Tight Bounds on Minimum Broadcast Networks

Michelangelo Grigni*
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139

David Peleg†
Department of Applied Mathematics
The Weizmann Institute
Rehovot 76100, Israel

October 1988
(revised May 1990)

Abstract

A *broadcast graph* is an n-vertex communication network that supports a broadcast from any one vertex to all other vertices in optimal time $\lceil \lg n \rceil$, given that each message transmission takes one time unit and a vertex participates in at most one transmission per time step. This paper establishes tight bounds for $B(n)$, the minimum number of edges of a broadcast graph, and $D(n)$, the minimum maxdegree of a broadcast graph. Let $L(n)$ denote the number of consecutive leading 1's in the binary representation of integer $n-1$. We show $B(n) = \Theta(L(n) \cdot n)$ and $D(n) = \Theta(\lg \lg n + L(n))$, and for every n we give a construction simultaneously within a constant factor of both lower bounds. For all n we also construct graphs with $O(n)$ edges and $O(\lg \lg n)$ maxdegree requiring at most $\lceil \lg n \rceil + 1$ time units to broadcast. Our broadcast protocols may be implemented with local control and $O(\lg \lg n)$ bits overhead per message.

*This material is based upon work supported under a National Science Foundation Graduate Fellowship. Work done partly at AT&T Bell Labs, Murray Hill, NJ 07974.

†Part of this work was carried out while this author was visiting Stanford university. Supported in part by a Weizmann fellowship, by contracts ONR N00014-85-C-0731 and ONR N00014-88-K-0166 and by a grant of Stanford Center for Integrated Systems.

1 Introduction

This paper deals with graphs suitable for performing broadcasts efficiently. We represent a communication network by a connected graph G, where the vertices of G represent processors and the edges represent bidirectional communication channels. We assume communication has the following constraints:

1. messages may be sent directly only between neighbors in the graph,
2. each message transmission takes one unit of time,
3. a vertex may participate in at most one message transfer at a time.

That is, if u sends a message to v, neither u nor v may send or receive another message on that step. A *broadcast protocol* for G allows any *originator* vertex to send a message to all other vertices in the network. This broadcast model is studied in several papers [BHLP1, BHLP2, CL1, CL2, F1, F2, FH, FP, FHMP, HHL, HL, L, MH, P, RL, SW, SCH, Wa].

Given G and vertex $v \in G$ let $b(v, G)$ be the minimum time needed to broadcast from v. Let $b(G) = \max_v b(v, G)$, the *broadcast radius* of G. Since the number of vertices knowing the message may at most double on each step, $b(G) \geq \lceil \lg n \rceil$ for any n-vertex graph G (lg denotes $\log_2$). A *broadcast graph* is an n-vertex graph G with $b(G) = \lceil \lg n \rceil$.

We consider three cost measures for broadcast graphs and their protocols. The first, which is often the most significant cost measure in network design, is the number of edges. Let $B(n)$ denote the minimum number of edges of any n-vertex broadcast graph. A *minimum broadcast graph* is a broadcast graph with $B(n)$ edges; a number of previous papers have dealt with determining values of $B(n)$ and finding minimum or near-minimum broadcast graphs. The values of $B(n)$ were determined precisely for $n \leq 18$ [FHMP, MH, Wa]. For general n it was shown that $B(n) = O(n \lg n)$ [F1] and that $B(n) = \Omega(n)$ (more precisely, $n - 1$ is a stated lower bound in [F1, L], and $B(n) \geq n$ for $n > 3$ is implied by the discussion in [F2]). For n a power of two, $B(n) = \frac{1}{2} n \lg n$ [FHMP], realizable by the hypercube graph. However, for n not a power of 2 the behavior of $B(n)$ was not precisely determined.

The second cost measure we consider is the maximum degree of broadcast graphs. This measure is not as well studied as the previous one in the context of broadcast graphs, but is no less important due to current limitations in networking technology. For vertex $v \in G$ let $d(v)$ denote its degree, so the maxdegree of G is

$\Delta(G) = \max_v d(v)$. Let $D(n)$ be the minimum maxdegree of any n-vertex broadcast graph. Several previous papers concentrated on broadcasting on bounded-degree graphs [BHLP1, LP].

The final cost measure we consider is the message overhead needed to implement the broadcast protocol under local control. We assume the broadcast messages may carry along extra control bits, and we bound the maximum number of extra bits needed on any message sent in the protocol. We assume that processors know the size of the graph and their own identity in the graph, as well as local information such as the identities of their neighbors. We assume also that processors know on which edge an incoming message arrives; in some situations this is all the information the processor needs. We use a synchronous model where all messages take unit time, although we do not assume processors have access to a global clock.

For an integer $n > 1$ let $L(n)$ denote the number of leading 1's in the binary representation of $n-1$; for example $L(14) = L(1101_2 + 1) = 2$. Then $1 \leq L(n) \leq \lceil \lg n \rceil$. $L(n)$ is monotone increasing in the range $2^{t-1} < n \leq 2^t$ for any $t \geq 1$. For n in such range we have $L(n) = t - \lceil \lg(2^t - n + 1) \rceil$. Note that $L(n)$ grows slowly in this interval; in particular, it equals 1 over the first half of the interval $(2^{t-1} < n \leq 2^{t-1} + 2^{t-2})$, 2 over the next quarter of the interval and so on. More generally, for all $n, l \geq 1$,

$$\frac{|\{i : 1 < i \leq n,\ L(i) > l\}|}{n} < 2^{-l},$$

so $L(n)$ is bounded by a constant for 'most' values of n.

We show $B(n) = \Theta(L(n) \cdot n)$ and $D(n) = \Theta(\lg \lg n + L(n))$, and we construct graphs meeting both bounds simultaneously. Since $L(n) = o(\lg \lg n)$ for most n, this implies that most minimum broadcast graphs must be irregular, since they have $O(L(n))$ (constant) average degree but $\Omega(\lg \lg n)$ maxdegree.

Furthermore, we give protocols which may be implemented with $O(\lg \lg n)$ bit overhead per message in the synchronous model. In the asynchronous model, where there is no guarantee on message transmission time, the same graphs need $O(\lg n)$ bits overhead per message to avoid message collisions. These asynchronous protocols are tree-shaped: there are exactly $n-1$ messages sent, one to each processor besides the originator.

In view of the practical significance of keeping $B(n)$ and $D(n)$ as small as possible, it may sometimes be desirable to allow a slight increase in broadcast time in order to allow a decrease in these cost parameters. This has led to the following relaxation of the problem [F1, L]. A *relaxed broadcast graph* G has $b(G) \leq \lceil \lg n \rceil + 1$. Let $B'(n)$ and

$D'(n)$ denote the minimum number of edges and maxdegree required for an n-vertex relaxed broadcast graph. In [F1] it is noted that $B'(n)$ may be significantly less than $B(n)$ when n is equal to or slightly less than a power of 2. They demonstrate this fact by considering $n = 16$ (where the minimum time requirement is 4 steps while the relaxed requirement is 5 steps) for which $B(n) = 32$ and $B'(n) = 19$. We construct relaxed broadcast graphs with $O(n)$ edges and $O(\lg \lg n)$ maxdegree, both within a constant factor of optimal. (Again *a priori* these must be irregular graphs.)

