Maximum Reliability K-Hop Multicast Strategy in Tree Networks

Mugurel Ionut Andreica, Nicolae Tapus

MAXIMUM RELIABILITY K-HOP MULTICAST STRATEGY IN TREE NETWORKS

Mugurel Ionut Andreica, Nicolae Tapus

Polytechnic University of Bucharest, Computer Science Department, Bucharest, Romania

ABSTRACT

In this paper we consider directed tree networks, for which the reliability of each edge (a real number between 0 and 1) is known. For these networks, we investigate the problem of finding a k-hop multicast strategy of maximum reliability. This problem is equivalent to the k-station placement problem, which was previously solved in polynomial time for trees. We present here $O(k \cdot n^2)$ and $O(k \cdot n^3)$ dynamic programming algorithms for the problem, which improve upon the previous best known solution, which is $O(k \cdot n^2 \cdot log(n))$ and rather complicated to implement. We then extend the algorithms to general directed graphs and also present some new algorithms for this case.

***Index Terms*—** maximum reliability, k-hop multicasting, k-station placement problem, tree network, reliable multicast.

1. INTRODUCTION

The reliability of network nodes and links is an important aspect which needs to be considered when developing fault-tolerant distributed algorithms. Usually, the reliability is only a statistical measure, representing the probability that the network node/link will not fail. In this paper, we consider the reliability of network links in a tree network, in the context of developing a multicast content distribution strategy with the highest reliability, subject to restrictions regarding the number of intermediate hops. We then consider general directed graphs and see how the results obtained for tree networks can be extended to this case. In the end of the paper we choose a slightly different reliability metric and present some exact algorithms for that case.

The paper is structured as follows. In Section 2 we define the problem of finding an optimal k-hop multicast strategy. In Sections 3 and 4 we present two algorithms for the problem. In Section 5 we extend the algorithms to general directed graphs and in Section 6 we consider a max-min reliability metric. Finally, in Section 7 we present related work and in Section 8 we conclude.

2. MAXIMUM RELIABILITY K-HOP MULTICAST

We are given a directed tree T with *n* nodes, in which the root of the tree wants to distribute some content to a set of destinations, which are the leaves of the tree. In order to send the content from a node u to a node v located in the subtree of u, the node u establishes a direct connection to node v and sends a message with the content on that connection. The transmission lasts for a fixed amount of time (one time unit). A node may establish any number of simultaneous connections to the nodes in its subtree.

Each directed edge *(u,v)* of the tree (oriented from u to v) has an associated reliability $r_{u,v}$. The reliability of transmitting a message on a direct connection from u to v is equal to the product of the reliabilities of the edges on the path from u to v. The reliability of the content distribution strategy is the product of the reliabilities of all the message transmissions performed. We are interested in finding a multicast strategy having the maximum reliability, subject to the constraint that it should not last for more than *k* time units. It is obvious that the root of the tree can send the content to every leaf during a single time unit, but the reliability of this strategy is

$$\prod_{(u,v)\in T} r_{u,v}^{nr\_leaves(v)}$$

where *nr_leaves(v)* is the number of leaves located in the subtree rooted at v.

By using intermediate nodes, the reliability of the strategy can be improved. We will now define the *k-hop multicasting problem*. We will build k+1 sets of nodes: S_0, S_1,...,S_k. The root is the only node in S_0. In the first time unit, the root sends a message to a subset of nodes S_1; every leaf is either in S_1 or is a descendant of exactly one node X in S_1. During the i^{th} time unit ($2 \le i \le k-1$), each node X in S_{i-1} which is not a leaf, sends a message to a subset $S_{i,X}$ of nodes from its subtree, such that

each leaf which is a descendant of X either belongs to $S_{i,X}$ or is also a descendant of exactly one node in $S_{i,X}$. The set of nodes S_i is the union of the sets $S_{i,X}$, for each X in S_{i-1}. During the k^{th} time unit, each node X in S_{k-1} which is not a leaf must send a message to each leaf node which is a descendant of X (the leaves receiving the message in the k^{th} time unit form the set S_k). The nodes belonging to the union of the sets S_i are called intermediate nodes. Every intermediate node (except the root) receives the content from exactly one other intermediate node. Obviously, there can be many multicast strategies and we are interested in the one with the maximum reliability.

