Published in Proceedings of INFOCOM ‘91, pp. 1274-1283, Miami, Florida, April 7-11, 1991.

On Multicast Path Finding Algorithms

Ching-Hua Chow

Bell Communications Research

445 South Street, MRE 2Q231
Morristown, NJ 07960-1910
Phone: (201) 829-4534
Email: chow@bellcore.com

Abstract

We have designed and implemented three multicast path finding algorithms for networks with directed links: an optimal algorithm based on the dynamic programming technique, a heuristic algorithm with the assumption that all vertices have the multicast capability, and a heuristic algorithm for networks where some vertices do not have the multicast capability. Computation results show that the heuristic algorithms can find multicast paths whose costs are close to optimal and can operate with reasonable response time for large networks. We discuss applications of these path finding algorithms to set up multipoint connections.

1. Introduction

With the recent advance in terminal, high-speed transmission and switching technology, it is now feasible to provide multimedia, multiparty communication services. In order to support these services, networks need to have the capability to setup/modify the following five basic types of connections: point-to-point, point-to-multipoint (also called **Multicast**), multipoint-to-point (also called **Concast**), multipoint-to-multipoint, and point-to-allpoint (also called **Broadcast**). Since the video connections and the high speed switching and transmission components implement uni-directional paths [Bus89], in this paper we are considering directed networks with uni-directional links.

In directed networks, the shortest path algorithm proposed by Dijkstra can be used to set up point-to-point connections. It is a polynomial algorithm with $O(v^2)$ time complexity where v is the number of vertices in the network [Dij59]. The minimum spanning tree algorithm can be used to set up broadcast connections and is also a polynomial algorithm with $O(e \log e)$ time complexity where e is the number of edges in the network [Kru56].

In a given directed graph $G = (V, E)$, with a vertex set V, and an edge set E, a vertex which has the multicast capability is called a **multicast vertex**. A **multicast path** $s \rightarrow D$ is a tree rooted at vertex $s \in V$, with all branching vertices to be multicast vertices, and with its leaf vertices constitute D, $D \subset V$. s is called the **source** of the multicast path and D is called the **destinations** of the multicast path. The **multicast path finding problem** can be defined as follows:

> Given a directed graph $G = (V, E)$, with a cost function c:$E \rightarrow \Re^+$, find a minimum cost multicast path $s \rightarrow D$.

The algorithms for finding a multicast path are called **multicast path finding algorithms**. They are similar to those for the **Steiner tree problem** which is defined as follows:

> Given an undirected graph $G = (V, E)$ together with a cost function c:$E \rightarrow \Re^+$, and $V' \subset V$, the Steiner tree problem for V' in G is to find a minimum cost, connected subgraph of G which contains all the vertices V' (and possibly some other vertices, which are called **Steiner vertices**).

The Steiner tree problem was shown to be NP-hard [Karp 72] and a good survey of algorithms is in [Win 87]. Among the optimal algorithms, the dynamic programming algorithm of Dreyfus and Wagner [DRE 72] seems to outperform others. For heuristic algorithms, the algorithm proposed by Takahashi and Matsuyama [TAK 80] and that by Rayward-Smith [RAY 86] perform quite well. The former is based on the shortest path to a subset of vertices. The later is based on a heuristic function to select Steiner vertices. Waxman further considered optimizing a sequence of multicast connection requests [WAX 88]. Tu and Leung gave a sketchy description of a source-based spanning tree algorithm used in setting up multicast connections in a wide-area multicast packet switching network [Tu90].

In this paper we only discuss centralized multicast path finding algorithms. Research results have been reported on distributed versions. Deering and Cheriton's paper provides a good survey on multicast routing in Datagram Internetworks and Extended LANs [Dee90]. However, the multimedia multipoint connections which we are considering are much more dynamic and complex. It is not restricted to the group address scheme, and involves reservation of resources such as conference bridges and protocol converters.

The rest of the paper is organized as follows: Section 2 describes an optimal multicast path finding algorithm. Section 3 describes a heuristic multicast path finding algorithm which performs well for networks where almost all vertices are multicast vertices. Section 4 presents a heuristic algorithm which perform well for networks with small number of multicast vertices. Section 5 presents computation results of our implementations of these three algorithms, showing that the two heuristic algorithms provide comparable performance at much lower complexity. Section 6 illustrates how other multipoint connections can be solved by the combination of the shortest path algorithm and the multicast path finding algorithm. Section 7 discusses related research issues.

2. An Optimal Multicast Path Finding Algorithm

Here we present an optimal multicast path finding algorithm based on the dynamic programming technique proposed by Dreyfus and Wagner [DRE 72]. Intuitively, for an optimal multicast path, $s \longrightarrow D$, the technique starts by first finding all the optimal multicast paths to small subsets of D. It then finds all the optimal multicast paths to larger subsets of D by merging previous obtained optimal multicast paths. To facilitate the presentation, first let us define some of the terms used in the algorithm.

