithm. This is not an independent variable, rather it is the function of other parameters. We chose the parameters so that the ratio was slightly over 90%. This selection of parameters is reasonable and makes the Link-State algorithm perform well. The hit ratio can be decreased by increasing group appearance/disappearance or group membership change frequency, or by increasing group size or the number of groups. However, the Link-State algorithm does not perform as well in these situations. We execute both our algorithm and the Link-State multicast algorithm with exactly the same network parameters.

We assume that the time for the reconstruction of the delivery tree for the Link-State multicast algorithm is $(E + N)\log N$ units [10], where N is the number of nodes, and E is the total number of links in the network. This is equivalent to transmitting $((E+N)\log N)/L$ packets from one node, where L is the packet length (in bits). We also assume that the time for selecting the next hop for our algorithm is $G_s \times N_l$ units for each node, where G_s is the destination group size and N_l is the number of links at that node. This time is equivalent to transmitting $G_s \times N_l/L$ packets.

The first 3,000 packets (on average) received at each node were discarded to ensure that the network reaches a steady state; then statistics are collected for the subsequent 10,000 packets (on average) received at each node.

The simulation results are shown in Figures 3 through 8. In these figures, the curves labeled 'Radius' are for our algorithm, those with 'LS' are for the compared algorithm. In Figures 3 through 7, the horizontal axes represent the message rate, which is the offered load. The vertical axes represent the delay (Figures 3, 4, 5) and utilization (Figures 6, 7). Figure 8 shows the average delay and utilization reductions by algorithm over the Link-State multicast algorithm. The reductions are calculated by the following formula:

$$A = \frac{\sum_{i=1}^{4} \frac{T_i - R_i}{T_i}}{4} \times 100\%$$

where

A	is the average percentage delay or utilization reduction.
LS_i	is the delay or utilization of the Link-State multicast algorithm for the ith network.
R_i	is the delay or utilization of our algorithm for the ith network.

The delays and utilizations for the smaller network of 40 nodes are based on a different range of message rates (100 to 800) from that used in the three larger networks (50 to 400), because the smaller network can tolerate higher message rates for both algorithms. To calculate the average, the message rates are normalized with the highest rate used in each network.

The simulation results show that our algorithm has better performance than the compared algorithm since both the delay and the utilization are lower on the four networks. For the same offered load, both the delays and network traffic are lower. At certain crucial message rates, the delay can be lowered by 40 percent. The delay reduction at these rates means our algorithm can support higher offered load than can the compared algorithm.

When the message rate increases, the corresponding utilization of links increases, as seen in Figures 6 and 7. Utilization reflects the traffic load. The increased traffic causes packets to be queued for the available links. In the case when the queuing delay plays the major role, our algorithm shows more advantages, since it is a low-cost algorithm. Reduced total cost will decrease the waiting time for queued packets. Therefore, the total delay is eventually decreased.

6 Conclusions and Future Work

It is known that a low-cost algorithm can reduce traffic and delay. This study shows that with our simple algorithm that requires only a linear time

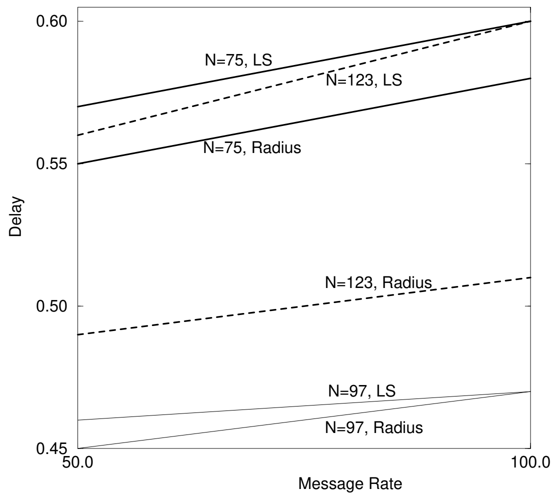

Figure 3: Delays (lower message rates)

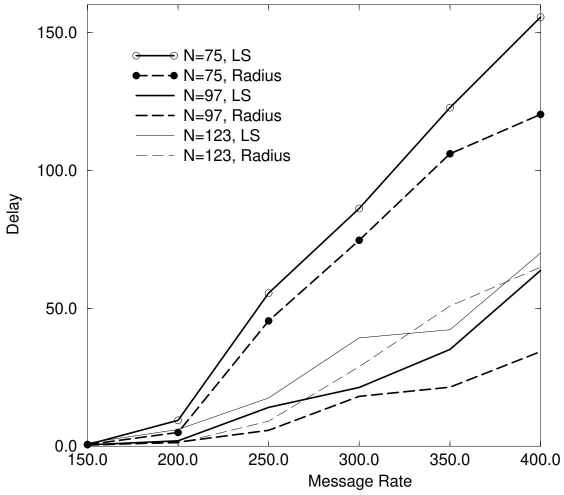

Figure 4: Delays (higher message rates)

