Rapid Rumor Ramification: Approximating the minimum broadcast time

(Extended Abstract)

R. Ravi [*]
DIMACS Center
Department of Computer Science
Princeton University
Princeton, NJ 08544-2087

Abstract

Given an undirected graph representing a network of processors, and a source node containing a message that must be broadcast to all the nodes, find a scheme that accomplishes the broadcast in the minimum number of time steps. At each time step, any processor that has received the message is allowed to communicate the message to at most one of its neighbors in the network, i.e. can communicate via a telephone call to a neighbor. This has been termed the minimum broadcast time problem under the telephone model and is known to be NP-complete.

The minimum broadcast time in a graph is closely related to the poise of the graph. The poise of a tree is defined to be the quantity (maximum degree of any node in the tree + diameter of the tree). The poise of a graph is the minimum poise of any of its spanning trees. Computing the poise of a graph is shown to be NP-hard and an algorithm for computing a spanning tree of approximately minimum poise is derived. This algorithm is then used to derive an $O(\frac{\log^2 n}{\log \log n})$*-approximation for the minimum broadcast time problem on an n-node graph.*

Our algorithm extends to many generalizations of the problem such as the multicast problem, a telephone model allowing conference calls, and to the closely related minimum gossip time problem.

[*]Most of this work was done while the author was at the Department of Computer Science, University of California, Davis, and supported by NSF Grant CCR-91-03937. The author also acknowledges support from a DIMACS postdoctoral fellowship.

1 Introduction

Motivation and formulation

The minimum broadcast time problem is the following. We are given a connected, undirected graph representing a set of processors interconnected by an arbitrary configuration of bidirectional links between them. The model of communication assumes a global clock, with the restriction that each processor can communicate (send or receive) with at most one of its neighbors in the graph per clock step. This model and its extensions have been variously termed as the telephone model, the telegraph model, or the single-port interconnection architecture. Given a source processor with a message, the problem is that of finding a scheme which disseminates this message to all the processors in the network in the minimum number of clock steps. This problem is known to be NP-complete [10], even in 3-regular planar graphs [21]. If the underlying undirected graph models communication possibilities between a group of people represented by the nodes, and the source node represents a person carrying a piece of rumor that is to be propagated to everyone in the graph, then the problem becomes one of rapid rumor ramification.

Let $b_v(G)$ denote the minimum broadcast time with v as the source node in a graph G. Define the minimum broadcast time of a graph G as $b(G) = \max_{v \in G} b_v(G)$. The following lower bound on the minimum broadcast time is immediate from the observation that the number of informed processors at most doubles with every time step.

Lemma 1 *For any source vertex r in a graph G,*

$$b_r(G) \geq \lceil \log_2 |G| \rceil$$

Previous work and our main result

Given that broadcasting is an important basic primitive in communication networks, it has received a lot of interest among researchers in the past [3, 4, 5, 6, 7, 8, 11, 12, 13, 20, 27, 31, 32]. [13] is a comprehensive survey of the literature related to the broadcast problem and its relative, the gossip problem, wherein every node has a unique message that must be disseminated to every other node. A substantial portion of the work in this area has been aimed at constructing good *broadcast graphs* on a given number of nodes. A broadcast graph on n nodes is one with the minimum number of edges in which broadcast from any source can be completed within the lower bound in Lemma 1. This line of work is pursued in [3, 4, 5, 11, 12, 20, 31, 32]. Fault-tolerant broadcast graphs were considered in [3, 19] while some generalizations of the problem were addressed in [27]. [30] considers broadcasting on trees and provides an optimal algorithm for the minimum broadcast time problem that runs in polynomial time while [8] consider the problem in trees with multiple originators. Broadcasting in grid-graphs was considered in [7].

In contrast, there is little known work on estimating the broadcast time of arbitrary networks, though many of the above-mentioned articles allude to this problem. Scheuermann and Wu [29] proposed a heuristic algorithm that works by computing a maximum matching along which to accomplish the next step in the broadcast scheme; no guarantees were provided. Feige, Peleg, Raghavan, and Upfal [9] analyze a randomized broadcasting scheme in arbitrary networks and and derive bounds on the expected and almost sure coverage times for such broadcast schemes. They also analyze the performance of such schemes in hypercubes and random graphs.

The only known approximation algorithm for the minimum broadcast time problem prior to this work was that of Kortsarz and Peleg [16]. They provide an approximation algorithm that computes a scheme that completes in time that is within a constant factor of the minimum plus an additive factor of $O(\sqrt{n})$ in an n-node graph. They give better approximation algorithms for chordal graphs, outerplanar graphs, series-parallel graphs, and trees of cliques.

The main result of this paper is the first polylogarithmic approximation algorithm for the minimum broadcast time problem.

Theorem 1 *There is an $O(nm\log^2 n)$-time algorithm that, given an undirected graph G on n nodes with m edges and a source node r, computes a broadcast scheme that completes in time $O(b_r(G)\frac{\log^2 n}{\log\log n})$.*

Our work improves the approximation result of Kortsarz and Peleg by removing the large additive $O(\sqrt{n})$ term. To achieve our result, we investigate a quantity closely related to the broadcast time of a graph that we call the *poise* of the graph. We believe that the poise of the graph is interesting in its own right and may have other applications as well. In what follows, we motivate the definition of the poise of a graph.

Broadcast arborescences

Any broadcast scheme from a source node can be represented by a directed spanning tree or arborescence of the graph where the edges are directed away from the source. Given a broadcast scheme, this tree is defined by choosing, for every node other than the source, the unique edge along which the message is conveyed to the node for the first time, as the single edge directed into this node. It is easy to see that the digraph defined in this way is a outward-directed arborescence rooted at the source. Conversely, given an outward-directed arborescence rooted at the source, it is easy to recursively compute, working bottom-up, an optimum broadcast scheme that completes in the minimum number of steps using only the edges in the tree (For more details, see [30]). Observe that this spanning tree need not even be directed, since the choice of a source node can be used to direct the edges of an undirected spanning tree appropriately to form an arborescence.