Given a directed graph $G = (V, E)$, $\boldsymbol{OT(v, X)}$ is an optimal multicast path $v \longrightarrow X$. The cost of a multicast path P, $\cos t(P)$, is the sum of all edge costs in P. $\boldsymbol{BT(v, D)}$ is a multicast path which is the minimum cost union of two multicast paths, say $v \longrightarrow X$ and $v \longrightarrow (D \text{ Ð } X)$, which span multicast vertex v and with their destinations being the non-empty disjoint subsets of D. The i-vertex set of D is the set of all vertex sets, X such that $|X| = i$ and $X \subset D$. For example, the 2-vertex set of $\{a, b, c\}$ is $\{\{a, b\}, \{a, c\}, \{b, c\}\}$ and has $\binom{3}{2}$ elements.

The optimal multicast path finding algorithm for $s \longrightarrow D$ is as follows:

```
for $i, j \in V, i \neq j$ do $OT(i, \{j\})$ = shortest path from $i$ to $j$ done
for $i \in \{2, ..., |D|\}$ do {
    for $D_i \in i$-vertex set of $D$ and $v \in V$ do
        find $BT(v, D_i)$ where
            $\cos t(BT(v, D_i)) = \min_{\varnothing \subset X \subset D_i}\{\cos t(OT(v, X)) + \cos t(OT(v, D_i \text{ Ð } X))\}$
    done
    if ($i = |D|$) break; // we do not need $OT(v, D_{|D|})$ where $v \neq s$.
    for $D_i \in i$-vertex set of D and $v \in V$ do
        find $OT(v, D_i)$ where
            $\text{cost}(OT(v, D_i)) = \min_{k \in V}\{\text{cost}(OT(v, \{k\})) + \text{cost}(BT(k, D_i))\}$
    done
}
find $OT(s, D)$ where
        $\text{cost}(OT(s, D)) = \min_{k \in V}\{\text{cost}(OT(s, \{k\})) + \text{cost}(BT(k, D))\}$
```

In terms of the time complexity, the inner loop that calculates the $\cos t(BT(v, D_i))$ will try all possible non-empty vertex sets of D_i which has $\frac{1}{2}\sum_{k=1}^{i \text{ Ð } 1}\binom{i}{k} = 2^{(i \text{ Ð } 1)} \text{ Ð } 1$ possibilities, and this inner loop will be executed $|V| \cdot \sum_{i=2}^{|D|}\binom{|D|}{i}$ times. The term

$\mathrm{cos}t(OT(v, X)) + \mathrm{cos}t(OT(v, D_i \text{ Đ } X))$ will be executed $V| \cdot \sum_{i=2}^{|D|} \left(\binom{|D|}{i} \cdot 2^{(i \text{ Đ } 1)} \text{ Đ } 1 \right)$ times. Similarly, the term $OT(s, \{k\}) + \mathrm{cos}t(BT(k, D))$ will be executed $|V|^2 \cdot \sum_{i=2}^{|D|} \binom{|D|}{i}$ times. Totally, the add operation will be carried out $V| \cdot \sum_{i=2}^{|D|} \binom{|D|}{i} \cdot ((2^{(i \text{ Đ } 1)} \text{ Đ } 1) + |V|)$ times. For network with 20 vertices, to find out a multicast path with 5 destinations will perform the addition 12,700 times. The number grows exponentially with the size of the destinations.

In the implementation, the key problem is to generate the vertex sets and to map them into unique numbers so that we can store all the BT and OT multicast paths in an array and retrieve them later on. A separate paper will describe our solution that allowed efficient implementation.

3. RST Heuristic Multicast Path Finding Algorithm

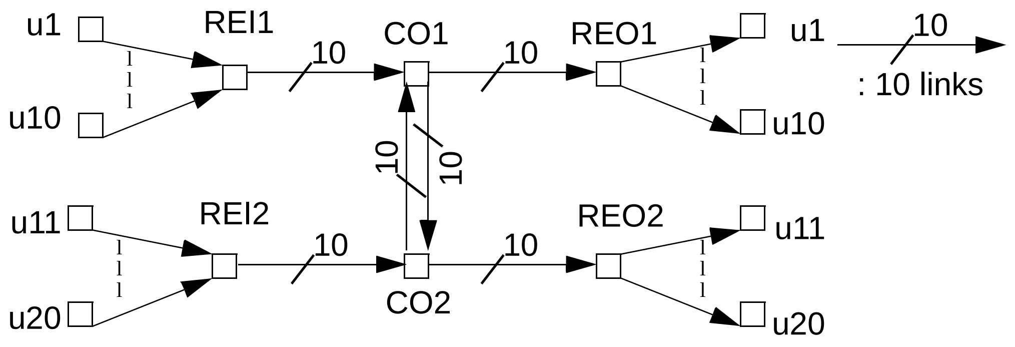
Here we present a heuristic multicast path finding algorithm for networks where all vertices have multicast capabilities. Since this algorithm applies the spanning tree algorithm in the reverse edge direction, we called it the Reverse Spanning Tree (RST) multicast path finding algorithm. In the next section, we consider another heuristic multicast path finding algorithm for networks with both multicast and non-multicast vertices.