Although we have made our definitions for undirected graphs, our constructions use directed graphs, where messages may only travel in the direction of the edge. This leads to the analogous definitions of broadcast digraphs, their minimum edge number $\vec{B}(n)$, and their minimum relaxed edge number $\vec{B}'(n)$. Clearly $B(n) \leq \vec{B}(n) \leq 2 \cdot B(n)$ and $B'(n) \leq \vec{B}'(n) \leq 2 \cdot B'(n)$, so edge counting results in either model are equivalent up to a factor of 2. In the directed model we will let $d_{\text{out}}(v)$ and Δ_{out} refer to outdegree while $d_{\text{in}}(v)$ and Δ_{in} refer to indegree. Let $d = d_{\text{out}} + d_{\text{in}}$ and $\Delta(G) = \max_v d(v)$, then $\vec{D}(n)$ is the minimum of $\Delta(G)$ over all n-vertex broadcast digraphs G, $D(n) \leq \vec{D}(n) \leq 2 \cdot D(n)$. Similarly define $\vec{D}'(n)$ as the minimum maxdegree of relaxed broadcast digraphs.

The paper is organized as follows. In section 2 we derive lower bounds. In sections 3 and 4 we construct preliminary graphs, serving as building blocks for our main constructions given in section 5. Section 6 discusses a generalization of the model allowing "conference calls". Finally in section 7 we offer some related problems and open questions.

2 Broadcast Tree Lower Bounds

For a vertex v in graph (or digraph) G, a *broadcast tree* T is a time-labeled directed subgraph describing a broadcast originated by v, by the following rules:

1. T is spanning in G rooted at v, directed toward the leaves.
2. Each vertex u is labeled with an integer $t(u)$, where $t(v) = 0$.
3. Whenever u is a parent of w in T, $t(u) < t(w)$.
4. Whenever u and w are siblings in T, $t(u) \neq t(w)$.

Given such a T, interpret label $t(u)$ as the step when u receives the message originated by v; the conditions guarantee that the parent of u has the message and is free to

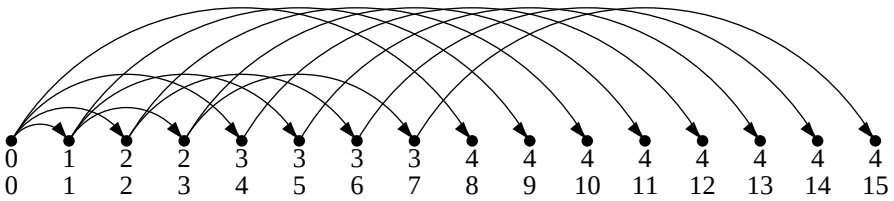

Figure 1: The boolean broadcast tree T_4 describing a 4-step broadcast from 0 in the hypercube H_4. The first row of numbers is the time labeling $t(\cdot)$; the second row is the usual binary numbering.

send it to u on step $t(u)$. Define $t(T) = \max_u t(u)$; we say T is a k-step broadcast tree when $k = t(T)$. A collection of such trees, one rooted at each $v \in G$, defines a broadcast protocol for G.

For example, let H_r be the r-dimensional hypercube, with vertices given the usual binary numbering $0, \ldots, 2^r - 1$. Broadcast from 0 by sending along the sth dimension on step s, for $1 \leq s \leq r$: each vertex v that knows the message sends it to $v+2^{s-1}$. The edges used by this protocol define the *boolean broadcast tree* T_r [F1]. (See Figure 1.)

Not every broadcast protocol is described by a broadcast tree, since a protocol may send more than one message to some vertex. Nevertheless, given a k-step broadcast from v, there exists a k-step broadcast tree from v, consisting of those edges on which each vertex first receives the message. Hence $b(v, G) \leq k$ iff there is a k-step broadcast tree T rooted at v. Any such tree protocol may be controlled with at most $O(\lg n)$ bits overhead per message (the identity of the originator), although the local program length may be long. A parent in T might as well tell all its children the message as quickly as possible, so we may also require the additional rule:

5. The children of any $u \in T$ have consecutive labels $t(u) + 1$, $t(u) + 2$, ...

Refer to those vertices v in T with $t(v) = s$ as *generation* s of T. If we don't require that T span G, then say T is a *partial* broadcast tree in G, i.e. it only broadcasts to those vertices that it spans.

Lemma 2.1 *In a (partial) broadcast tree T, the subtree rooted at a vertex u has size at most $2^{t(T)-t(u)}$.*

Proof: The subtree can at most double on each step after u gets the message. ∎

Theorem 2.2 *Let G be an n-vertex broadcast graph. Then every vertex $v \in G$ has degree $d(v) \geq L(n)$.*

Proof: Let $k = \lceil \lg n \rceil$, let T be a k-step broadcast tree from v, let $\delta \leq d(v)$ be the degree of v in T, and let $v_1, \ldots, v_\delta$ be the children of v in T, labeled $t(v_s) = s$. Then the subtree rooted at v_s has size at most 2^{k-s}. Since these subtrees contain all vertices except v, $n - 1 \leq \sum_{i=1}^{\delta} 2^{k-i} = 2^k(1 - 2^{-\delta})$, so $\delta \geq k - \lg(2^k - (n-1)) \geq L(n)$. ∎

Note the last inequality is not tight; for example when $n = 14$ we have $L(n) = 2$ but the proof really shows that the degree is at least 3. For directed G the same argument shows $d_{\text{out}}(v) \geq L(n)$ for all v; by averaging there also must be a vertex v_0 with $d_{\text{in}}(v_0) \geq L(n)$, hence $d(v_0) \geq 2 \cdot L(n)$.

Corollary 2.3 *For all $n \geq 1$, we have $B(n) \geq \frac{1}{2}L(n) \cdot n$, $\vec{B}(n) \geq L(n) \cdot n$, $D(n) \geq L(n)$, and $\vec{D}(n) \geq 2 \cdot L(n)$.*

Constructions in section 5 show the above bounds on $B(n)$ and $\vec{B}(n)$ are tight up to a constant factor. We need a further argument to get tight lower bounds for $D(n)$ and $\vec{D}(n)$.

For a given outdegree bound d and time bound t, we inductively construct $T_{d,t}$, the largest broadcast tree with $t(T) \leq t$ and $\Delta_{\text{out}}(T) \leq d$. The root v should have as many children as possible, so $d_{\text{out}}(v) = \min(d, t)$. Each child u of v must be the root of a maximum size subtree, so the tree rooted at u must be $T_{d,t-t(u)}$; this recursive construction uniquely defines $T_{d,t}$.

Let $b_d(t)$ be the number of vertices in $T_{d,t}$, and let $f_d(s)$ be the size of generation s in $T_{d,t}$, so $b_d(t) = \sum_{s=0}^{t} f_d(s)$. Since the parents of generation s are the vertices of the previous d generations, we have recurrences for $s, t > 0$:

$$b_d(t) = 1 + \sum_{1 \leq i \leq d} b_d(t-i), \tag{1}$$

$$f_d(s) = \sum_{1 \leq i \leq d} f_d(s-i) \tag{2}$$

where $b_d(0) = f_d(0) = 1$ for the originator and $b_d(s) = f_d(s) = 0$ for $s < 0$. The recurrence (2) defines the dth-order Fibonacci sequence [K, 5.4.2]. If $d \geq t$ then the tree $T_{d,t}$ is simply the boolean broadcast tree T_t.

The generating polynomial $x^d = x^{d-1} + \cdots + x + 1$ has one large real root λ near 2 dominating the growth rate of $f_d(t)$ and $b_d(t)$ (all other roots lie in the unit circle):

$$\lambda \sim 2 - 2^{-d} - \frac{d}{2}2^{-2d} - O(d^2 2^{-3d}) \text{ as } d \to \infty.$$

For our purposes it suffices that $2 - 2^{1-d} < \lambda < 2 - 2^{-d}$ for all $d \geq 2$.

d	λ	$\frac{(2/\lambda)^{d-1}}{\lambda-1}$	$f_d(0)$	$f_d(1)$	$f_d(2)$	$f_d(3)$	$f_d(4)$	$f_d(5)$	$f_d(6)$
2	1.6180	2.0000	1	1	2	3	5	8	13
3	1.8393	1.4088	1	1	2	4	7	13	24
4	1.9276	1.2043	1	1	2	4	8	15	29
5	1.9659	1.1089	1	1	2	4	8	16	31
6	1.9836	1.0595	1	1	2	4	8	16	32

Table 1: λ, $(2/\lambda)^{d-1}/(\lambda-1)$, and $f_d(t)$ for $2 \le d \le 6$, $0 \le t \le 6$.