Poise of a graph

We can use the characterization of a broadcast scheme as an arborescence to derive a good lower bound on the broadcast time in the graph. The telephone model restriction dictates that the maximum out-degree of any node in the arborescence defined by a broadcast scheme is a lower bound on the broadcast time for this scheme. The maximum depth of any node from the root in this arborescence is also a lower bound on the broadcast time for this scheme since only neighbors can be informed in a time step. If A^* corresponds to the arborescence defined by the optimum broadcast scheme for a source node r, then the two observations above imply that

$$\begin{aligned} b_r(G) &\geq \frac{1}{2}(\text{Max. out-degree}(A^*) + \text{Depth}(A^*)) \\ &\geq \frac{1}{2}\min_{\text{all arborescences } A}\{\text{Max. out-degree}(A) \\ &\qquad + \text{Depth}(A)\} \end{aligned}$$

Motivated by this we define the following problem: given an undirected graph, find an undirected spanning tree in which the quantity (maximum degree of any node in the tree + diameter of the tree) is minimum over all the spanning trees. For any tree T, we call the quantity (maximum degree of any node in the tree + diameter of the tree) the *poise* of the tree[1] and denote this value by $P(T)$. The poise of a graph G, denoted $P(G)$, is defined as the minimum poise of any of its spanning trees. A tree on n nodes with the least poise is one with branching x at each internal node and with diameter roughly x. The value of x then obeys $x^x = n$ solving to $x = \Omega(\frac{\log n}{\log\log n})$. Thus the poise of any graph on n nodes is $\Omega(\frac{\log n}{\log\log n})$.

We observed that the poise of a graph is a lower bound on the minimum broadcast time from any source node in the graph. The next theorem shows that it is a good lower bound.

Theorem 2 *Let $P(G)$ denote the poise of G and $|G| = n$. Then*

$$\Omega(P(G)) \leq b(G) \leq O(P(G)\frac{\log n}{\log\log n}).$$

The proof of the second inequality in the above theorem is constructive: given a tree with poise P, we demonstrate a broadcast scheme starting at any root in this tree which completes in $O(P\frac{\log n}{\log\log n})$ steps. This allows us to reduce the problem of finding a good broadcast scheme to one of finding a tree of minimum poise, within a factor of $O(\frac{\log n}{\log\log n})$. However, it is hard to compute the poise of an undirected graph.

Theorem 3 *Computing the poise of an undirected graph is NP-hard.*

The reduction is from the Hamiltonian path problem and is omitted. We present the first approximation algorithm for this problem.

Theorem 4 *There is an $O(nm\log^2 n)$-time algorithm to compute a spanning tree of an undirected graph on n nodes with m edges, such that the poise of this tree is within $O(\log n)$ times the poise of the graph plus an additive term of $O(\log^2 n)$.*

Given this theorem, we can now apply the results in Theorem 2 along with the lower bound in Lemma 1 to derive the approximation algorithm for minimum broadcast time in Theorem 1.

[1]This name is inspired by the fact that this quantity represents how well poised between width (the maximum degree) and height (the diameter) the tree is.

Extension: the multicast problem

Our algorithms extend naturally to a generalization of the broadcast problem commonly referred to as the *multicast* problem. As before, we are given a source node in an undirected graph, but we only require the message to reach a subset of the nodes called the *terminal nodes*. Note that the terminal nodes may not necessarily induce a connected subgraph so we may need to use other non-terminal nodes to accomplish the multicast. The objective is to devise a scheme that achieves the multicast in the minimum time. Since this problem generalizes the minimum broadcast time problem, it is NP-complete.

Our treatment above for approximating the broadcast time using spanning trees of minimum poise extends in a straightforward manner by considering the analogous problem of computing a Steiner tree for the terminal nodes with minimum poise. Details are in [24].

Gossiping in minimal time

The minimum gossip time problem [17] is an extension of the minimum broadcast time problem in which every node in the graph has a unique message that must be disseminated to every other node in the minimum number of time steps. As before, at each clock step, every node can only communicate with one of its neighbors. However any number of messages may be transmitted in a single communication step between two nodes.

The unlimited bandwidth assumption at each step immediately gives an approximate reduction between the minimum time broadcast and gossip problems. The minimum broadcast time from any node in a graph is a trivial lower bound on the minimum gossip time in the graph. On the other hand, a broadcast scheme from any node in the graph completing in time b can be used to accomplish gossip in time $2b$ as follows: Use the broadcast scheme in the reverse order to collect all the messages from all nodes at the root of the broadcast tree in b steps[2]. Then use the scheme in the forward direction to broadcast all the collected messages and complete the gossip in the next b steps.

Given this approximate equivalence between the two problems, our results for the broadcast time problem extend naturally to the minimum gossip time problem. The extension to minimum time gossip among a subset of terminal nodes is also equally direct

[2]This process has been termed *census-taking* in [1].

using our results on approximate multicasting. The details are omitted.

Extension: Conference calls

Our approach also extends to solving a variant of the broadcast problem where at every clock step, any processor can communicate with up to a given number, c, of its neighbors, thus allowing more communication per clock step. This extension of the telephone model can be thought of as one allowing "conference calls" with up to c neighbors per time step. This model is fairly realistic since such a bound may be imposed either by the communication hardware at each processor or by the high setup costs involved in exchanging a message with a neighbor. Our approximation algorithm extends to this version and we have the following theorem.

Theorem 5 *There is an $O(nm\log^2 n)$-time algorithm that, given an undirected graph G on n nodes with m edges, a parameter c and a source node r, computes a broadcast scheme allowing c conference calls to neighbors from every node at each clock step, such that this scheme completes in time $O(b_r^c(G)\frac{\log^2 n}{\log\log n})$ where $b_r^c(G)$ is the minimum broadcast time of any scheme for this root under this model. When $c = \Omega(P(G)\log n)$ where $P(G)$ denotes the poise of G, the performance ratio can be improved to $O(\log n)$.*

The above result generalizes in a straightforward manner to the corresponding multicast problem allowing conference calls. We omit a description of these extensions here.

Special cases: Networks with bounded degree or diameter

In the special case of graphs with bounded degree, we can approximate the minimum broadcast time within a constant factor, the factor depending on the bound on the degree.