Given a connection request to set up a multicast path $s \rightarrow D$, the RST multicast path finding algorithm performs as follows:

```
Step 1.  Use spanning tree algorithm to find a path $P$ from $s$ to any vertex in $D$, say $d$.
         Let $N$ be the set of vertices in $P$.
Step 2.  **for** each vertex in $D \text{ Đ } \{d\}$, say vertex $x$, **do {**
             Use the spanning tree algorithm which traverse edges in reverse direction
             to find the shortest path $SP$ from $x$ to any multicast vertex in $N$.
             Add $SP$ to $P$.
             Update $N$.
         **}**
```

The algorithm has the time complexity of $O(|D| \cdot e \cdot \log e)$ where e is the number of edges in the networks. It is not clear which order of the vertex selection in this algorithm will find a path with the lowest cost. For a multicast path with large destinations, doing the exhausted search on all the possible vertex selection sequences may not be feasible.

Consider a network such as that in Figure 1a, which has 26 vertices and 80 links. Assuming that the link costs are the same and that all REs and COs are multicast vertices, this algorithm will perform as is depicted in Figure 1b to find multicast path $u1 \rightarrow \{u2, u4, u11, u15\}$. In Step 2, the



a). A network with 20 users

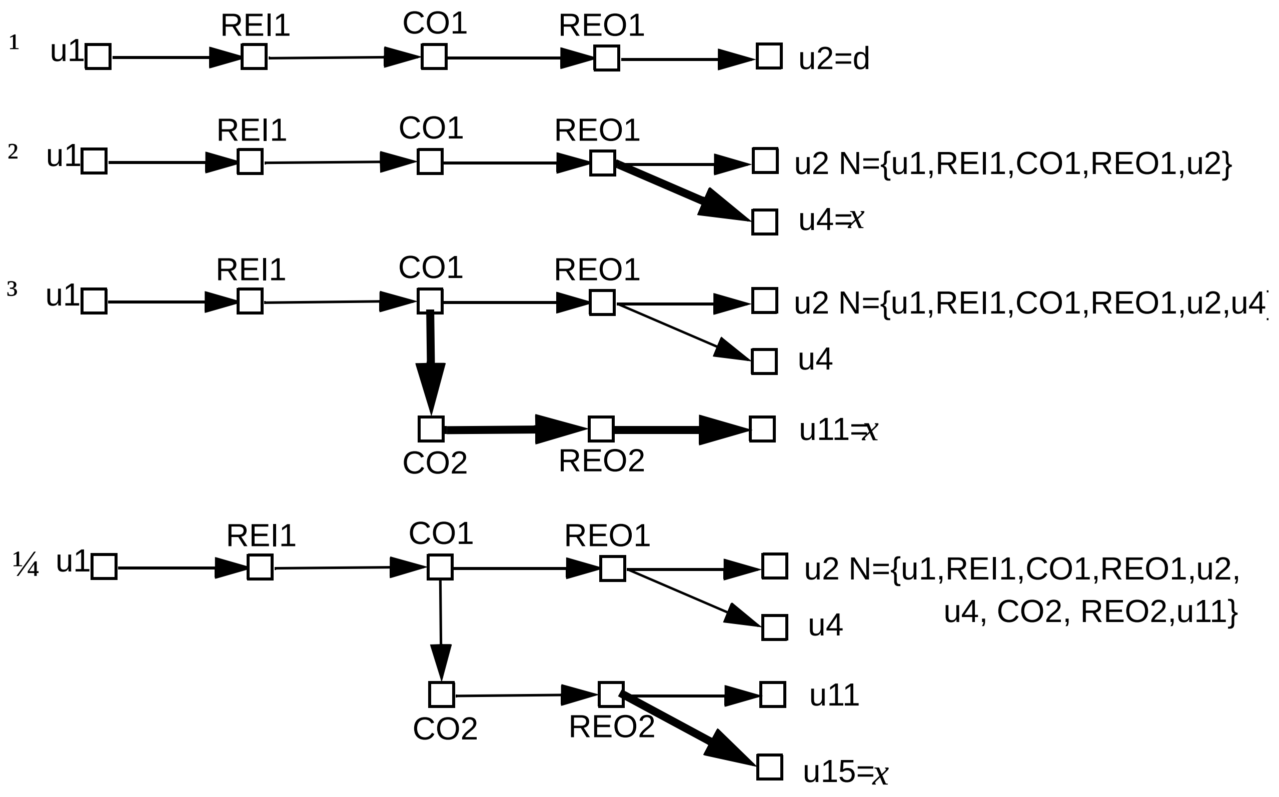

1
u1
REI1
CO1
REO1
u2=d
2
u1
REI1
CO1
REO1
u2 N={u1,REI1,CO1,REO1,u2}
u4=x
3
u1
REI1
CO1
REO1
u2 N={u1,REI1,CO1,REO1,u2,u4
u4
CO2
REO2
u11=x
¼
u1
REI1
CO1
REO1
u2 N={u1,REI1,CO1,REO1,u2, u4, CO2, REO2,u11}
u4
CO2
REO2
u11
u15=x
b). The 4 steps that derive the multicast path

Figure 1. Use RST path finding algorithm to find $u1 \rightarrow \{u2, u4, u11, u15\}$

algorithm tries to find the shortest path from any of the multicast vertices in the vertex set

N={u1,REI1,CO1,REO1,u2} to vertex u4. The bold edge from vertex REO1 to vertex u4 is selected. In this case the algorithm finds the optimal path.