By replacing the reliability of each edge *(u,v)* by $cost(u,v)=-log(r_{u,v})$, the requirement to maximize the reliability becomes equivalent to minimizing the total cost of message transmissions, where the cost of sending a message is equal to the sum of the costs of the edges composing the connection along which the message is sent.

3. A DYNAMIC PROGRAMMING SOLUTION

First, we will transform the directed tree into a binary directed tree. This transformation is quite standard. For each node i having $q>2$ sons $s_1,s_2,...,s_q$, we keep his first son s_1 and insert an extra node x as his second son. We make $s_2,..., s_q$ the sons of x and then recursively repeat the procedure for the node x and for the son s_1. The edge between i and x will have cost 0 (or, equivalently, reliability 1).

With this modified tree, we will compute the following values in a bottom-up fashion: $C(i,j,p)$ = the minimum cost of distributing the content to all the leaves in the subtree of node i, using at most j ($0\le j\le k$) time units and considering only the edges in i's subtree when computing the cost and:

- if $1<p\le n$, then i is on the paths between an intermediate node x located above i and p intermediate nodes located below i to which x sends messages directly.
- if $p=1$, then either i is an intermediate node or i is on the path between an intermediate node x located above it and the only intermediate node y located below i to which x sends a message directly.

For all nodes i, we define $C(i,-1,p)=+\infty$. If i is a leaf, then for all $0\le j\le k$ we have $C(i,j,1)=0$ and $C(i,j,p)=+\infty$ (for $1<p\le n$). If i is not a leaf, then it has either one or two sons. If it has only one son *s*, we have the following equations:

$$C(i,j,p)=p\cdot cost(i,s)+C(s,j,p),1<p\le n$$

$$C(i,j,1)=\min\begin{cases}\min\limits_{1\le p1\le n}\{p1\cdot cost(i,s)+C(s,j-1,p1)\}\\ cost(i,s)+C(s,j,1)\end{cases}$$

The first case is straightforward: i is not an intermediate node, therefore it lies on the paths between p intermediate nodes below it and another intermediate node above it. Therefore, i will have to forward p messages to the p intermediate nodes on the edge *(i,s)*. In the second case, either i is an intermediate node and consumes one time unit and the number of intermediate nodes to which i sends a message directly can be any number p_1, or i is not an intermediate node, and we find the same situation as before.

If i has two sons s_1 and s_2, we have the equations:

$$\min_{1<p\le n}C(i,j,p)=\min_{1\le p1<p}\begin{Bmatrix}p1\cdot cost(i,s1)+C(s1,j,p1)+\\ (p-p1)\cdot cost(i,s2)+C(s2,j,p-p1)\}\end{Bmatrix}\quad C(i,j,1)=\min_{\substack{1\le p1,1\le p2,\\ p1+p2\le n}}\begin{Bmatrix}p1\cdot cost(i,s1)+C(s1,j-1,p1)+\\ p2\cdot cost(i,s2)+C(s2,j-1,p2)\}\end{Bmatrix}$$

In the first case, i is not an intermediate node, and all the p messages coming from the intermediate node closest to i and above it will be forwarded to the p intermediate nodes below it. Out of these, p_1 intermediate nodes are located in s_1's subtree and $p-p_1$ are located in s_2's subtree. In the second case, i is an intermediate node and consumes one time unit and there are p_1 intermediate nodes located in s_1's subtree and p_2 in s_2's subtree, to which i will send the message directly. If i is an extra node inserted during the tree transformation, then i has two sons, but cannot act as an intermediate node, so C(i,j,1) will be $+\infty$ for all values of j.