Lemma 2.4 *For $t \ge 1, d \ge 2$, let λ be defined as above. Then*

$$\begin{aligned} \lambda^{t-1} &\le f_d(t) \le (2/\lambda)^{d-1}\lambda^{t-1}, \\ \tfrac{\lambda^t-1}{\lambda-1}+1 &\le b_d(t) \le (2/\lambda)^{d-1}\tfrac{\lambda^t-1}{\lambda-1}+1. \end{aligned}$$

Proof: Fix d. For $1 \le t \le d$ we have $f_d(t) = 2^{t-1}$, which lies in the claimed range. Now for $t > d$ use induction on t and the fact that λ satisfies the generating polynomial. The bounds on $b_d(t)$ follow from summing the bounds on $f_d(t)$. ∎

Theorem 2.5 *Let T be an n-vertex k-step broadcast tree with $d = \Delta_{\text{out}}(T)$. Then*

$$d > \lg\left(\frac{k \lg e}{k+1-\lg n}\right) - 1.$$

Proof: Since $T_{d,k}$ is the largest possible such tree, $n \le b_d(k)$. Estimate $(2/\lambda)^{d-1}/(\lambda-1) \le 2$ (see table 1), so $n \le b_d(k) \le 2(\lambda^k-1)+1 < 2\lambda^k$, so $\lg n < 1 + k\lg\lambda$. Now estimate $\lg\lambda < 1 - 2^{-(d+1)}\lg e$ and solve for d. ∎

In particular we consider trees arising in broadcast graphs and relaxed broadcast graphs:

Corollary 2.6 *Let T be an n-vertex broadcast tree and $t(T) \le c + \lg n$. Then $\Delta_{\text{out}}(T) > \lg\lg n - 0.5 - \lg(c+1)$. In particular if $t(T) \le \lceil \lg n \rceil$ then $\Delta_{\text{out}}(T) > \lg\lg n - 1.5$, and if $t(T) \le \lceil \lg n \rceil + 1$ then $\Delta_{\text{out}}(T) > \lg\lg n - 2.1$.*

Corollary 2.7 *$D(n)$ and $\vec{D}(n)$ are $\Omega(L(n) + \lg\lg n)$; $D'(n)$ and $\vec{D}'(n)$ are $\Omega(\lg\lg n)$.*

We have shown that any n-vertex $\lceil \lg n \rceil$-step broadcast tree T has $\Delta(T) \ge \max(L(n), \lg\lg n - 1.5)$. We now show this lower bound is tight up to a leading factor of 2 and a small additive constant.

Lemma 2.8 *Given d, l, t with $d \ge l + \lg t$, $b_d(t) > (1 - 2^{-l})2^t$.*

Proof: Estimate $b_d(t) \geq \lambda^t > 2^t(1-2^{-d})^t > 2^t(1-t2^{-d}) = 2^t - t2^{t-d} \geq 2^t - 2^{t-l}$ by the condition on d. ∎

Corollary 2.9 *For $n \leq 2^t$, let $d = L(n) + \lceil \lg \lg n \rceil + 1$. Then $n \leq b_d(t)$.*

3 Boolean Constructions

This section and the following one describe some initial constructions, which will be combined in Section 5 to yield the desired results. Specifically, this section concerns constructions based on variations of the hypercube.

Given n and a $S \subset \mathbb{Z}_n = \mathbb{Z}/n\mathbb{Z}$, the difference digraph $\mathbb{Z}_n[S]$ is defined with vertex set $\mathbb{Z}_n$ and edge set $\{i \to i+s : i \in \mathbb{Z}_n, s \in S\}$ (these are also known as directed star polygons). $\mathbb{Z}_n[S]$ has $|S|n$ edges. For example define the *boolean difference digraph* as $BD(n) = \mathbb{Z}_n[\{2^i : 0 \leq i < \lceil \lg n \rceil\}]$. $BD(n)$ has broadcast properties similar to the hypercube, but it is defined even for n not a power of two.

Theorem 3.1 *$BD(n)$ is a broadcast digraph, with a 1-bit overhead protocol.*

Proof: By translational symmetry we may assume 0 is the originator. Let $k = \lceil \lg n \rceil$. On step s, $1 \leq s \leq k$, every vertex i which knows the message and knows $i + 2^{k-s} < n$ sends the message to $i + 2^{k-s}$ (note we have reversed the bit order used in the H_k protocol described at the beginning of the previous section). This protocol will reach every processor exactly once.

Note the processors know the time step by observing on which edge the message arrives (so this works asynchronously as well). To decide whether $i + 2^{k-s} < n$, the processors pass along an extra bit. When processor i sends a message to $j = i + 2^{s-1}$, the extra yes/no bit tells j whether $j + 2^{s-1} \geq n$. If 'no,' then j knows that its subtree will not be truncated anywhere, and so sends 'no' bits to all its children. Otherwise, j computes $n' = n \bmod 2^{s-1}$ and recursively originates the $BD(n')$ protocol (this recursion adds no overhead to the messages, we only require that each processor knows the value of n). An originator knows n and should send a 'yes' message to the first child such that $2^{s-1} < n$, and a 'no' message to every child after that.

We observe the resulting broadcast is tree-shaped, and hence this protocol will also work in the asynchronous model. ∎

Let H_r denote the directed r-dimensional hypercube (i.e. H_r has a pair of directed edges wherever the undirected hypercube has an edge). From H_r we construct a

related digraph $H_{r,t}$ with 2^{r+t} vertices: at each $v \in H_r$ root a copy of the boolean broadcast tree T_t. Refer to the original H_r vertices as 'root' vertices and the new $2^r(2^t-1)$ vertices as 'tree' vertices. Any root of $H_{r,t}$ may originate an $(r+t)$-step broadcast: for the first t steps broadcast across H_r to all the roots, and then for the remaining t steps broadcast up all the trees. This protocol requires no overhead bits since the dimensions are always used in a fixed order. Now modify $H_{r,t}$ by adding a back-edge from every tree vertex back to the root of its tree; call the resulting digraph $H'_{r,t}$. Then $H'_{r,t}$ is a relaxed broadcast digraph. Root vertices originate a broadcast as before; tree vertices take one step to notify their root, and then let the root take care of the broadcast from there. In this case the protocol does not trace out a tree of messages, since the originator will receive a copy of the message; nevertheless the protocol is still valid in the asynchronous model because the first message of the originator cannot collide with any future messages.

Just using $H'_{r,t}$ we may construct a sparse ($O(n)$-edge) relaxed broadcast digraph. Given n let $k = \lceil \lg n \rceil$, $t = \lceil \lg k \rceil$, and $r = k - t$. Then $H'_{r,t}$ has 2^k vertices; throw out $2^k - n$ leaves (this will not disrupt the protocol). The resulting digraph has n vertices, $(r-2)2^r + 2n < 3n$ directed edges and maxdegree $\Delta = 2r + t + 2^t - 1 < 4k = O(\lg n)$.

4 Fibonacci Constructions

In this section we construct partial broadcast digraphs *FIB1*, *FIB2*, *FIB3*; all rely on one idea, an "addressing" scheme based on the generalized Fibonacci numbers of section 2. Construction *FIB1* is the simplest illustration of the idea. Construction *FIB2* takes care of some wraparound problems, allowing any node to be an originator. Construction *FIB3* allows the originator to send fewer messages; this graph will be the "backbone" for the final broadcast graph constructions in section 5. These constructions have parameters d, t, and l (corresponding roughly to maxdegree, broadcast time, and $L(n)$).