4. RMV Heuristic Multicast Path Finding Algorithm

For networks with both multicast and non-multicast vertices, the RST algorithm may not be able to find a multicast path even though it exists. For example, if the path found in Step 1 of the RST algorithm does not contain a multicast vertex, then the algorithm simply fails. In this section, we present a heuristic algorithm that deals with this new constraint. It is based on the observation that a multicast path will consist of two parts: one is a shortest path from the source vertex to a multicast vertex, say m, which can reach all destination vertices, and the other is a multicast path from vertex m to all destination vertices. The goal here is to find such an intermediate multicast vertex with which the cost of the corresponding multicast path is minimum. Since this algorithm tries to find the reachable multicast vertices first, we called it, the RMV multicast path finding algorithm.

Given a connection request to set up a multicast path $s \to D$, the RMV multicast path finding algorithm performs as follows:

```
Step 1.  Find the reachable multicast vertex set, $RMVS$, from $s$.
Step 2.  for each vertex in $RMVS$, say vertex $m$, do
             find all the shortest paths from $m$ to $D$.
Step 3.  Select a multicast vertex in $RMVS$, say vertex $x$, where
         $\cos t(shortestpath(s \to m)) + \cos t(shortestpaths(m \to D))$
             is minimum.
Step4.   Use RST algorithm to find the multicast path $m \to D$, say $MP$.
         Join $MP$ with the shortest path $s \to m$.
```

This algorithm has the time complexity of $O(|D| \cdot |RMVS| \cdot |V|^2)$. The larger the size of D or $RMVS$, the longer it takes to compute the path.

5. Computation Results

In this section we present the computation results of our implementations of the three multicast path finding algorithms. It is shown that the computation time differences between the optimal and the heuristic algorithms are two to four orders of magnitude, while the costs of multicast paths generated are very close. The program which implements these three algorithms was written in GNU C++ and the results were obtained by running the program on a 24 MIPS machine with 32 MBytes main memory.

Figure 2 shows a ring network with a center hub, n ring access vertices, and m users connected

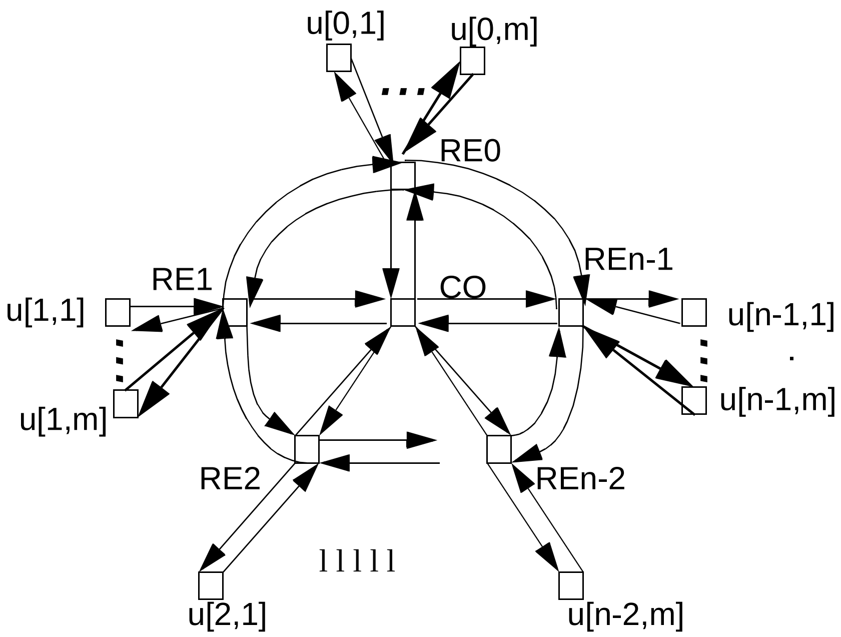


Figure 2. A network Topology: RingWithHub-n-m

	Optimal Algorithm		**RST Algorithm**		**RMV Algorithm**	
Destinations	**Time(μsec)**	**Cost**	**Time(μsec)**	**Cost**	**Time(μsec)**	**Cost**
2	**3621193**	**4.8**	**5452**	**5**	**183889**	**4.8**
3	**4836429**	**6.4**	**7313**	**7**	**244284**	**6.4**
4	**4095774**	**8.0**	**8806**	**9**	**299050**	**8.0**
5	**4504613**	**9.6**	**11651**	**11**	**415480**	**9.6**
6	**7167733**	**11.2**	**12869**	**13**	**478796**	**11.2**
7	**9580842**	**12.8**	**13464**	**15**	**537617**	**12.8**
8	**34719706**	**14.4**	**15813**	**17**	**586038**	**14.4**