If i is the root of the tree and has only one son s, then the only entry defined is:

$$C(root,k,1)=\min_{1\le p1\le n}\{p1\cdot cost(root,s)+C(s,k-1,p1)\}$$

If the root has two sons s_1 and s_2, then the only entry defined is $C(root,k,1)$. This entry represents the minimum cost of the k-hop multicast strategy. In order to actually find the strategy, we can trace the way the $C(i,j,p)$ values were computed and for each intermediate node we can find out to which other intermediate nodes it sends messages directly.

Let's analyze the time complexity of the algorithm. The most complex case is when a node i has two sons. For each $0\le j\le k$ and $1<p\le nr\_leaves(i)$, C(i,j,p) can be computed in O(n) time. For each j and p=1, C(i,j,1) can be computed in $O(n^2)$ time. The time complexity for a node i with two sons is $O(k\cdot n^2)$ and the overall time complexity is $O(k\cdot n^3)$.

4. AN IMPROVED SOLUTION

We will present now a dynamic programming solution with a better time complexity. First, we assign a label from 1 to M to each leaf of the tree, where M is the total number of leaves. The j^{th} leaf visited by a depth-first traversal of the tree (starting from the root) receives the label j. It is obvious that the labels of the leaves located in the subtree of a node i (including node i itself) form an interval of consecutive values, denoted by [lmin(i), lmax(i)]. This interval can be computed in O(n) time for all the nodes, with a simple bottom-up traversal. We will also compute in $O(n^2)$ time the values *dist(i,j)*=the sum of the costs of the edges on the directed path from i to j. We will now compute the values *C(i,j,p)*=the minimum cost for distributing the content to the first p leaves in node i's subtree (denoted by ST_i), using at most j time units, with i being an intermediate node. The first p leaves are the leaves labeled lmin(i), …, lmin(i)+p-1. If i is a leaf with label q, then C(i,j,1)=0 for all the values of j and lmin(i)=lmax(i)=q. If i is not a leaf, then we have:

$C(i,j,0)=0, 0\le j\le k\,;\, C(i,0,p)=+\infty, 1\le p\le nr\_leaves(i)$

$$C(i,j,p)=\min_{\substack{1\le j\le k,\ p\ge 1\\ x\in ST_i\\ lmax(x)=lmin(i)+p-1}}\left\{\begin{matrix} C(x,j-1,nr\_leaves(x))+\\ dist(i,x)+C(i,j,lmin(x)-lmin(i))\end{matrix}\right\}$$

The last equation considers the following case: node i sends a message to a node x in ST_i with lmax(x) equal to the label of the p^{th} leaf in ST_i, letting x take care of sending the content further to all the leaves in ST_x, using at most j-1 time units; node i takes care of the lmin(x)-lmin(i) remaining leaves (with labels in [lmin(i), lmin(x)-1]), using at most j time units. Each of the O(n) nodes x in ST_i must only be considered for only one of the O(n) values of p, equal to lmax(x)-lmin(i)+1. Thus, as a preprocessing step, for each node i, we will compute in O(n) time an array of lists L(i), where L(i,p) is a list containing all the nodes x in ST_i with lmax(x)=lmin(i)+p-1. When computing C(i,j,p), only the nodes x in L(i,p) will be considered. This way, we can compute the values C(i,j,p) in O(n) time for a given pair (i,j) and all the values of p (and in O(1) amortized time for each tuple (i,j,p), if we also do not add to L(i,p) any node x in ST_i with exactly one son). The time complexity of the algorithm is $O(k\cdot n^2)$ and the minimum cost is *C(root,k,M)*.