For string $\alpha = \alpha_1 \cdots \alpha_t \in \{0,1\}^t$, let $\langle \alpha \rangle_d$ denote $\sum_{i=1}^{t} a_i \cdot f_d(i)$. Let $\mathcal{B}_{d,t} \subset \{0,1\}^t$ denote the strings which do not have the substring 0^d1, and let $\mathcal{F}_{d,t} \subset \mathcal{B}_{d,t}$ be those with $\alpha_t = 1$ ($\mathcal{F}_{d,0}$ and $\mathcal{B}_{d,0}$ both contain the empty string). We have the following numbering theorem; it is a 'dense' version of the dth-order Fibonacci number system [K, exercise 5.4.2.10]:

Lemma 4.1 *For $d \geq 2$, $|\mathcal{F}_{d,t}| = f_d(t)$, $|\mathcal{B}_{d,t}| = b_d(t)$, and the map $\alpha \mapsto \langle \alpha \rangle_d$ from $\mathcal{B}_{d,t}$ to $\{0, \ldots, b_d(t) - 1\}$ is bijective.*

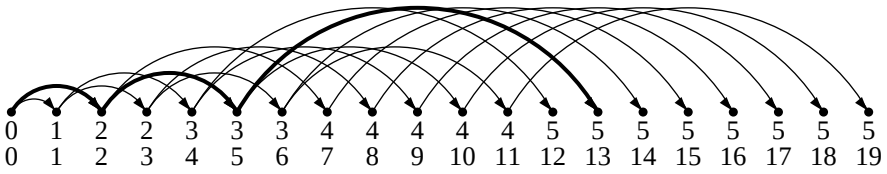

Figure 2: Broadcast tree $T_{2,5}$ with generation labels and the numbering of lemma 4.1. Bold edges show $13 = 2 + 3 + 8 = \langle 01101 \rangle_2$.

Proof: Give each vertex u of $T_{d,t}$ a t-bit address α corresponding to the path from the root to u, where $\alpha_s = 1$ iff the path includes a vertex of generation $s \geq 1$ (see figure 2). Inductively the addresses in generation s are $\alpha = \beta 0^{t-s}$ where $\beta \in \mathcal{F}_{d,s}$, $b_d(s-1) \leq \langle \alpha \rangle_d < b_d(s)$. Finally note $\mathcal{B}_{d,t}$ is the disjoint union $\mathcal{B}_{d,t} = \bigcup_{s=0}^{t} \mathcal{F}_{d,s} 0^{t-s}$. ∎

Given $d \leq t$, we construct a digraph $FIB1_{d,t}$ (see figure 3) based on this addressing scheme. $FIB1_{d,t}$ has vertex set $\mathbb{Z}_{b_d(t)} \times \mathbb{Z}_t$; we let $x \in \mathbb{Z}_{b_d(t)}$ index columns and $s \in \mathbb{Z}_t$ index rows. It has the following single class of edges (we start a list of classes here because we will add more soon):

Class 1: For all x, $1 \leq s \leq t$, and $1 \leq i \leq \min(d, s)$, connect $(x - f_d(s), s - i)$ to (x, s).

$FIB1_{d,t}$ has $b_d(t) \cdot t$ vertices, less than $b_d(t) \cdot dt$ directed edges, and maxdegree $2d$.

Theorem 4.2 *For every vertex $(x, 0) \in FIB1_{d,t}$ there is a t-step partial broadcast tree $T1_{d,t}(x)$ rooted at $(x, 0)$ such that:*

(i) $T1_{d,t}(x)$ contains exactly one vertex in each column y.

(ii) Generation s of $T1_{d,t}(x)$ lies entirely in row s.

(iii) $T1_{d,t}(x)$ is isomorphic to $T_{d,t}$.

(iv) The protocol for $T1_{d,t}(x)$ needs no bit overhead on messages.

Proof: Let $T1_{d,t}(x)$ correspond to the following protocol: on step s, each vertex (y, r) that has received the message within the last d steps ($s - d \leq r < s$) sends it to $(y + f_d(s), s)$ (see figure 3). Then we are simply reconstructing $T_{d,t}$: vertex u in generation s of $T_{d,t}$ is reconstructed as $(s, x + \langle \alpha \rangle_d)$ in $T1_{d,t}(x)$ where α is the address of u. Hence we generate all $\alpha \in \mathcal{B}_{d,t}$, and touch every column exactly once. Note the vertices don't need to know any dynamic information such as α or x to run

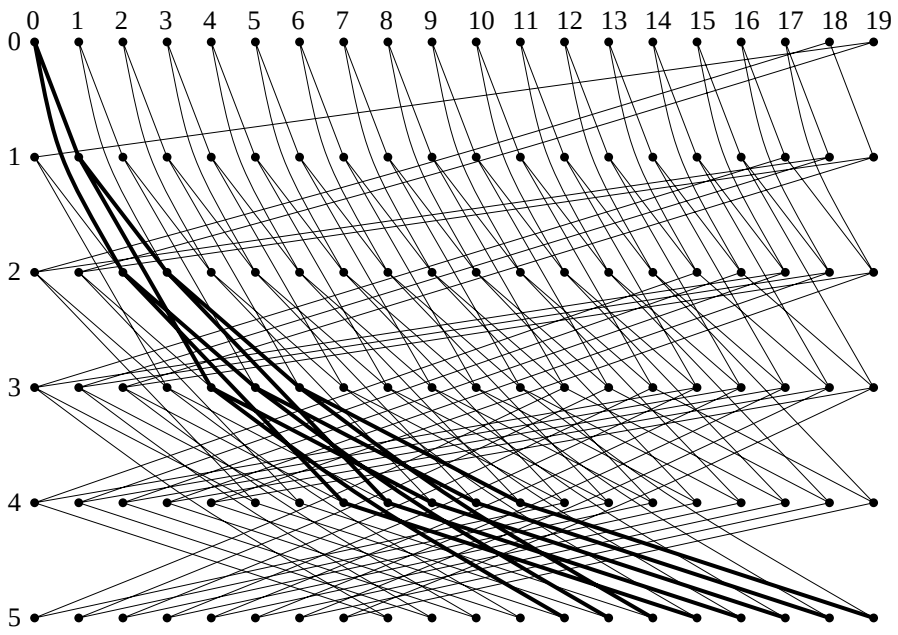

Figure 3: The digraph $FIB1_{2,5}$ with subtree $T1_{2,5}(0)$. All edges are directed downward; the first and last rows are identified.

this protocol (the vertices may even be oblivious to which edge delivers an incoming message). ∎

$FIB1_{d,t}$ is still a long way from a broadcast digraph; our next construction allows *any* vertex to originate a partial broadcast, with only slightly higher costs. Construct a second digraph $FIB2_{d,t}$ by augmenting $FIB1_{d,t}$ with three further edge classes (figure 4 illustrates how the four classes connect the rows of $FIB2_{d,t}$):

Class 2: For all x, $t-d \le r < t$, $1 \le s \le d$, connect $(x - f_d(s), r)$ to (x, s).

Class 3: For all x and $1 \le s \le t-1$, add an edge from (x, s) to $(x, s+1)$.

Class 4: For all x and $1 \le s < t-d$, add an edge from (x, s) to $(x, s+d+1)$.

We refer to edges of classes 1 and 2 as 'Fibonacci' edges because sending a message along such an edge always involves a jump of $f_d(s)$ columns, where $s \in \{1, \ldots, t\}$ is the row in which the jump ends. We refer to edges of classes 3 and 4 as 'Zero' edges since they begin and end in the same column.