Table 1. Performance on RingWithHub-16-1 (33 nodes, 96 links) with the link cost between CO and REs to be 0.6 and other link cost to be 1.0

to each ring access vertex. We call this type of network topology RingWithHub-n-m. Table 1 lists the computation time and the cost of multicast paths generated by the three algorithms on a network of type RingWithHub-16-1 with the following cost function: the link cost between the center hub and a ring access vertex is 0.6 and the cost of other links is 1.0. Because the system load difference, we see time fluctuate at the run for three destination users case using the optimal multicast path finding algorithm. The source is u[0,1] and the destination users are {u[1,1], ..., u[x,1]} where x is the number of destination users. The multicast paths generated by the RST heuristic algorithm have costs about 4-20% higher than those of the corresponding optimal paths. Those generated by the RMV algorithm are optimal. The computation time of the RST algorithm is about four orders of magnitude less than that of the optimal algorithm. The computation time of RMV algorithm is about an order of magnitude less than that of the optimal algorithm. Since the optimal algorithm grow exponentially with respect to the size of destinations, the time difference between the optimal algorithm and the two polynomial heuristic algorithms will be even greater as the size of destinations become larger.

The two multicast paths generated by the program for a network of type RingWithHub-3-1 are as follows:

```
The Heuristic multicast path U[0,1]->{U[1,1] U[2,1]} source U[0,1] Destination
U[1,1] U[2,1]
U[0,1].outputport.0 --Link-- Link.U[0,1]-RE[0] --> RE[0].inputport.0
RE[0].outputport.2 --Link-- Link.RE[0]-RE[1] --> RE[1].inputport.2
RE[1].outputport.0 --Link-- Link.RE[1]-U[1,1] --> U[1,1].inputport.0
RE[0].outputport.3 --Link-- Link.RE[0]-RE[2] --> RE[2].inputport.3
RE[2].outputport.0 --Link-- Link.RE[2]-U[2,1] --> U[2,1].inputport.0
endPath
    cost=5
    time=4427microsec


The optimal multicast path U[0,1]->{U[1,1] U[2,1]}
    its optimal cost is 4.8
    it takes 116096 microsecs
    its path is as follows:
Path sp(U[0,1]->CO) Source U[0,1] Destination CO Segment_List:
    U[0].outputport.0 --Link-- Link.U[0,1]-RE[0] --> RE[0].inputport.0
    RE[0].outputport.1 --Link-- Link.RE[0]-CO --> CO.inputport.0
endPath
Path sp(CO->U[1,1]) Source CO Destination U[1,1] Segment_List:
    CO.outputport.1 --Link-- Link.CO-RE[1] --> RE[1].inputport.1
    RE[1].outputport.0 --Link-- Link.RE[1]-U[1,1] --> U[1,1].inputport.0
endPath
Path sp(CO->U[2,1]) Source CO Destination U[2,1] Segment_List:
    CO.outputport.2 --Link-- LInk.CO-RE[2] --> RE[2].inputport.1
    RE[2].outputport.0 --Link-- LInk.RE[2]-U[2,1] --> U[2,1].inputport.0
endPath
```

Figure 3 shows a star-like network with COs forming a ring and with REs connecting to groups