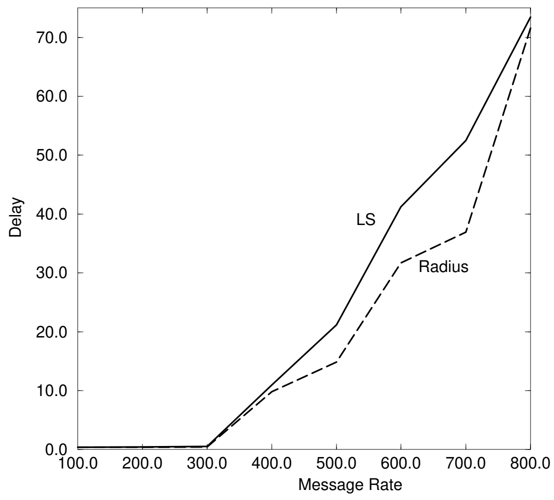

Figure 5: Delays for $N = 40$

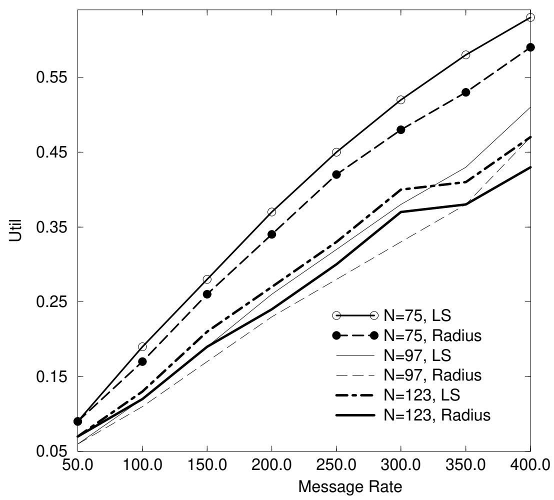

Figure 6: Utilizations

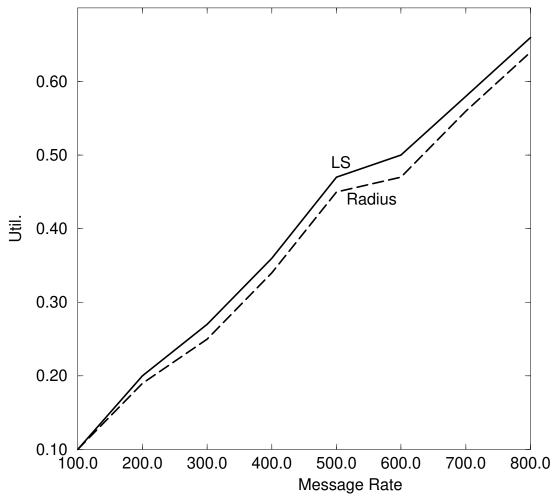

Figure 7: Utilizations for $N = 40$

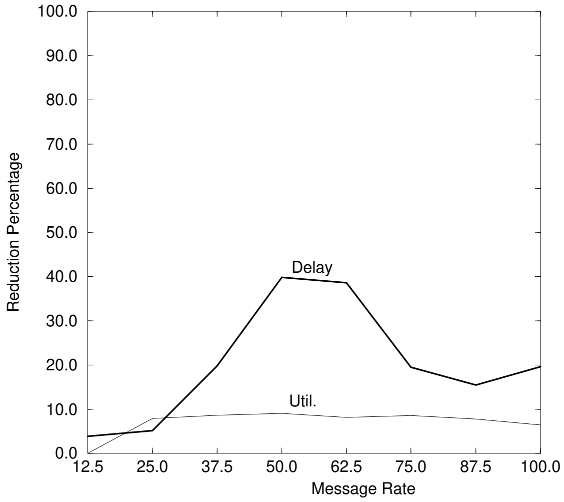

Figure 8: Reductions

complexity, the delay can be substantially lower than that offered by always taking the shortest path, and yet this algorithm has all the elements that are required for multicasting, such as efficient computation, group addressing, and compatibility with unicasting. The radius concept and the minimum average distance method are effective in resolving group addressing and provide a practical low-cost algorithm for multicasting in inter-networks.

For more efficiency, we can allow the packet to carry a limited number of member addresses. Suppose we let the packet carry the addresses of the four most remote members, the radius R will be smaller and the algorithm will converge faster. As shown by Aguilar [3] the IP header usually has 40 bytes for optional use; therefore, such consideration is highly feasible.

References

[1] C. Huitema, *Routing in the Internet*, Prentice Hall PTR, Englewood Cliffs, New Jersey 07632, 1995.

[2] F. Hwang, D. Richards, and P. Winter, *The Steiner Tree Problem*, volume **53** of *Ann. Discrete Math.*, North-Holland, Amsterdam, 1992.

[3] L. Aguilar, "Datagram routing for internet multicasting," in *Proc. ACM SIGCOMM '84 Communications Architectures and Protocols*, pp. 58–63. ACM, June 1984.

[4] K. Bharath-Kumar and J. M. Jaffe, "Routing to multiple destinations in computer networks," *IEEE Trans. Comm.*, vol. **COM-31**, no. 3, pp. 343–351, March 1983.

[5] G. I. Stassinopoulos, "Optimal dynamic routing in multidestination networks," *IEEE Trans. Comm.*, vol. **COM-35**, no. 4, pp. 472–475, April 1987.

[6] G. I. Stassinopoulos and M. G. Kazantzakis, "A computationally efficient iterative solution of the multidestination optimal dynamic routing problem," *IEEE Trans. Comm.*, vol. **39**, no. 9, pp. 1370–1378, 1991.

[7] S. E. Deering, *Multicast routing in a datagram internetwork*, PhD thesis, Dept. of Electrical Engineering, Stanford University, December 1991.

[8] N. E. Belkeir and M. Ahamad, "Low cost algorithms for message delivery in dynamic multicast groups," in *Proc. 9th Intl. Conf. Distributed Computing Systems*, pp. 110–117. IEEE, June 1989.

[9] D. Bertsekas and R. Gallager, *Data networks*, Prentice-Hall, Inc., 2nd edition, 1992.

[10] R. Sedgewick, *Algorithms*, Addison-Wesley Publishing Company, Inc, New York, 2nd edition, 1989.