Theorem 4.3 *For every vertex $(x, r) \in FIB2_{d,t}$ there is a t-step partial broadcast tree $T2_{d,t}(x, r)$ rooted at (x, r) such that:*

(i) $T2_{d,t}(x, r)$ contains at least one vertex in each column y.

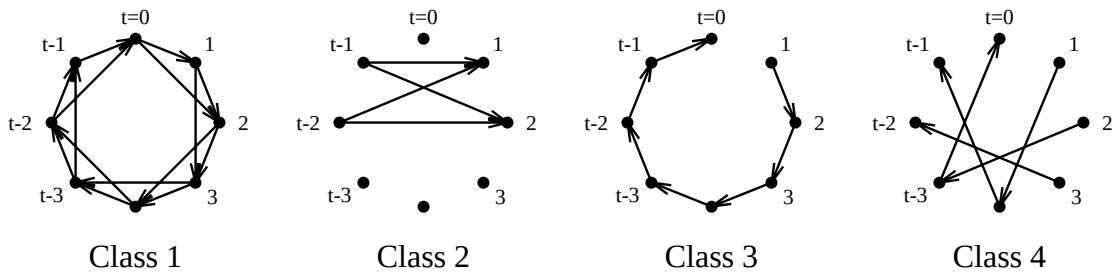

Figure 4: The classes of edges between rows of $FIB2_{2,8}$. Edges here represent collections of edges between the corresponding rows (Fibonacci edges for classes 1 and 2, Zero edges for classes 3 and 4).

(ii) *Generation s of $T2_{d,t}(x,r)$ lies entirely in row $r+s$.*

(iii) $\Delta_{\text{out}}(T2_{d,t}(x,r)) \leq 2d$.

(iv) *With $\lg t + \lg d + O(1)$ bits overhead per message we may implement this protocol and furthermore appoint a unique 'leader' vertex of $T2_{d,t}(x,r)$ in each column.*

Proof: By translational symmetry among the columns we may assume $x = 0$. If $r = 0$ we use the protocol from $FIB1_{d,t}$, otherwise $1 \leq r < t$. Again we construct the tree by describing a protocol. In the previous protocol we constructed all addresses $\alpha \in \mathcal{B}_{d,t}$ by jumping along Fibonacci edges; in this protocol we will construct all addresses $\alpha = \gamma\beta$, where $\gamma \in \mathcal{B}_{d,r}$ and $\beta \in \mathcal{B}_{d,t-r}$. In particular this includes all $\alpha \in \mathcal{B}_{d,t}$. The protocol proceeds in two phases: in phase I (the first $t-r$ steps) we construct β while leaving $\gamma = 0^r$, and in phase II (the last r steps) we wrap around and construct γ. In phase I we use Zero edges to pass completed $0^r\beta$ addresses down their columns to where they will eventually wrap around; phase II is essentially identical to the $FIB1_{d,t}$ protocol. We maintain the invariant that all messages sent on step τ arrive in row $s = \tau + r \bmod t$, so each generation is confined to a single row.

In the following discussion let (y,s) refer to a vertex receiving the message on time step τ, $1 \leq s, \tau \leq t$. We call messages 'Fibonacci' or 'Zero' depending on the type of edge they traverse. Let messages carry the following additional information fields:

- The current time step τ and the row r of the originator. Since the receiver is in row $s = r + \tau \bmod t$, only one of these fields really needs to be sent.
- A routing address $\alpha \in \{0,1\}^t$ telling the receiver (y,s) what path the message has followed so far on its way here from the originator (x,r). Bit α_i tells whether

the message made a Fibonacci jump to row i, hence $y = x + \langle\alpha\rangle_d$. We maintain $\alpha = \gamma\beta$ for some $\gamma \in \mathcal{B}_{d,r}$, $\beta \in \mathcal{B}_{d,t-r}$.

We will show later that the protocol does not need to carry along all t bits of α, but only $\lg d + O(1)$ bits, so the total message overhead will be $\lg t + \lg d + O(1)$ bits.

To start the protocol we pretend that on step $\tau = 0$ the originator (x, r) receives a Fibonacci message with address $\alpha = \gamma\beta = 0^t$. We describe the protocol by the actions of any vertex (y, s) after receiving a message on step τ. There are several cases:

I. If the received message is a phase I message (precisely, $\tau = 0$ or $r < s \le t$), the protocol is still working on β while $\gamma = 0^r$. The received message could be of either type:

Zero: If the received message is type Zero, then we are simply passing this $0^r\beta$ down the column without changing β further. There are two cases:

- If $s < t$, then just send the same Zero message on to the next row of this column, using an edge of class 3.
- If $s = t$, then phase II will begin on the next step, so the receiver (y, t) must start sending Fibonacci messages to modify γ. It sends them consecutively to rows $1, \ldots, \min(d, r)$, using edges of class 1.

Fibonacci: If the received message is type Fibonacci, we are actively building β, although γ is still 0^r. We have two goals: continue modifying β by sending Fibonacci messages to higher rows, and also let this current $\alpha = 0^r\beta$ wrap-around the column to fill in its low-order bits. There are two cases:

- If $s < t - d$, then the next $d + 1$ steps (rows) are also phase I. Send Fibonacci messages to rows $s + 1, \ldots, s + d$, using edges of class 1. Finally send a Zero message to row $s + d + 1$ using an edge of class 4.
- If $s \ge t - d$, then (y, s) must both finish phase I and start phase II. For the remaining $t - s$ phase I steps, (y, s) sends Fibonacci messages to rows $s + 1, \ldots, t$ using edges of class 1. On the next $\min(d, r)$ phase II steps (y, s) sends Fibonacci messages to rows $1, \ldots, \min(d, r)$ using edges of class 2.

II. In phase II we fill in γ; only Fibonacci messages may be received in this phase. (y, s) reacts by sending Fibonacci messages to rows $s + 1, s + 2, \ldots$ for the next

$\min(d, t-\tau = r-s)$ steps using edges of class 1. This is essentially the protocol of $FIB1_{d,t}$.

Recall that all messages sent on step τ arrive in row $s = r+\tau \bmod t$. The following claims characterizing the messages sent on each step may be proven inductively:

I. On step τ in phase I there is one Fibonacci message sent for each $\alpha = 0^r\beta 0^{t-s}$ where $\beta \in \mathcal{F}_{d,\tau}$. There is one Zero message sent for each $\alpha = 0^r\beta 0^{(d+1)+t-s}$ where $\beta \in \mathcal{B}_{d,\tau-(d+1)}$.

II. On step τ in phase II only Fibonacci messages are sent, one for each $\alpha = \gamma 0^{r-s}\beta$ where $\gamma \in \mathcal{F}_{d,s}$ and $\beta \in \mathcal{B}_{d,t-r}$.

Every $\alpha \in \mathcal{B}_{d,r}\mathcal{B}_{d,t-r}$ appears exactly once as the address of a Fibonacci message (an α may appear several times as a Zero message address), so in particular each $\alpha \in \mathcal{B}_{d,t}$ appears exactly once as the address of a Fibonacci message (we count the dummy message sent to the originator). Then there is a natural 'leader' in each column: the vertex receiving an address $\alpha \in \mathcal{B}_{d,t}$ in a Fibonacci message. To show the protocol defines a tree, we first check there are no collisions on any step, i.e. all addresses α sent on a step map to distinct $\langle\alpha\rangle_d$ modulo $b_d(t)$:

I. In phase I, all addresses of messages of either type are of the form $\alpha = 0^r\beta$ where $\beta \in \mathcal{B}_{d,t-r}$. Since $1^r\beta 0^{t-s} \in \mathcal{B}_{d,t}$, the values $\langle 1^r\beta 0^{t-s}\rangle_d$ are all distinct modulo $b_d(t)$ by lemma 4.1. Since $\langle\alpha\rangle_d = \langle 1^r\beta 0^{t-s}\rangle_d - \langle 1^r 0^{t-r}\rangle_d$, the $\langle\alpha\rangle_d$ must also be distinct.