5. EXTENSIONS TO GENERAL DIRECTED GRAPHS

We will now consider the situation in which a general directed graph G with n nodes is given, where each directed edge (u,v) has an associated reliability $r_{u,v}$. We may consider that the given graph is complete - if some edges (u,v) should not be used, we can set $r_{u,v}$=0. A source node *src* needs to send some important content to a subset D of nodes (called destinations), using a k-hop multicast strategy. In this case, we need to find a directed tree rooted at the node *src*, where all the leaves belong to the set D and all the nodes in D belong to the tree, for which a k-hop multicast strategy with maximum reliability can be developed. If the tree is known, we can use one of the algorithms presented before, with some slight adjustments: if a node i which is not a leaf belongs to the set D, then we will insert a leaf node i', whose parent will be the node i. The directed edge (i,i') will have cost 0. Afterwards, we remove the node i from D and insert the leaf node i' in D in its place. This way, all the nodes in D will be leaves in the multicast tree.

Finding the multicast tree with the minimum cost strategy (with the costs defined like in the previous case) is NP-hard, as it is a variation of the well-known Steiner tree problem. We will now present an exact exponential time algorithm for finding the maximum reliability k-hop multicast strategy in general directed graphs, based on the second algorithm presented for directed tree networks. We will compute in $O(n^3)$ time all the values *dist(i,j)*=the minimum sum of the edge costs of a directed path in G between the pair of nodes (i,j). Afterwards, we will compute the values: *C(i,j,S)*=the minimum cost of distributing the content to all the nodes in S, starting from node i and using at most j time units. S is a subset of nodes from D. We have:

$$C(i,j,\{i\}\cap D)=0, 0\le j\le k\,;\, C(i,0,S)=+\infty, S\ne\{i\}\cap D$$

$$C(i,j,S)=\min_{\substack{W\subseteq(S\setminus\{i\})\\ W\ne\phi\\ 1\le x\le n}}\{dist(i,x)+C(x,j-1,W)+C(i,j,S\setminus W)\}$$

The algorithm is similar to the one for directed trees, except for the parameter describing the subset of nodes. For each pair (i,j), we have $2^{|D|}$ subsets S. For each subset S, we need to consider all the subsets W which are included in S. There are $C_{|D|}^{q}$ subsets S with q nodes and for each of them, there are 2^q subsets W. Overall, we have:

$$\sum_{q=0}^{|D|}C_{|D|}^{q}\cdot 2^q=3^{|D|}\cdot$$

The time complexity is $O(k\cdot n^2\cdot 3^{|D|})$. The algorithm can be implemented by considering the sets S in ascending order of their cardinality. The multicast tree and the multicast strategy can be obtained from the values *C(i,j,S)* (or by storing additional information about the way the values were computed), starting from *C(src, k, D)*. The first algorithm for directed trees can also be extended to directed graphs, with a time complexity of $O(k\cdot n^3\cdot 3^{|D|})$.

6. A MAX-MIN RELIABILITY METRIC

In this section we study a different problem, in which a directed graph G is given, where each directed edge (u,v) has an associated reliability $r_{u,v}$, duration $d_{u,v}$ and transmission processing time $tp_{u,v}$. Moreover, a source node *src* and a set of destinations D are given. The source node needs to send a piece of content to all the nodes in D, under the following conditions:

- the time after which every node in D receives the content is at most K
- the minimum reliability of an edge traversed by a message must be maximized

A node u can send a message directly to a node v only if the directed edge (u,v) exists in G. If the message is sent at time t, the node v receives the message at time $t+d_{u,v}$. Then, node u can send the next message at time $t+tp_{u,v}$.

Our solution is based on binary searching the minimum reliability R of an edge traversed by a message. Then, all the edges (u,v) with reliability $r_{u,v}<R$ are discarded from the graph, obtaining a new graph G'. The feasibility test consists of determining the minimum time multicast strategy in G' and verifying if the minimum time is at most K; if it is, then a larger value of R is tested; otherwise, we will test a smaller value. When all the values $tp_{u,v}$ are 0, the minimum time multicast strategy is obtained using the well-known pruned Dijkstra method [7] with the durations $d_{u,v}$ in G'.