Theorem 6 *There is a polynomial-time algorithm that, given an undirected graph in which the maximum degree of any node is bounded by a fixed number B, compute a broadcast scheme from a source r that completes in time $2B \cdot b_r(G)$.*

The above theorem also extends to the multicast problem on graphs of bounded degree.

The case of bounded diameter graphs is however nearly as hard as the original problem as testified by the following theorem.

Theorem 7
Given a polynomial-time ρ-approximation algorithm to the minimum broadcast time problem on a graph of bounded diameter (in fact, for a graph of diameter at most two), there is a polynomial-time $2\rho + 2$-approximation algorithm for the minimum broadcast time problem on any graph.

In the next section, we detail the connection between the minimum broadcast time and the poise of a graph. In Section 3, the algorithm for computing a spanning tree of approximately minimum poise is presented. Section 4 contains the proofs of a few extensions and we conclude with some open problems.

2 Broadcast schemes and trees of poise

We prove Theorem 2 in this section. Let r be a node from which the minimum time to accomplish broadcast is $b(G)$. Consider an optimum broadcast scheme finishing in time $b(G)$ starting at r. This defines an outward-directed arborescence rooted at r in a natural way as described in Section 1. Let T be the underlying undirected spanning tree defined by this arborescence rooted at r. Then $b(G)$ is at least as much as the maximum degree of any node in T minus one. Furthermore, $b(G)$ is also at least as much as the depth of the arborescence from which T is derived, and thus at least half the diameter of T. Since the poise of T is defined as the sum of the maximum degree and the diameter of T, we have that $b(G) = \Omega(P(G))$.

We prove the second inequality in Theorem 2 by showing that $b(G) = O(P(G)\frac{\log n}{\log P(G)})$, and applying the lower bound of $P(G) = \Omega(\frac{\log n}{\log\log n})$. To show the former claim, we exhibit a simple algorithm to complete broadcast in this many time steps starting from any root r and using only edges in a spanning tree of G with poise $P(G)$. Let T be such a spanning tree. Let T_r denote the outward arborescence derived from T by rooting it at r and directing all the edges in T away from r. The maximum out-degree of any node in T is at most $P(G)$ and the longest directed path in T has length at most $P(G)$.

Note that given the tree T and the root r, the minimum time to accomplish broadcast from r using only edges of T can be determined in polynomial time by using dynamic programming and working bottom-up in the rooted tree T_r [30]. However, we wish to bound the broadcast time of the resulting scheme in terms of the poise of T, so we use a broadcast scheme that is simpler to analyze.

The scheme to broadcast the message from the root r is specified completely by specifying for each internal node v in T_r, the order in which the children of v in T_r will be informed. We determine this order according to the number of nodes in the subtree of T_r rooted at this child. For any node v, let T_v denote the subtree of T_r rooted at node v and $|T_v|$ denote the number of nodes in T_v. We order the children of an internal node v of T_r as $v_1, v_2, \ldots, v_d$ where d is the out-degree of v and $|T_{v_i}| \geq |T_{v_{i+1}}|$ for $1 \leq i \leq d-1$. After the time at which v first receives the message from its parent in T_r, for the next d steps, it sends the message to its children in the order $v_1, v_2, \ldots, v_d$. The broadcast is then accomplished recursively in each of the subtrees T_{v_i}.

Let $B(n, d, \delta)$ denote the time taken by this scheme to complete broadcast using a directed tree on n nodes with maximum out-degree d and depth δ. We have the recurrence

$$B(n, d, \delta) = \max_{1 \leq i \leq d'} \{B(n_i, d, \delta - 1) + i\}$$

where n_i is the number of nodes in the ith subtree of the root, and d' is the out-degree of the root of this tree on n nodes. Since we ordered the subtrees in non-increasing order of size we have $n_i \leq \frac{n}{i}$.

Claim 1

$$B(n, d, \delta) \leq d \cdot \frac{\log_2 n}{\log_2 d} + \delta.$$

Proof: We verify that the above solution satisfies the recurrence for $B(n, d, \delta)$. The left-hand side becomes $d \cdot \frac{\log_2 n}{\log_2 d} + \delta$ while the right hand side has value $d \cdot \frac{\log_2 n_i}{\log_2 d} + i + \delta - 1$. It is easy to verify that this is at most the left-hand side using $n_i \leq \frac{n}{i}$ and that $\frac{d}{\log_2 d} \geq \frac{i}{\log_2 i}$ for $2 \leq i \leq d$. The case $i = 1$ is handled by using the slack of one in the additive δ term. □

Applying Claim 1 to the tree T_r and noting that the sum (maximum degree + depth) of this tree is $\Theta(P(G))$, we have that the broadcast completes in $O(P(G)\frac{\log n}{\log P(G)})$ steps.

The bound of $O(P\frac{\log n}{\log P})$ on the broadcast time of a scheme derived from a tree of poise P is existentially tight in the following sense: For any value of P, there is a tree of poise P using which any broadcast scheme takes $\Omega(P\frac{\log n}{\log P})$ steps to complete. An example of such a tree is a complete P-ary tree of depth roughly $\log_P n = \frac{\log n}{\log P}$.

3 Approximating the poise of a graph

A bicriteria formulation

To find a spanning tree of approximately minimum poise, we first recast this problem as the following bicriteria problem: given an undirected graph and a positive integer Δ at least as much as the diameter of the graph, find a spanning tree of the graph of diameter at most Δ such that the maximum degree of any node in this tree is minimum. We call this problem the diameter-constrained minimum degree spanning tree problem. It is easy to see that this problem is NP-hard since the degree-constrained spanning tree problem [10] is a special case of this problem without the diameter restriction. We provide the following bicriteria approximation result.