II. In phase II, since all addresses of messages sent to row s are of the form $\gamma 0^{r-s}\beta$ for $\gamma \in \mathcal{F}_{d,s}, \beta \in \mathcal{B}_{d,t-r}$. The shifted addresses $\gamma 1^{r-s}\beta$ are all in $\mathcal{B}_{d,t}$, hence the $\langle\alpha\rangle_d$ are distinct by a similar argument.

To finish showing the protocol defines a tree, we check there are no vertices appearing in two generations. This could only happen if generation t collides with generation 0 (the originator) in row r. But all the addresses received by generation t are in $\mathcal{F}_{d,r}\mathcal{B}_{d,t-r}$, and hence not congruent to 0.

Now we consider the bit overhead to implement this protocol. The entire protocol is oblivious to x, which just translates the tree vertices in $\mathbb{Z}_{b_d(t)}$. During the protocol no use is ever made of the bits of α; all the vertices need to know is the time τ (which must be given to them in the message overhead) and what kind of edge delivers the message (which we have assumed they know for free). At the end of the protocol each

vertex $(y,s) \in T2_{d,t}(x,r)$ needs to know whether their $\alpha \in \mathcal{B}_{d,t}$ to decide if it is the leader of column y. Since $\gamma \in \mathcal{B}_{d,r}$ and $\beta \in \mathcal{B}_{d,t-r}$, the only way α could fail to be in $\mathcal{B}_{d,t}$ is if the substring 0^d1 appears on the boundary between γ and β. To decide, (y,s) needs to know:

- whether γ has a 1 in its last d bits and where it is,
- how many consecutive 0's begin β (either $< d$ or all of β).

The first may be determined just from the identity of the sender (local information); the second may be maintained with $\lg d + O(1)$ additional bits overhead per message. With $\lg t$ more bits to maintain the clock, we use the claimed number of bits. Since the resulting partial broadcasts are tree-shaped, this protocol works asynchronously as well. ∎

For our third construction *FIB3* our goal is to reduce the number of messages sent by the originator. We introduce a new parameter $l \leq d$ that corresponds to the $L(n)$ function of section 2. We modify *FIB2* to work when the originator sends only l messages. Define

$$b_d(l,t) = 1 + \sum_{i=1}^{l} b_d(t-i).$$

In particular $b_d(0,t) = 1$. Then $b_d(l,t)$ is the size of the maximum t-step broadcast tree $T_{d,l,t}$ whose root has degree l and all other vertices have outdegree at most d. We make the following simple estimate:

Lemma 4.4 $b_d(l,t) \geq (1-2^{-l})b_d(t) + 1$.

Proof: Since $b_d(t-i) \geq 2^{-i}b_d(t)$, $b_d(l,t) - 1 \geq \sum_{i=1}^{l} 2^{-i}b_d(t) = (1-2^{-l})b_d(t)$. ∎

We construct digraph $FIB3_{d,l,t}$ on vertex set $\mathbb{Z}_{b_d(l,t)} \times \mathbb{Z}_t$. We use the same edge class definitions 1, 3, and 4 used for constructing $FIB2_{d,t}$ (but now x in the definitions ranges over a different number of columns). We enlarge class 2 and define a new class 5 of 'Root' edges going horizontally across rows (see figure 5(a)):

Class 2': For all x, $t-l-d \leq r < t$, $1 \leq s \leq d$, connect $(x - f_d(s), r)$ to (x,s).

Class 5: For all (x,r), $1 \leq i \leq l$, connect (x,r) to $(x + b_d(i-1,t), r)$.

Theorem 4.5 *For every vertex $(x,r) \in FIB3_{d,l,t}$ there is a t-step partial broadcast tree $T3_{d,l,t}(x,r)$ rooted at (x,r) such that:*

(i) $T3_{d,l,t}(x,r)$ contains at least one vertex in each column y.

(ii) *The root (x, r) has l children, all connected by Root edges. The subtree rooted at the ith child spans columns $x + b_d(i-1, t)$ through $x + b_d(i, t) - 1$.*

(iii) *All vertices besides the root have degree at most $2d$.*

(iv) *With $\lg l + \lg d + \lg t + O(1)$ bits overhead per message we may implement this protocol and furthermore appoint a unique 'leader' vertex of $T3_{d,l,t}(x, r)$ in each column.*

Proof: We have arranged the edges so that for any originator (y, r) and any i such that $1 \leq i \leq l$, we may run the broadcast protocol for $FIB2_{d,t-i}$ in rows $0, \ldots, t-i$. The only new trick is that if $t - i \leq r < t$, then we use the $FIB1_{d,t-i}$ protocol, i.e. (y, r) sends Fibonacci messages to rows $1, \ldots, d$, using class 2' edges. This generates a copy of $T2_{d,t-i}(y, r)$ spanning and confined to columns $y, \ldots, y + b_d(t-i) - 1$ of $FIB3_{d,l,t}$. We call this the $FIB2_{d,t-i}$ subprotocol started by (y, r).

To start the main $FIB3_{d,l,t}$ protocol from the root (x, r), the root on step i ($1 \leq i \leq l$) sends a message to its ith child $(x + b_d(i-1, t), r)$ telling it to start running the $FIB2_{d,t-i}$ subprotocol. We have spaced the Root edges so that the child subtrees span disjoint consecutive blocks of columns, hence the subprotocols never collide. Together with the root, they span every column of $FIB3_{d,l,t}$. To control the ith subprotocol we need to send along the value of i with those messages, so that they know which rows to skip when they wrap around. This introduces $\lg l$ additional message overhead bits to those already needed to control the $FIB2_{d,t-i}$ subprotocols. To select column leaders, we simply select the leaders from each subprotocol together with the root (x, r). ∎

We remark that this *FIB3* protocol in fact never uses the Fibonacci edges to row t, and so there is a slightly better construction where *FIB3* has only $t - 1$ rows. We have chosen to avoid this modification for simplicity of presentation.

5 Main Constructions

Using the constructions of the previous two sections, we are now ready to construct broadcast digraphs and relaxed broadcast digraphs within constant factors of the lower bounds of section 2. Note that for broadcast graphs, if $L(n)$ is within a constant factor of $\lg n$, say $L(n) \geq \lceil \lg n \rceil / 3$, then we may use the boolean difference digraph $BD(n)$, which has $n \lceil \lg n \rceil \leq 3 \cdot \vec{B}(n)$ edges and maxdegree $2 \lceil \lg n \rceil \leq 3 \cdot \vec{D}(n)$, i.e. both are both within constant factors of the lower bounds.

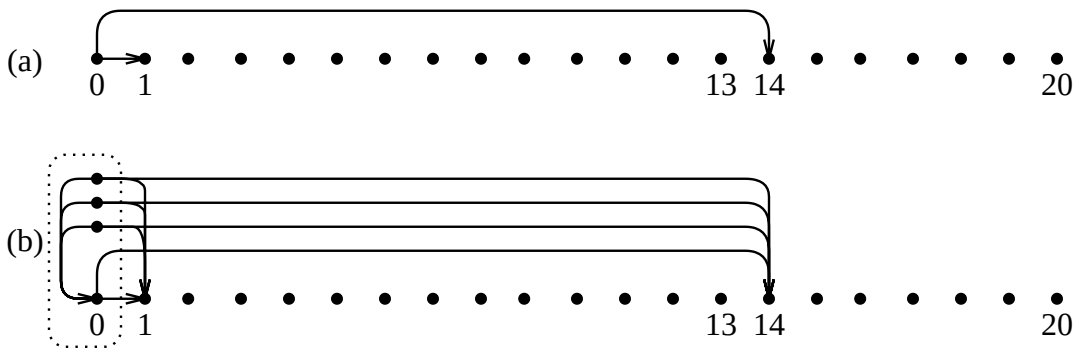

Figure 5: (a) Class 5 edges from node 0 in some row r of *FIB3* ($d = 3$, $l = 2$, $t = 5$). (b) Corresponding class 5' edges from $S(0, r)$ in *FIB4*. Node 1 is responsible for columns 1–13, node 14 for columns 14–20.