We will now present an algorithm based on a technique similar to the one in the previous section. We compute Tmin(i,S)=the minimum duration of distributing the content to all the nodes in $D \bigcap S$, starting from the node i; S is the set of nodes in node i's subtree (including i). We have:

$$\mathrm{Tmin}(i,\{i\}) = 0;\ \mathrm{Tmin}(i,S) = 0, \text{if } S \bigcap D = \phi$$

$$\mathrm{Tmin}(i,S) = \min_{\substack{W \subseteq (S\setminus\{i\}) \\ W \neq \phi}} \{ \max_{x \in W} \left\{ \begin{cases} d_{i,x} + \mathrm{Tmin}(x,W) \\ 0, \text{if } (S \setminus W) \bigcap D = \phi \end{cases} \right. \left. \begin{cases} \\ tp_{i,x} + \mathrm{Tmin}(i, S \setminus W), \text{otherwise} \end{cases} \right\}$$

Tmin(src, {1,2,…,n}) is the minimum multicast duration. The time complexity of the algorithm is $O(n^2 \cdot 3^n)$.

We will now consider a special case, which occurs frequently: $d_{u,v}=tp_{u,v}=1$ and |D| is bounded by a constant C. The values of $d_{u,v}$ and $tp_{u,v}$ suggest that during every time unit, a node may either send a message or receive a message (but not both). For this case, we present an $O(C \cdot n^{2 \cdot C})$ polynomial time algorithm for determining the minimum time multicast strategy. We denote the C vertices in D by d_1, d_2, …, d_C. We will compute a table Tmin, where each entry is a tuple of C nodes. $\mathrm{Tmin}[(v_1,v_2,\ldots,v_C)]$ represents the minimum amount of time after which the message destined to each node d_i reached the node v_i. Initially, we have Tmin[(src,…,src)]=0 and all the other entries are uninitialized (denoted by a special value Ω). The algorithm uses a queue of states Q and considers transitions from a state S_1 to another state S_2: a transition lasts for one time unit and every node u_i in S_1 sends the message to the receiving node v_i in S_2 (if $v_i \neq u_i$) or does not send anything ($v_i=u_i$); (u_i,v_i) is called a transition pair.

Q={(src,…,src)}
while (*Q≠empty*)
  $(u_1, \ldots, u_C)$=Q.head_element(); Q.remove_head()
  for each state *$(v_1,\ldots,v_C)$* **such that** *$\mathrm{Tmin}[(v_1,\ldots,v_C)] = \Omega$* **do**
    if (*$\exists i.((u_i \neq v_i)$* **and** *$((u_i,v_i) \notin G'))$* **then continue**
    transition_ok=true // transition from $(u_1,..,u_C)$ to $(v_1,..,v_C)$
    for each *x* **in** *$\{u_1,\ldots,u_C,v_1,\ldots,v_C\}$* **do**
      *snd_x=***cardinality**(*$\{(u_i,v_i) | 1 \leq i \leq C$* ***and*** *$(u_i=x)$* **and** *$(v_i \neq x)\}$*)
      *rcv_x=***cardinality**(*$\{(u_i,v_i) | 1 \leq i \leq C$* ***and*** *$(v_i=x)$* **and** *$(u_i \neq x)\}$*)
      if (*$snd_x+rcv_x > 1$*) **then** *transition_ok = false*
    if (*transition_ok = true*) **then**
      $\mathrm{Tmin}[(v_1,\ldots,v_C)] = \mathrm{Tmin}[(u_1,\ldots,u_C)]+1$
      Q.add_to_tail($(v_1,\ldots,v_C)$)