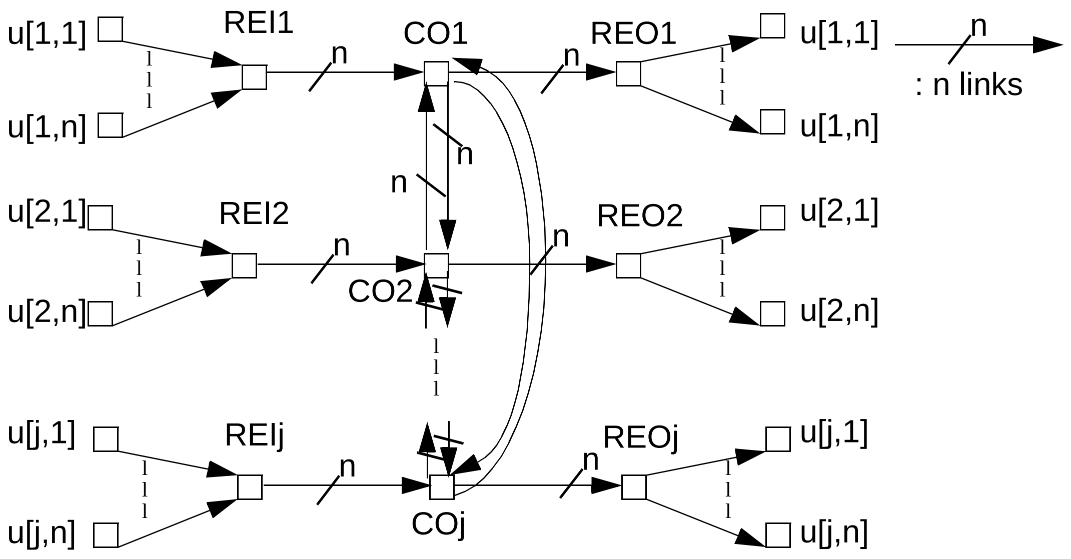


Figure 3. Net-n-j Topology

of users. This type of network topology is identified as Net-n-j, where n is the size of users connected to an RE unit and j is the number of COs. This regular network structure was used to try out the performance of the program for large networks.

Tables 2 and 3 show the results on a ring network which is similar to RingWithHub-16 except without the center hub, and a network of type Net-8-4, which has 44 vertices and 128 links. We also try out these algorithms on a real network which does not have the regular network structure and was used in many routing studies. Table 4 shows the results on such a network with the assumption that all vertices are multicast vertices. The link costs for networks in Tables 2-4 are generated by a random function. Figure 4 shows that for networks with as large as 500 vertices and 2000 links, the execution time performance of the RST heuristic algorithm is still quite good.

In summary, these results indicate that the optimal algorithm is not good for setting up connections with realtime constraints. It can be used, however, for small networks, and for long term reservation requests with small number of destination users. The RMV heuristic algorithm incurs more computation overhead than the RST algorithm but it can handle networks with small number of multicast vertices. The RST heuristic algorithm can find a multicast path within one second for networks with 200~300 vertices and 1000 links. For even larger networks, hierarchical algorithms must be used. See the discussion in Section 7.

6. Applications of Multipoint Connections in Multimedia Broadband Networks

The multicast path finding algorithm can be used within a network node or central office to set up connections among its internal switching elements, intra-office junctors, and interoffice trunks. It

	Optimal Algorithm		RST Algorithm		RMV Algorithm	
Destinations	Time(μsec)	Cost	Time(μsec)	Cost	Time(μsec)	Cost
2	3738623	245	3530	245	137013	245
3	3035860	268	3927	268	168072	268
4	3081676	437	6504	437	220896	437
5	5004564	506	6978	506	279298	509
6	4684517	649	8819	649	320138	652
7	8885562	745	9452	745	362018	748
8	33691894	853	11145	853	418909	853

Table 2. Performance on Ring-16-1 (32 vertices, 64 links)

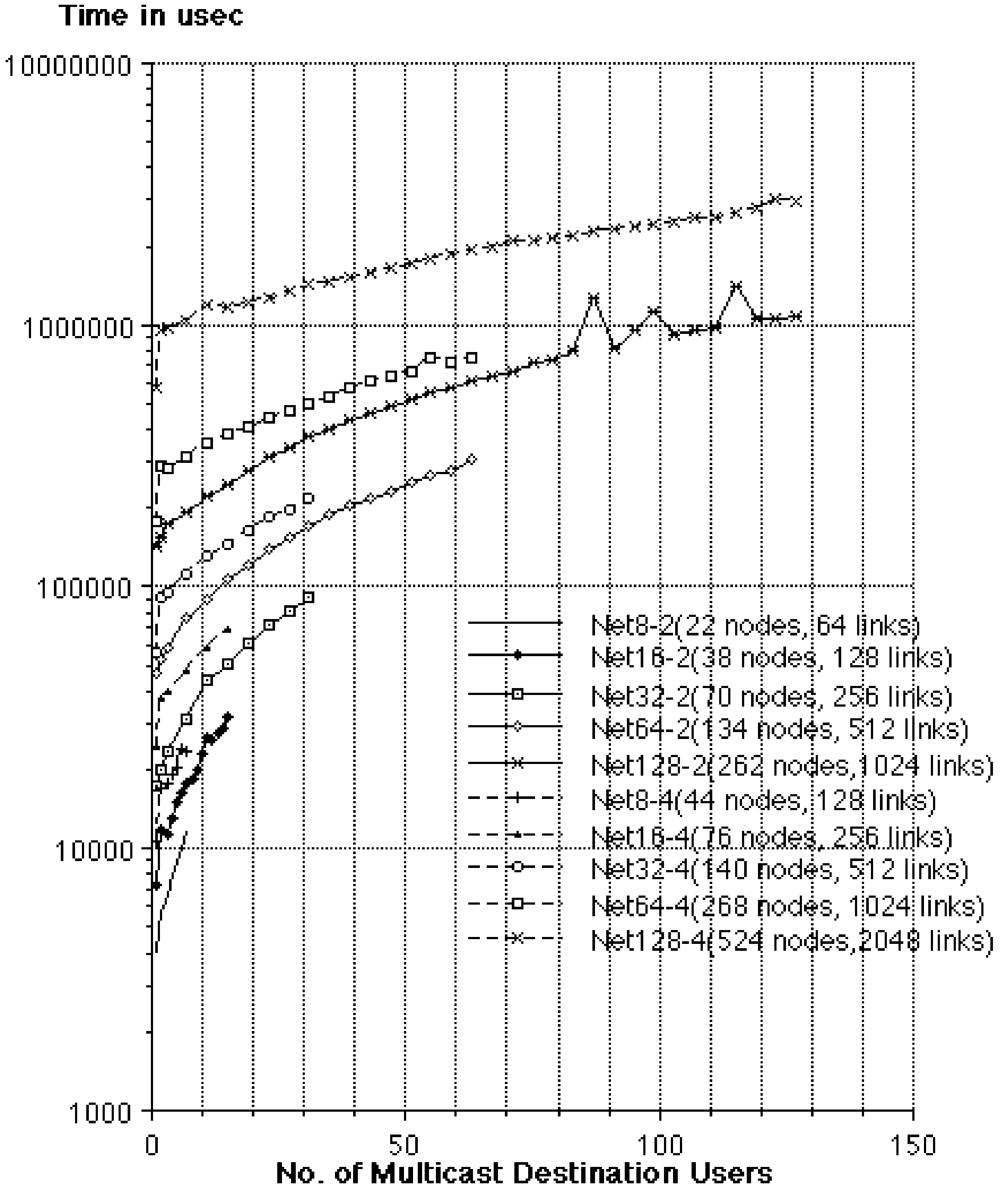
	Optimal Algorithm		RST Algorithm		RMV Algorithm	
Destinations	Time(μsec)	Cost	Time(μsec)	Cost	Time(μsec)	Cost
2	13732685	266	11815	279	249684	266
3	16490996	329	13310	342	301387	329
4	12749484	357	12201	357	442450	371
5	28928245	455	16418	457	581108	471
6	47542879	499	15032	501	644531	515
7	134365158	561	17116	563	659927	577
8	209514401	646	25133	648	786699	662