Otherwise (when $L(n)$ is $o(\lg n)$, or when we want relaxed broadcast graphs) we rely on the constructions of section 4. We want to augment *FIB3* with additional vertices so that most vertices have degree $O(l)$ but are still able to originate a broadcast. The general idea is to put each vertex (x, r) of *FIB3* in charge of a small set of vertices $S(x, r)$, such that every vertex in the set may originate a broadcast in the same way as (x, r), and (x, r) can broadcast a received message back to all of $S(x, r)$. Thus most of the vertices in $S(x, r)$ may have lower degree than their leader (x, r). This idea is detailed below.

We construct digraph $FIB4_{d,l,t_1,t_2}$ (with parameters $1 \le l \le d \le t_1/2$, $\lg t_1 \le t_2$) by augmenting $FIB3_{d,l,t_1}$. We add new vertices in this construction; we refer to the original vertices of $FIB3_{d,l,t_1}$ as *FIB3*-vertices.

Define $\tau = \lceil \lg t_1 \rceil$. For each column x of $FIB3_{d,l,t_1}$ construct a copy $H(x)$ of $H_{\tau,t_2-\tau}$ (from section 3). Identify the t_1 *FIB3*-vertices of column x with t_1 distinct roots of $H(x)$; the other $2^{t_2} - t_1$ vertices of $H(x)$ are new to the digraph *FIB4*. Repeating this for every column x defines all the new vertices of *FIB4*.

Partition the vertices of $H(x)$ into t_1 subsets $S(x, s)$ of size at most $\lceil 2^{t_2}/t_1 \rceil \le 2^{t_2-\tau+1}$, so that $(x, s) \in S(x, s)$. Thus we have partitioned all the vertices of *FIB4* into approximately equal size sets represented by the *FIB3*-vertices. Now add the following class of Root edges, extending the previous definition of class 5 (see figure 5(b)):

Class 5': For all *FIB3*-vertices (x, r) and all $1 \le i \le l$, connect each $v \in S(x, r)$ to $(x + b_d(i-1, t), r)$. Also if $v \ne (x, r)$, connect v to (x, r).

We now bound the number of edges and the maxdegree of *FIB4*.

Lemma 5.1 *Given $1 \le l \le d \le t_1/2$, $\tau = \lceil \lg t_1 \rceil \le t_2$, then $FIB4_{l,d,t_1,t_2}$ has exactly $n = 2^{t_2} \cdot b_d(l, t_1)$ vertices, less than $n(l+2) + b_d(l,t_1) \cdot 2t_1(d+\tau+1)$ directed edges, and maxdegree less than $3(d+l+t_2+2) + (l+1)2^{t_2}/t_1$.*

Proof: There are $b_d(l,t_1)$ columns y and 2^{t_2} vertices in each $H(y)$, hence the number of vertices. To count edges and maxdegree, there are six kinds of edges: those in classes 1, 2', 3, 4, and 5', and those in the $H(y)$ subgraphs. First we count edges. Every vertex has at most $l+1$ outedges of class 5', and at most one incoming tree-edge from $H(y)$, hence the leading term of $n(l+2)$. Now we count the remaining edges contributed per column, and then multiply by $b_d(l,t_1)$, the number of columns. In column y, class 1 contributes $< dt_1$ edges, class 2' contributes $d(l+d) \le dt_1$ edges, classes 3 and 4 contribute $< t_1$ edges each, and the digraph $H(y)$ has $\tau 2^\tau < 2t_1\tau$ root edges besides the tree edges already counted above. Altogether this gives the claimed bound on edges.

To bound maxdegree we simply add the maxdegrees of the six cases. Class 1 has maxdegree $2d$ (note we must count both in and out degree), class 2' has maxdegree $l+d$, classes 3 and 4 have maxdegree 2 each, class 5' has maxdegree $\lceil 2^{t_2}/t_1 \rceil (l+1)+l < (l+1)(2+2^{t_2}/t_1)$ (attained at the *FIB3*-vertices), and each $H(y)$ subgraph has maxdegree $t_2 + \tau < 3t_2$. Adding these gives the claimed bound on maxdegree. ∎

Theorem 5.2 *For every vertex $v \in FIB4_{d,l,t_1,t_2}$ there is a (t_1+t_2)-step broadcast protocol starting from v. Furthermore:*

(i) After t_1 steps of the protocol, for each column y there is a root vertex in $H(y)$ that knows the message.

(ii) The maximum number of messages sent by any vertex is $2d + t_2$.

(iii) We may implement this protocol with $\lg l + \lg t + \lg d + O(1)$ bits overhead per message in the synchronous model.

Proof: Let (x,s) be the *FIB3*-vertex such that $v \in S(x,s)$. The protocol proceeds in two phases; the first phase of t_1 steps is essentially like the *FIB3* protocol, with v taking the role of (x,s). For steps $1 \le i \le l$, v sends the message along a Root edge (class 5') to $(x + f_d(i-1,t), s+i)$. Just as in the *FIB3* protocol, these children of v initiate a local $FIB2_{d,t_1-i}$ protocol. Hence after t_1 steps, all of these subprotocols finish simultaneously and appoint a *FIB3*-vertex leader in every column except column x (since these leaders are *FIB3*-vertices they are also roots of their respective $H(y)$'s). Column x itself is a special case; if $v = (x,s)$, then v itself is the leader of column x.

Otherwise, v sends the message to (x, s) on step $l + 1$; there is time to do this since we have assumed $l < t_1$.

At the start of the second phase, the *FIB3* protocol has established a unique root leader in each *FIB3*-column y. This leader is a root in the corresponding $H(y)$, so simply follow the $H(y)$ t_2-step protocol to notify every vertex of *FIB4*. Hence every vertex knows the message in $t_1 + t_2$ steps.

Note that the leader of column x receives the message on step $l + 1$ and may as well start broadcasting up $H(x)$ on step $l + 2$. Since v will then receive a copy of its own message, we cannot claim that the $H(x)$ protocol is tree shaped; nevertheless it will work asynchronously because the message that v receives is a descendant of v's last outgoing message. The message overhead is essentially the same as for the $FIB3_{d,l,t_1}$ protocol, since the second phase introduces no new costs. ∎

This *FIB4* protocol will not work in the asynchronous model. Phase 1 will appoint a unique leader in a given column y, but it may also make temporary use of other vertices from that column (as in the *FIB2* protocol). In phase 2 the leader broadcasts to all of $H(y)$, including these 'temporary' vertices used in the phase 1. Hence in the asynchronous model one of the temporary vertices may simultaneously receive two messages, one from phase 1 and the other from phase 2. Just from the general arguments of section 2 we know there is an asynchronous tree-shaped protocol, but with $O(\lg n)$ bits overhead.

Note that if t_2 is strictly greater than τ (i.e. $H(y)$ is not just a hypercube) then at least half the vertices in *FIB4* are deletable; these are the leaves of the $T_{t_2-\tau}$ subtrees used to build each $H(y)$. Removing some or all of these vertices will not disrupt the protocol, since they are always the last to receive the message in any broadcast. Deleting vertices preserves the leading term $n(l + 2)$ in the statement of lemma 5.1, since each vertex contributed at most $l + 2$ edges to that term.