Theorem 8 *There is an $O(nm \log n)$-time algorithm that, given an undirected graph G on n nodes with m edges and a positive integer Δ greater than the diameter of G, finds a spanning tree of of degree $O(D^* \log n + \log^2 n)$ and diameter $O(\Delta \log n)$ where D^* is the minimum maximum degree of any spanning tree of diameter at most Δ.*

The above theorem is the essential ingredient in the proof of Theorem 4. Suppose we run the above algorithm with the value of Δ set to $P(G)$, then by the definition of $P(G)$ and the performance guarantees in Theorem 8, the output spanning tree has maximum degree $O(P(G) \log n + \log^2 n)$ and diameter $O(P(G) \log n)$. Thus this spanning tree has poise as claimed in Theorem 4. To complete the proof of Theorem 4, we must describe how we can determine $P(G)$ so as to run the above algorithm. To do this we simply run the above algorithm for decreasing candidate values of Δ starting from n to find the smallest value Δ such that the output spanning tree has maximum degree $O(\Delta \log n + \log^2 n)$. We can achieve the running time bound in Theorem 4 by performing a binary search for Δ in the range $[2, \ldots, n]$ and running the algorithm in Theorem 8 at every step of this search. Thus we use $O(\log n)$ invocations of the algorithm in Theorem 8 giving the running time in Theorem 4.

The proof of Theorem 8 uses the rough framework of the algorithms in [25] and randomized rounding [23] applied to a concurrent multicommodity flow problem. Before we prove this theorem, we describe some background material that will be useful in understanding our algorithm and its analysis.

Background

Concurrent multicommodity flow with length bounds, and randomized rounding

Concurrent flow is an optimization version of multicommodity flow defined by Shahrokhi and Matula [28]. For our purpose, we consider a multicommodity flow problem specified as follows: a network with a node-capacity of one on every node, and a set of *commodities*, each specifying a pair s_i, t_i of nodes and a demand of one unit of flow between them. The edges in the network are assumed to have unbounded capacity.

An assignment of flow is *feasible* if for every node, the flow through that node is at most its capacity. The instance is *feasible* if there exists a feasible flow. Even for a non-feasible assignment of flow, it is possible to measure how far it is from being feasible. For any assignment of flow satisfying the demands, the *congestion* u is the maximum ratio of flow through a node to the node-capacity (one in our case). The concurrent flow problem is to find the minimum congestion u, i.e. the minimum number such that the instance becomes feasible when all capacities are multiplied by u.

We consider a more restricted version of the congestion problem described above where we require each flow path to be bounded in length, i.e., number of edges in the flow path. We can use integer programming to set up a multicommodity flow problem on the input network G where each node has unit capacity and the number of edges in any flow path for any commodity is restricted to be at most L. We call this the L-bounded congestion problem. We can do this as in [18] where such a formulation is used in approximating network embeddings. We detail this below.

To limit the length of a flow path, we specify that for some input number L, no flow can travel more then L edges. We set up one unit of flow of commodity i from s_i to t_i. Let $N(v)$ denote the set of nodes adjacent to v in the input network. We interpret $f^i_d(uv)$ as being the flow of commodity i through edge (u, v) at distance d from s_i. Using these variables we have the following integer program.

Minimize u

subject to constraints

$$\sum_{k\in N(l)} (f^i_d(kl) - f^i_{d+1}(lk)) \quad \geq \quad 0$$
$$\forall i \in \{1,\ldots,p\},\quad l \in V,\quad 0 \leq d < L$$

$$\sum_{k\in N(s_i)} f^i_0(s_ik) \quad = \quad 1 \quad \forall i \in \{1,\ldots,p\}$$

$$\sum_{0\leq d\leq L}\sum_{k\in N(l)} (f^i_d(kl) - f^i_{d+1}(lk)) \quad = \quad \begin{cases} 1 & \text{if } l = t_i \\ 0 & \text{otherwise}\end{cases}$$
$$\forall i \in \{1,\ldots,p\},\quad l \in V$$

$$\sum_{k\in N(l)}\sum_{0\leq d\leq L}\sum_i (f^i_d(lk) + f^i_d(kl)) \quad \leq \quad u \quad \forall l \in V$$

$$f^i_j(lk) \quad \in \quad \{0,1\}$$
$$\forall i \in \{1,\ldots,p\} \quad \forall (l,k) \in E \quad \forall j \in \{0,\ldots,L-1\}$$

The first set of equations enforce flow conservation. They require that any flow that is traversing the $(d+1)^{st}$ edge on leaving a node l must have entered the node on an incoming edge that was the d^{th}, i.e., must have traveled distance d. The second and third groups of equations specify the conditions on the sources and sinks of each commodity. The fourth group of equations define the node-capacity constraints of the flow problem. The last set of constraints enforce integrality.

It is easy to find a *fractional* solution for the above linear program minimizing the congestion ratio in which a unit of flow is split among various paths carrying *fractional* flow values. This can be done by solving the above linear program without the integrality constraints. For our application, we shall require that flow paths carry *integral* flow values. Minimizing congestion under this condition is NP-complete [10]. However, Raghavan and Thompson have given a polynomial time algorithm for approximately minimizing congestion under the integral flow restriction. They show that given any fractional flow solution with congestion u_F, one can use their *randomized rounding* technique to obtain an integral flow solution[3].

Theorem 9 (Raghavan & Thompson) *Let the value of the minimum congestion for a fractional flow solution for an L-bounded congestion problem be denoted by u_F. Then there is an integral flow solution in which each integral flow-path has length at most L and the congestion u_I is $O(u_F + \log n)$, where n is the number of nodes in the graph.*

Raghavan [22] has shown that the rounding can be done deterministically and such an integral solution

[3]Though the original result of Raghavan and Thomson did not address the length-bounded flow problem, the extension of their result to this case, as observed in [18], is immediate.

can be found in polynomial time. Alternatively, one can use the fast approximation algorithm of Klein, Stein, and Tardos [15] to directly obtain an integral solution. For the special case we consider here, their algorithm can be shown to give an expected running time of $O(nm \log n)$ in the randomized version, with a factor of n overhead for the deterministic version.

A tree-pairing result

Next, we recall a tree decomposition result used in [14, 25].

Lemma 2 *Let T be a tree with an even number of marked nodes. Then there is a pairing $(v_1, w_1), \ldots, (v_k, w_k)$ of the marked nodes such that the $v_i - w_i$ paths in T are edge-disjoint.*

A pairing of the marked nodes that minimizes the sum of the sizes of the tree-paths between the nodes paired can be verified to obey the property in the lemma above. We use this lemma in the proof of the performance guarantee.