Table 3. Performance on Net-8-4 (44 nodes, 128 links)

	Optimal Algorithm		RST Algorithm		RMV Algorithm	
Destinations	Time(μsec)	Cost	Time(μsec)	Cost	Time(μsec)	Cost
2	26136588	1.70	15332	1.70	1446697	1.70
3	23719097	2.56	19907	2.56	2101778	2.56
4	45517107	3.68	39155	3.68	3219576	3.68
5	42485951	4.76	38601	4.77	2602598	4.77
6	149048089	5.20	43302	5.42	2551690	5.40
7	149852270	6.03	58478	6.25	3400339	6.23
8	176587456	6.67	62739	6.89	4315308	6.87

Table 4. Performance on Chicago-38-net (38 nodes, 1176 links)

Computation Result of RST Multicast Path Finding Algorithm



Note that the result was generated on a SUN Sparc Workstation and the program was written in GNU C++.

can also be used in a centralized routing scheme to set up interoffice, interexchange, or internetwork multicast paths. The protocol for setting up/modifying multicast paths and updating routing information among interoffice is an interesting research issue and has been an ongoing project here in Bellcore.

In this section we would like to illustrate how variants of the multicast path finding algorithm in conjunction with the shortest path algorithm can be used to set up other multipoint connections.

6.1 Concast

Convergence Cast, or Concast, is a connection from multipoint to a point. It will merge the traffic along the convergence path so that the closer it reaches the destination point the more bandwidth it requires for the links. A typical application of concast connections is a monitoring system which collects information sent from a group of geographically separated sensors.

Since the concast path is a multicast path with all its edges in reverse direction and with difference in bandwidth requirement in edges, the heuristic multicast path finding algorithm can be modified to traverse the edge in reverse direction and to increase the reserved edge bandwidth along the path.

6.2 Information Distribution Services

A information provider may have several centers within a network that have identical copies of certain pieces of information. The problem of distributing information to the subscriber within the network is a multiple multicast connection problem [Bus90]. It can be solved by the combination of network partition and the multicast path find algorithm. The difficulty lies in finding the optimal partition of local exchanges to each of these distribution centers. The multicast path finding algorithm plays a role in calculating the multicast path cost for each proposed partition and in setting up/ modifying the multicast paths that really distribute the information.

6.3 Video Conference with Centralized Presentation Control

Here we are considering a video conference service with participants located at geographically separated locations and with a centralized presentation control. The service ensures all the participants see the same image, either the composite image of all the participants, or a particular speaker, or viewgraphs. This type of services basically requires to set up a multipoint-to-multipoint connection.

In the simple case, it may imply finding within the network a conference bridge located at the optimal location which results in the cost of all the paths in the connection to be minimum. The paths include the shortest paths from all the participants to the video conference bridge and the multicast path from the video conference bridge to all the participants.

The image composition can also be done in distributed fashion by dividing participants into subgroups, merging subgroup’s images at distributed locations and then exchanging partially done composite image with other subgroups. The purpose here is to save bandwidth but it requires additional image composition facilities. If the presentation is dynamically changing, the control scheme for the distributed scheme can be very complicated.