$\mathrm{Tmin}[(d_1,\ldots,d_C)]$ contains the minimum time after which all the destinations receive the message from the source node. The multicast strategy can be computed from the values stored in the table Tmin. This table contains n^C states and for each state, $O(n^C)$ transitions to other states are considered, with O(C) time spent per transition. The time complexity is $O(C \cdot n^{2 \cdot C})$. The algorithm can be extended to handle other communication constraints, like the following ones: during every time unit, a node u may process at most $O_{s,u}$ send operations, at most $O_{r,u}$ receive operations and at most $O_{s+r,u}$ operations overall (send+receive). In this case, we only need to count the number of operations of each type performed by a node during a transition from one state to another and invalidate the transition if the numbers are too large. The algorithm can also be

extended to the case when the source wants to send a different message to each destination. In this case, two transition pairs (u_i,v_i) and (u_j,v_j) with $u_i=u_j$, $v_i=v_j$, $u_i \neq v_i$ and $i \neq j$ must be considered distinct, because the two message transmissions between u_i and v_i correspond to two different messages.

7. RELATED WORK

Reliable multicast strategies and reliable multicast trees have been the object of many research papers, like [1,2,3]. The k-station placement (k-SP) problem (equivalent to the k-hop multicasting problem) was investigated in [4] and was solved in polynomial time for directed trees, with a time complexity of $O(k \cdot n \cdot M(n))$, where $M(n)$ is the fastest min-cut algorithm on a graph with n vertices and $O(n)$ edges. $M(n)$ is $O(n \cdot log(n))$ [5] and the time complexity of the algorithm in [4] is $O(k \cdot n^2 \cdot log(n))$, but the implementation is cumbersome. We are not aware of any other results on the k-SP problem. Finding optimal multicast trees and strategies in directed graphs is related to the well-known Steiner tree problem, for which many types of algorithms were developed [6,7].

8. CONCLUSIONS

In this paper we presented two algorithms for computing the maximum reliability k-hop multicast strategy in directed tree networks, which is equivalent to the k-SP problem. The algorithms are either faster or easier to implement than the previous best known solution. They are interesting for computing optimal multicast strategies which can either be implemented directly or can be used in the performance evaluation of other multicast techniques. We also extended the algorithms to general directed graphs, but the time complexity became exponential. In the end, we presented exponential and polynomial time algorithms for a max-min multicast reliability metric and a different transmission model (without any long-distance connections).

9. REFERENCES

[1] I. Miloucheva, N. Reyes, J. Mahnke, and K. Jonas, "Efficient reliable on-demand multicast for content delivery," *Proceedings of the International Conference on Software and Computer Networks*, IEEE Press, 2006.

[2] S. Kinoshita, T. Shiroshita, and T. Nagata, "The RealPush Network: a new push-type content delivery system using reliable multicasting," *IEEE Transactions on Consumer Electronics*, IEEE Press, pp. 1216-1224, 1998.

[3] J.R. Lai, and W. Liao, "Mobile Multicast with Routing Optimization for Recipient Mobility," *IEEE Transactions on Consumer Electronics*, IEEE Press, pp. 199-206, 2001.

[4] C. Galdi, C. Kaklamanis, M. Montangero, and P. Persiano, "Optimal and Approximate Station Placement in Networks", *Proceedings of the 18th Symposium on Theoretical Aspects of Computer Science (STACS), LNCS 2010*, Springer Verlag, Heidelberg, pp. 271-282, 2001.

[5] G. Borradaile, and P. Klein, "An O (nlogn) algorithm for maximum st-flow in a directed planar graph," *Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, ACM Press, pp. 524-533, 2006.

[6] C.-H. Chow, "On Multicast Path Finding Algorithms," *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, IEEE Press, pp. 1274-1283, 1991.

[7] A. Penttinen, "Minimum cost multicast trees in ad hoc networks," *Proceedings of the IEEE International Conference on Communications (ICC)*, IEEE Press, pp. 3676-3681, 2006.