The approximation algorithm

We describe now the algorithm referred to in Theorem 8. Let Δ denote the bound on the diameter of the tree specified in the problem, and D^* the minimum maximum degree of any spanning tree of diameter at most Δ.

Overview

The algorithm begins with the empty set of edges as the solution subgraph where each node is in a singleton connected component. A connected component maintained in the current solution is called a *cluster*. The algorithm works in $O(\log n)$ iterations where n is the number of nodes in the original graph, merging clusters during each iteration by adding paths between them. We reduce the number of clusters by a constant fraction at each iteration via merging to ensure that the number of iterations is as desired.

The clusters maintained by our algorithm represent node-subsets of the input graph G. However, they *do not* represent a partition of the nodes of the input graph. This is because of the way in which we merge the clusters in the algorithm. For each cluster we maintain a spanning tree on the nodes in the cluster. The spanning trees of the clusters maintained by the algorithm are not necessarily edge-disjoint as a result of our merging procedure. We sketch this procedure below. We identify a *center* in the spanning tree of each cluster[4]. In each iteration, most of the clusters are grouped with other clusters, and one of the clusters in each group is chosen as the mid-cluster. For every cluster in a group other than the mid-cluster, we add a path from its center to the center of the mid-cluster. This path may involve nodes that occur in other clusters currently maintained by the algorithm. The addition of these paths allows us to merge all the clusters in this group into one. However, while merging, we ensure that the new cluster formed has at most one copy of any node or edge.

If we ensure that the paths added to merge any cluster in a group to the mid-cluster has length at most Δ, then we can ensure inductively that the spanning trees we maintain for the clusters have approximately low diameter. Furthermore, if we add paths to merge clusters such that the maximum degree of any node due to all the paths added in an iteration is small, since the number of iterations is small, we ensure that the spanning tree of the single cluster formed at the end of the algorithm also has low maximum degree. This is how the performance guarantees of the algorithm are proved.

The Algorithm

```
1  Initialize the set of clusters $\mathcal{C}$ to contain $n$ singleton sets, one for each node of the input graph.
   For each cluster in $\mathcal{C}$, define the single node in the cluster to be the center for the cluster. Initialize
   the iteration count $i := 1$.
2  Repeat until there remains a single cluster in $\mathcal{C}$
3    Let $\mathcal{C}$ denote the set of clusters at the beginning of this iteration.
     Define $V_i$ to be $\{v : v \text{ is the center of a cluster in } \mathcal{C}\}$, i.e. the set of all the centers of clusters in $\mathcal{C}$.
4    Set up a $\Delta + 1$-bounded congestion problem on the input graph $G$ as follows.
5      Define a commodity $j$ for each node $v_j$ in $V_i$. The source of this commodity is the node $v_j$
       itself. We connect all the nodes $v_k$ such that $k \neq j$ using directed arcs to a new node $t_j$.
       This new node is the sink for commodity $j$. The nodes are assigned unit capacity and the
       length bound on the flow-paths is $\Delta + 1$, where $\Delta$ is the diameter bound input to the problem.
6    Find an approximately optimal integral solution to the flow problem as in Theorem 9. This
```

[4] Intuitively, a center is a node that is roughly in the middle of a longest path in the tree.

```
        provides a set of flow paths, one for each commodity $j$ going from $v_j$ to some $v_k$ where $k \neq j$ and the length of this path is at most $\Delta$.
7       Construct an auxiliary digraph $G_i$ on the node set $V_i$ modeling the way the flow for the commodities were satisfied. For each commodity $j$, we include the arc $(v_j, v_k)$ in $G_i$ where $v_k$ is the destination node for the flow path for this commodity. This arc denotes a flow path $P_j$ of length at most $\Delta$ from $v_j$ to $v_k$ in $G$.
8       Identify a subgraph $H_i$ of the auxiliary graph $G_i$ that is a forest of directed trees. This step is elaborated below in Lemma 3. Each of these directed trees is an inward arborescence on at least two nodes in $V_i$ and each node of $V_i$ is in exactly one of these trees
9       For each directed tree $T_x$ in the forest $H_i$
10        Bicolor the arcs in $T_x$ by partitioning them into arcs in odd and even levels. The level of a node is its distance from the root. Odd level arcs are those with their heads pointing to nodes in odd-numbered levels and the even level arcs are defined similarly. Let $E_x$ denote the color class with the maximum number of arcs. Note that $E_x$ is a set of stars, where all the arcs are directed towards the center of the star. Let $S = \{s_1, \ldots, s_q\}$ denote the subset of these stars that have at least two nodes in them. For a star $s_y \in S$, the node into which all the edges are directed is defined as the *midpoint* and denoted $c_y$.
11        For each star $s_y \in S$
12          For each node $c_z$ in $s_y$ that is not the midpoint $c_y$ of this star, add the flow path $P_z$ corresponding to the commodity for $c_z$. Note that this path joins $c_z$ to $c_y$ and has at most $\Delta$ edges.
13          Denote the union of all the paths described in the previous step along with the spanning trees of all the clusters whose centers occur in $s_y$ by $E_y$; the set of nodes in $E_y$ form the new cluster. The graph formed by the edges in $E_y$ is connected and may even be cyclic or contain multiple copies of an edge. Define the spanning tree for this new cluster as a breadth-first tree of this graph rooted at the midpoint $c_y$. This node is also designated the center of the new cluster.
14          Update $\mathcal{C}$ by removing all the clusters whose centers occur in $s_y$ and inserting the new cluster formed by merging these clusters.
15      $i := i + 1$.
16      Output the spanning tree of the single cluster in $\mathcal{C}$.
```

We now detail Step 7 by proving the following lemma.