6.4 Multiparty Video Call

In a multiparty video call, each party may select a subset of the video sources and compose them according to a presentation description into a single video stream for his/her display device. One possibility is to have one bridge gather all the video sources and compose images which are customized to each user’s presentation description. The other possibility is to merge some of the video sources along the way to a particular user. The latter approach requires more than one bridge but can save bandwidth and can be implemented with many simple modules. The trade-off depends on the cost ratio between bridges and links.

7. Discussion

We have presented three multicast path finding algorithms: an optimal algorithm based on the dynamic programming technique, a polynomial algorithm called RST multicast path finding algorithm which performs well for networks where almost all vertices are multicast vertices, and a polynomial algorithm called RMV multicast path finding algorithm which performs well for networks with small number of multicast vertices. The computation results from efficient implementations of these three algorithms indicate that the heuristic algorithms perform well and are suitable for the switch control software for connection management and resource allocation in large networks.

These multicast path finding algorithms can be applied in intra-switch and inter-switch situations. We have illustrated that the variants of them serve as a basis for setting up various multipoint connections.

For larger networks, a hierarchical approach can be applied to find multicast paths which span several switches. Since each group in the hierarchy is smaller, either the heuristic or the optimal algorithm can be applied. With the assumption that some network components may not have multicast capability, the hierarchical algorithms need to implement backtracking scheme dealing with negotiation between high level vertices to minimize cost. To set up a multicast path connection, vertex a may first assume that vertex b has multicast capability and only reserve one link to the other vertex. If vertex b gets responses from its low level vertices that the multicast capability is currently not available, then vertex a will have to retry to request multiple links to vertex b.

Multicast path finding is one of the functions needed in multimedia multiparty switching networks. The specification of the path is derived from the analyses of user-to-network signaling messages or network-to-network signaling messages. The user-to-network signaling protocol [Min89] provides users a language to express the multimedia, multiparty connections and their presentation control. The network-to-network signaling protocol is used by switches to exchange path related information and routing information to reserve or manage a connection. These two protocols post challenging research questions to the protocol and the network designers and open up new research directions for protocol specification and network routing fields.

Acknowledgment

The author would like to thank Yowjian Lin and Victor Mak for their valuable comments and their help in deriving the equation for mapping one-to-one ordered sequences of elements selected from a finite set into continuous natural numbers. The equation is used in the optimal multicast path finding algorithm for storing and retrieving paths. The author also thank Howard Bussey for his insightful comments.

References

[Bus89] Bussey, H. E. and Porter, F. D., “A Second Generation BISDN Prototype,” *International Journal of Digital and Analog Cabled Systems*, **1, 4,** January 1989.

[Bus90] Bussey, H. E., Egido, C., Kaplan A., Rohall, S. L., and Yuan, R., “Service Architecture, Prototype Description, and Network Implications of A Personalized Information Grazing Service,” *Proc. of Infocom’90*. pp. 1046-1053.

[Dee90] Deering, S. E. and Cheriton, D. R., “Multicast Routing in Datagram Internetworks and Extended LANs,” *ACM Trans. on Computer Systems*, **8**, **2**, May 1990, pp. 85-110.

[Dij59] Dijkstra, E. W., “A note on two problems in connexion with graphs,” *Numerische Mathematik* **1**, 1959, pp. 269-271.

[Dre72] Dreyfus, S. E. and Wagner, R. A., “The Steiner problem in graphs,” *Networks*, **1**, 1972, pp. 195-207.

[Karp72] Karp, R. M., “Reducibility among combinatorial problems,” In *Complexity of Computer Computations* (R. E. Miller and J. W. Thatcher, Eds.), Plenum Press, New York, 1972, pp. 85-104.

[Kru56] Kruskal, J. B., Jr., “On the shortest spanning subtree of a graph and the traveling salesman problem,” *Proc. Amer. Math. Soc.* **7:1**, 1956, pp. 48-50.

[Min90] Minzer, S. E. and Spears, D. R., “New Directions in Signalling for Broadband ISDN,” *IEEE Comm. Magazine*, Feb. 1989, pp. 6-14.

[Ray86] Rayward-Smith, V. J. and Clare, A., “On Finding Steiner Vertices,” *Networks*, **16**, 1986, pp. 283-294.

[Tak80] Takahashi, H. and Matsuyama, A., “An approximate solution for the Steiner problem in graphs,” *Math. Japonica*, **6**, 1980, pp. 573-577.

[Tu90] Tu, S-C. and W-H. Leung, “Multicast Connection-Oriented Packet Switching Networks,” *Proc. of ICC’90*, 1990, pp. 308.4.1-7.

[Wax88] Waxman, B. M., “Routing of Multipoint Connections,” IEEE J. Select. Areas Comm., **6**, **9**, 1988, pp. 1617-1622.

[Win87] Winter, P., “Steiner Problem in Networks: A Survey,” *Networks*, **17**, 1987, pp. 129-167.