Lemma 3 *Given a directed graph G' in which every node has exactly one outgoing arc, we can find an edge-induced subgraph H' of G' that is a forest of directed trees, where each directed tree in the forest is an inward arborescence on at least two nodes of G' and every node of G' is in exactly one of these trees.*

Proof: We prove the lemma by exhibiting a simple algorithm to construct such a forest. Starting at an arbitrary vertex, consider the directed path defined by tracing for every node, the unique arc going out of the node, until this walk meets itself, i.e., the arc out of the last node is directed to a node already in the path. Contract this path into a *supernode*. This supernode represents a collection of at least two nodes, with no edges directed out of it in G'. Also the directed path constructed defines an inward arborescence rooted at the last node in the path.

We can now repeat the above algorithm starting at another arbitrary vertex in G', the only difference being that this walk may end in a supernode. In this case, we add the nodes in the walk to the supernode and also add the directed path leading to the supernode to the arborescence for the supernode. We continue this process until no original vertex remains.

It is easy to see that the set of edges defined by the arborescences for all the remaining supernodes defines a subgraph H' obeying the conditions in the Lemma. □

The Performance Guarantees

We prove the performance guarantees using a series of lemmas. Before that, we prove a simple property of the algorithm that will be useful later. The following lemma shows that though the same node of G may appear in several clusters in the same iteration of the algorithm, the centers of these clusters are all distinct nodes of G. The proof is by a simple induction on the iteration count and is omitted.

Lemma 4 *At any iteration i in the running of the algorithm, the centers of the clusters at the beginning of this iteration are all distinct.*

Next we show that the number of clusters reduces by a constant factor at each iteration.

Lemma 5 *At any iteration i in the above algorithm, the number of clusters reduces by a factor of one-third as a result of the merges done in this iteration.*

Proof: The reduction in the number of clusters at any iteration i is exactly equal to the number of arcs in all the stars in the set S identified in this itertion by the algorithm. As a result of the bicoloring and choosing the bigger color class in Step 10, this is at least as much as half the sum of all the edges in all the trees T_x in the auxiliary subgraph H_i for this iteration. For a tree T_x on t nodes, the contribution by this tree to this sum is $\lceil \frac{t-1}{2} \rceil$. Since $t \geq 2$, this is at least $t/3$. Since any node in G_i is in some tree T_x by the property of H_i, and since $|G_i| = |\mathcal{C}|$, we have that $\sum_{T_x} \frac{|T_x|}{3}$ is at least $\frac{|\mathcal{C}|}{3}$ proving the lemma. □

Since the algorithm starts with n clusters where n is the total number of nodes in the input graph, and stops when there is one cluster, the following corollary is immediate.

Corollary 1 *The number of iterations of the above algorithm is $O(\log n)$.*

In the next lemma, we show that the diameter only increases by an additive amount of 2Δ per iteration as we merge clusters.

Lemma 6 *Let C be a cluster formed at iteration i of the algorithm. Then the diameter of the spanning tree of C maintained by the algorithm is at most $2i\Delta$.*

Proof: We prove the lemma by maintaining the following stronger claim inductively.

Claim 2 *Let C be a cluster with center c_y formed at iteration i of the algorithm. Then any node u in C has a path of length at most $i\Delta$ to c_y in the the spanning tree of C maintained by the algorithm.*

Proof of claim: The proof is by induction on the iteration count i. The basis when $i = 1$ is trivial.

To prove the induction step, consider a cluster C formed at iteration i (> 1) by merging many clusters that are grouped in a star s_y in H_i. Recall that c_y is the midpoint of the star s_y. Let C_y be the mid-cluster in the group defined by s_y. i.e., the one whose center is c_y. Consider how the spanning tree for the new cluster C is constructed in Step 13. We form the edge set E_y by unioning all the paths of length at most Δ from centers $c_z \neq c_y$ in the star s_y to the midpoint c_y and all the spanning trees for all the clusters in the group of s_y. We then choose a breadth-first tree rooted at c_y in this graph to obtain the spanning tree for the new cluster with center c_y. To prove the claim, we show that any node $u \in C$ has a path of at most $i\Delta$ edges to c_y.

Consider a node $u \in C$. The node u must be a node in either a cluster C_z in the group of s_y or in a flow path P_z. If u is in a flow path P_z, since this path has at most Δ edges, there is trivially a subpath of this path from u to c_y with at most Δ edges. If u is in the mid-cluster C_y, then by the induction hypothesis there is a path with at most $(i-1)\Delta$ edges from u to c_y in the spanning tree for C_y, and hence a path of length at most $i\Delta$ edges to c_y. Finally if u is in some cluster $C_z \neq C_y$, concatenating the path from u to c_z with at most $(i-1)\Delta$ edges in the spanning tree for C_z (by the induction hypothesis) and the flow path P_z between c_z and c_y, there is a path from u to c_y with at most $i\Delta$ edges. Thus the Claim and the Lemma are proved. □

We have the following corollary from the above lemma and Corollary 1.

Corollary 2 *The diameter of the spanning tree output by the above algorithm is $O(\Delta \log n)$.*

The following lemma bounds the degree of any node due to edges added in each iteration.

Lemma 7 *Let D^* be the minimum maximum degree of any spanning tree of the input graph with diameter at most Δ. At each iteration i of the algorithm, the maximum degree of any node due to edges added to merge clusters in this iteration is $O(D^* + \log n)$.*

Proof: To prove the above lemma, we show that at any iteration i of the algorithm, there is an integral solution to the concurrent flow problem set up in Step 4 of the algorithm of congestion at most $2D^*$. Since this also yields a fractional flow with the same value of congestion, it then follows from the performance ratio in Theorem 9 that the set of flow paths found in this iteration has congestion $O(D^* + \log n)$ at any node. But the paths chosen to merge clusters in this iteration are a subset of all the flow paths, and the maximum degree at any node due to these paths is at most the node-congestion due to the flow paths at this node, proving the lemma.

It remains to prove that there is an integral solution to the flow problem of congestion at most $2D^*$ at any iteration i. To see this, suppose that the number of center nodes in G_i is even. Consider an optimal solution T^* to the bicriteria problem, namely, a spanning tree of diameter at most Δ and maximum degree at most D^*. By Lemma 4, the centers in G_i are all distinct, so we can apply Lemma 2 to T^* with the nodes in G_i as the marked nodes. This gives a pairing of the centers such that the paths between pairs

in T^* are edge-disjoint. Consider the integral solution to the concurrent flow problem where for a pair (v_j, v_k) of centers in the pairing, we send one unit of commodity j from v_j to v_k and one unit of commodity k from v_k to v_j along the path in T^*. The length of this path is at most Δ since the diameter of T^* is at most Δ. The congestion at any node due to all such flow paths is at most $2D^*$ since each edge in the tree is used at most twice (once in each direction), the paths between pairs in T^* are disjoint, and the maximum degree of any node in T^* is D^* by definition. This provides the solution to the concurrent flow problem as desired. When there are an odd number of nodes in G_i, we can omit one node and apply the above argument and finally route flow from the omitted node to a closest node in T^* while obeying the bound on the node congestion. This completes the proof. $\square$
We have the following corollary from Corollary 1 and the above lemma.

Corollary 3 *The maximum degree of any node in the spanning tree output by the above algorithm is* $O(D^* \log n + \log^2 n)$.

Corollaries 2 and 3 together prove the performance guarantees in Theorem 4. As mentioned earlier in the remarks following Theorem 9, the flow-rounding step can be accomplished in expected $O(nm \log n)$ time and this is the dominating step in each iteration of the algorithm. The running times for the flow problems in the successive iterations reduces geometrically since the number of commodities reduces geometrically. Thus the total running time for all the flow problems over all iterations is still $O(nm \log n)$. This completes the proof of Theorem 8.

4 Extensions

4.1 Bounded-degree graphs

For the special case of graphs with bounded degree we can approximate the minimum broadcast time easily within a constant factor, the factor depending on the bound on the maximum degree. We use the simple observation below to obtain this result.

Lemma 8 *Given a spanning tree with maximum degree D and diameter Δ, there is a broadcast scheme starting from any node as the root that completes in $D\Delta$ steps.*

Any greedy broadcast scheme in which an internal node, after having received the message, communicates it to its children in any order in the subsequent steps without any delay obeys the above lemma. This is because the message to the last node to be informed travels a path of length at most Δ with the possibility of being delayed at each intermediate node for D steps as a result of queuing.

Given the above lemma and bound on the maximum degree of the graph, the problem of computing a good tree in which to accomplish the broadcast reduces to that of finding a tree of minimum diameter. This problem is exactly solvable [2, 26] in polynomial time. Observing that half the diameter of the graph is a lower bound on the broadcast time and using a minimum diameter tree to accomplish broadcast as in Lemma 8 gives Theorem 6. The results in [2, 26] on the minimum-diameter Steiner tree problem allow a direct extension of Theorem 6 to the corresponding multicast problem.

4.2 Bounded-diameter graphs

We now prove Theorem 7 showing that the case of bounded-diameter graphs is nearly as hard as arbitrary graphs with respect to approximability of the minimum broadcast time problem.

Suppose we are given a ρ-approximation algorithm for the minimum broadcast time problem for graphs of diameter two. Given an arbitrary input graph G with minimum broadcast time $b(G)$, we construct a graph H from G by adding a new node r with edges to all nodes in G. Clearly H has diameter two and the minimum broadcast time of H denoted $b(H)$ is at most $b(G) + 1$.

Claim 3 *Any broadcast scheme for H starting from the node r and completing in C steps can be used to accomplish broadcast in G in $2C + dia(G)$ steps where $dia(G)$ denotes the diameter of G.*

A ρ-approximation to the minimum broadcast time problem in H from r outputs a scheme taking at most $\rho \cdot b(H) \leq \rho(b(G) + 1)$ steps. Using this scheme and Claim 3, we can obtain a scheme for broadcasting in G from any node within $2\rho(b(G) + 1) + dia(G)$ steps. Since $b(G) \geq \frac{dia(G)}{2}$, this is at most $(2\rho + 2)b(G)$ steps asymptotically, giving Theorem 7.

Proof of Claim: We show that any broadcast scheme for H starting from the node r and completing in C steps can be used to accomplish broadcast from any source node s in G in $2C + dia(G)$ steps. From the source s, we first send the message to all the nodes that the root r informs directly in the scheme for H. Call such nodes *primary* nodes. Since the whole broadcast in H takes at most C steps, there are at most

C primary nodes. To send the message from s to the primary nodes, we use a breadth-first tree rooted at s pruned appropriately to contain all primary nodes and only the primary nodes as leaves. Thus this tree has at most C leaves. Furthermore, any path in this tree from s to a leaf has length at most $dia(G)$ since we chose a breadth-first tree. It is easy to now verify that any greedy scheme that broadcasts using this tree takes time at most $C + dia(G)$ to complete. The key observation is that the maximum time for the message to reach any of the leaves is at most the sum of the delays along the path from the source to this leaf. The delay due to the length of the path is at most $dia(G)$, and the queuing delays along the path sum to at most the sum of the degrees of all the nodes. This latter sum is at most C, the total number of leaves in the tree. To summarize, all the primary nodes in H can be informed in G starting from any source s within $C + dia(G)$ steps. Emulating the scheme for H in G for the next C steps ensures that all nodes in G are informed within at most $2C + dia(G)$ steps in total. This completes the proof of Theorem 7.

5 Open Problems

We have provided approximation algorithms for a variety of problems involving message dissemination in a network in the minimum number of steps under a telephone model. Finding an approximation algorithm for minimum broadcast time with a better performance ratio is an important open problem. The broadcast problem for digraphs modeling networks with unidirectional links does not seem amenable to our techniques. A good definition of a fault-tolerant broadcast scheme for a given network (as opposed to building fault-tolerant networks supporting rapid broadcasting) and an approximation algorithm that computes such a scheme would be useful. A distributed algorithm for broadcasting in approximately minimum time would also be desirable.

Acknowledgements

I have greatly benefited in the course of this work as a result of stimulating discussions with Ajit Agrawal, Philip Klein, Madhav Marathe, and Balaji Raghavachari. Thanks to Balaji also for reading over an early version of this article and pointing out an error and a correction leading to Theorem 7.

References

[1] A. Bagchi, S. L. Hakimi, J. Mitchem, and E. Schmeichel, "Parallel algorithms for gossiping by mail," *Inf. Proc. Lett.* 34, pp. 197-202 (1990).

[2] P. M. Camerini, G. Galbiati, and F. Maffioli, "Complexity of spanning tree problems: Part 1," *Euro. J. of O. R.* 5, (1980), pp. 346-352.

[3] S. C. Chau and A. L. Leistman, "Constructing fault-tolerant minimal broadcast networks," *J. Combin. Info. & Syst. Sci.*, Vol. 11, No. 1m pp. 1-18 (1986).

[4] M. J. Dinneen, M. R.. Fellows, and V. Faber, "Algebraic construction of efficient broadcast networks," *Applied Algebra, Algebraic Algorithms, and Error Correcting Codes 9*, LNCS 539, pp. 152-158 (1991).

[5] A. M. Farley, "Minimum-time line broadcast networks," *Networks*, Vol. 10, pp. 59-70, (1980).

[6] A. M. Farley, "Broadcast time in communication networks," *SIAM J. Appl. Math.* 39, pp. 385-390 (1980).

[7] A. M. Farley and S. T. Hedetniemi, "Broadcasting in grid graphs," *Proc. 9th Southeastern Conf. Combin., Graph Theory, Comput.*, pp. 275-288, (1978)

[8] A. M. Farley and A. Proskurowski, "Broadcasting in trees with multiple originators," *SIAM. J. Alg. Disc. Meth.*, Vol. 2, pp. 381-386, (1981).

[9] U. Feige, D. Peleg, P. Raghavan, and E. Upfal, "Randomized broadcast in networks," *Proceedings of the 1990 SIGAL Symposium on Algorithms*, Springer-Verlag LNCS 450.

[10] M. R. Garey and D. S. Johnson, *Computers and Intractability: A guide to the theory of NP-completeness*, W. H. Freeman, San Francisco (1979).

[11] L. Gargano and U. Vaccaro, "On the construction of minimal broadcast networks," *Networks* 19, pp. 673-389 (1989).

[12] M. Grigni and D. Peleg, "Tight bounds on minimum broadcast networks," *SIAM J. on Disc. Math.*, May 1991, pp. 207-222.

[13] S. M. Hedetniemi, S. T. Hedetniemi, and A. L. Leistman, "A survey of gossiping and broadcasting in communication networks," *Networks*, Vol. 18, pp. 319-349, (1988).

[14] P. N. Klein, and R. Ravi, "A nearly best-possible approximation for node-weighted Steiner trees," *Proceedings of the third MPS conference on Integer Programming and Combinatorial Optimization Conference* (1993), pp. 323-332.

[15] P. Klein, C. Stein, and É. Tardos, "Leighton-Rao might be practical: Faster approximation algorithms for concurrent flow with uniform capacities," *Proc. 22nd ACM Symposium on the Theory of Computing*, pp. 310-321, (1990).

[16] G. Kortsarz and D. Peleg, "Approximation algorithm for minimum time broadcast," *Proceedings of the 1992 Israel Symp. on Theor. Comp. Sci.*, Springer-Verlag LNCS 601.

[17] D. W. Krumme, G. Cybenko, and K. N. Venkataraman, "Gossiping in minimal time," *SIAM J. on Computing*, Vol. 21, No. 1, pp. 111-139, (1992).

[18] F. T. Leighton and S. Rao, "An approximate max-flow min-cut theorem for uniform multicommodity flow problems with application to approximation algorithms," *Proceedings of the 29th Symposium on Foundations of Computer Science* (1988), pp. 422-431.

[19] A. L. Leistman, "Fault-tolerant broadcast graphs," *Networks*, Vol. 15, pp. 159-171, (1985).

[20] A. L. Leistman and J. G. Peters, "Broadcast networks of bounded degree," *SIAM J. Disc. Math.*, Vol. 1, pp. 531-540, (1988).

[21] M. Middendorf, "Minimum Broadcast Time is NP-complete for 3-regular planar graphs and deadline 2," *Inf. Proc. Lett.* 46, pp. 281-287 (1993).

[22] P. Raghavan, "Probabilistic construction of deterministic algorithms: approximating packing integer programs," *Proceedings, 27th Annual Symposium on Foundations of Computer Science* (1986), pp. 10-18.

[23] P. Raghavan and C.D. Thompson, "Provably good routing in graphs: regular arrays," in *Proceedings, 17th Annual Symposium on Theory of Computing* (1985), pp. 79-87.

[24] R. Ravi, "An approximation algorithm for the minimum broadcast time problem," Technical Report TR-CS-94-01, Dept. of Computer Science, University of California, Davis, (January 1994).

[25] R. Ravi, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, and H.B. Hunt III, "Many birds with one stone: Multi-objective approximation algorithms," *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing* (1993), pp. 438-447.

[26] R. Ravi, R. Sundaram, M. V. Marathe, S. S. Ravi, and D. J. Rosenkrantz, "Spanning trees short or small," in *Proceedings, Fifth Annual ACM-SIAM Symposium on Discrete Algorithms,* (1994), pp. 546-555.

[27] D. Richards and A. L. Leistman, "Generalizations of broadcasting and gossiping," *Networks*, Vol. 18, pp. 125-138, (1988).

[28] F. Shahrokhi and D. W. Matula. "The maximum concurrent flow problem," *Journal of the ACM*, 37 (1990), pp. 318-334.

[29] P. Scheuermann and G. Wu, "Heuristic algorithms for broadcasting in point-to-point computer networks," *IEEE Trans. on Computers*, Vol,. 33, pp. 804-811, (1984).

[30] P. J. Slater, E. J. Cockayne, and S. T. Hedetniemi, "Information dissemination in trees," *SIAM J. on Computing*, Vol. 10, pp. 692-701, (1981).

[31] J. A. Ventura and X. Weng, "A new method for constructing minimal broadcast networks," *Networks*, 23, pp. 481-497 (1993).

[32] D. B. West, "A class of solutions to the gossip problem, Part I," *Disc. Math.*, Vol. 39, pp. 307-326, (1982).