Given n, we now show that by setting the *FIB4* parameters appropriately and possibly deleting some vertices, we get broadcast digraphs and relaxed broadcast digraphs of size n with degree and edge costs within constant factors of their lower bounds. The choices for l, d, and $t = t_1 + t_2$ are more or less fixed for us; our main freedom is in partitioning t into t_1 and t_2. As t_2 increases, the number of edges decreases and the maxdegree increases, but both are reasonably small for t_2 around $\lg dt/l$. The following two theorems make this precise.

Theorem 5.3 *Given n such that $L(n) < (1/3)\lg n$, let $t = \lceil \lg n \rceil$. Choose $l = L(n) + 2$, $d = \lceil \lg \lg n \rceil + l$, $t_2 = \lceil \lg dt/l \rceil + 1$, $t_1 = t - t_2$. Then $FIB4_{d,l,t_1,t_2}$ (possibly*

after deleting some leaves) is a t-step n-vertex broadcast digraph with $O(L(n) \cdot n)$ edges and $O(L(n) + \lg\lg n)$ maxdegree.

Proof: First we show *FIB4* has at least n vertices. By the definition of $L(n)$ we know $n \le 2^t(1 - 2^{-(L(n)+1)})$. Thus $n \le 2^t(1 - 2^{1-l}) < 2^t(1 - 2^{-l})^2$. By lemma 4.4 we have $b_d(l,t) \ge (1 - 2^{-l})b_d(t)$, and by lemma 2.8 we have $b_d(t) \ge (1 - 2^{-l})2^t$, hence $b_d(l,t) \ge n$. For any $k \le t$ we may estimate $2^k b_d(l, t-k) \ge b_d(l,t)$, so in particular ($k = t_2$) *FIB4* has at least n vertices.

We estimate $t_1 < t$, $\tau \le \lceil \lg\lg n \rceil$, $2dt/l \le 2^{t_2} < 4dt/l$, $t_2 < 2\lceil \lg\lg n \rceil$, and $b_d(l, t_1) \le 2^{t_1} = 2^t/2^{t_2} < 2^t(l/2dt) < nl/dt$. Now we apply lemma 5.1. The number of edges is $< n(l+2) + b_d(l,t_1) \cdot 2t_1(d + \tau + 1) < n(l+2) + (nl/dt) \cdot 2t \cdot 2d = 5nL(n) + 12n = O(nL(n))$. Similarly the maxdegree is $< 3(d + l + t_2 + 2) + (l+1)2^{t_2}/t_1 < 3(3d) + (l+1)(4dt/l)/t_1$. Since $t/t_1 \le 3/2$ (for sufficiently large n) and $(l+1)/l \le 4/3$, the maxdegree is at most $17d = O(L(n) + \lg\lg n)$.

Finally we need $2^{t_2} \ge 2t_1$ so that there are enough deletable leaf vertices to get exactly n vertices. This is guaranteed by our choice of t_2. ∎

Corollary 5.4 *$B(n), \vec{B}(n) = \Theta(L(n) \cdot n)$, and $D(n), \vec{D}(n) = \Theta(L(n) + \lg\lg n)$.*

We comment that an alternative construction (presented in a previous version of this paper [Pe]) yields a better bound of $(L(n)+2)n$ on the number of edges. However, that construction has linear indegree.

Theorem 5.5 *Given n such that $L(n) < (1/3)\lg n$, let $t = \lceil \lg n \rceil + 1$. Choose $l = 2$, $d = \lceil \lg\lg n \rceil$, $t_2 = \lceil \lg dt \rceil$, $t_1 = t - t_2$. Then $FIB4_{d,l,t_1,t_2}$ (possibly after deleting some leaves) is a t-step n-vertex relaxed broadcast digraph with $O(n)$ edges and $O(\lg\lg n)$ maxdegree.*

Proof: Again we start by showing *FIB4* has at least n vertices. We know $n < 2^{t-1} < (1 - 2^{-l})^2 2^t$. Arguing as in the last proof we have $n < b_d(t)(1 - 2^{-l}) < b_d(l,t) < 2^{t_2} b_d(l, t_1)$, hence the graph is large enough.

We estimate $t_1 < t$, $\tau \le \lceil \lg\lg n \rceil$, $dt \le 2^{t_2} < 2dt$, $t_2 < 2\lceil \lg\lg n \rceil$, and $b_d(l, t_1) \le 2^{t_1} = 2^t/2^{t_2} \le 2^t/dt < 4n/dt$. Again we apply lemma 5.1. The number of edges is $< 21n = O(n)$ and the maxdegree is $< 17d + 12 = O(\lg\lg n)$.

We need $2^{t_2} \ge 4t_1$ to insure there are enough deletable vertices; this is guaranteed by our choice of t_2 for n sufficiently large. ∎

Corollary 5.6 *$B'(n), \vec{B}'(n) = \Theta(n)$, and $D'(n), \vec{D}'(n) = \Theta(\lg\lg n)$.*

By theorem 5.2 we know that the graphs of theorems 5.3 and 5.5 have synchronous protocols with $O(\lg\lg n)$ bit overhead, while we need $O(\lg n)$ bits to get tree-shaped asynchronous protocols. We have no corresponding lower bounds on bit complexity.

6 c-Broadcast Graphs

A wide variety of models for broadcast (and communication networks in general) are considered in the literature. Examples of such models are radio broadcast networks (cf. [GVF]) and line broadcasts (cf. [F2]). One particular variant of the model considered here is a model which allows vertices to communicate simultaneously with all their neighbors in one time unit. This model, which is quite common in the field of synchronous distributed and parallel computing (cf. [A, BD, Fi] among others), is a natural one to consider when the system supplies hardware mechanisms enabling such an operation, or in cases where message transmission time is negligible compared to the time required for processing within the vertices between consecutive rounds. In such a model, broadcast can be achieved in time D in every network of diameter D, hence in the complete network broadcast requires only one time unit.

The dichotomy between the above two extreme models suggests a natural intermediate model which provides for "conference calls" of limited size, i.e., in which a vertex is allowed to send a message simultaneously to up to c neighbors at a time, for some constant $c \geq 1$. We refer to broadcast in this model as *c-broadcast*. In this section we extend the basic results known for $c = 1$ to every $c \geq 1$. (A related generalization is studied in [RL]; there, the communication network is represented by a $(c+1)$-uniform hypergraph, and each conference call involves the vertices of some hyperedge.)

Similar to the definitions of the standard (1-broadcast) model, let $b_c(u, G)$ denote the minimum time required to broadcast from the vertex u in the graph G in the c-broadcast model, and let $b_c(G) = \max\{b_c(u, G) \mid u \in V(G)\}$. The obvious lower bound on the time needed for c-broadcast is

Lemma 6.1 *$b_c(G) \geq \left\lceil \log_{c+1} n \right\rceil$ for every n-vertex network G.* ∎

Consequently, a c-broadcast graph (respectively, relaxed c-broadcast graph) is an n-vertex communication network G such that $b_c(G) = \left\lceil \log_{c+1} n \right\rceil$ (resp., $b_c(G) = \left\lceil \log_{c+1} n \right\rceil + 1$), and $B_c(n)$ (resp., $B'_c(n)$) denotes the minimum number of edges of any n-vertex c-broadcast graph (resp., relaxed c-broadcast graph).

We first extend the $n = 2^k$ result of [FHMP] to the c-broadcast model.

Theorem 6.2 *$B_c(n) = \frac{c}{2}nk$ for every $n = (c+1)^k$, $k \geq 1$.* ∎

The results of the previous sections can be extended as well. Denote the exact number of consecutive leading c's in the $(c+1)$-ary representation of $n-1$ by $L_c(n)$.

Extending the lower bound of theorem 2.2, and using upper bound constructions from [Pe], we have the following.

Theorem 6.3 *For all $n \geq 1$, $\frac{c}{2}(L_c(n)-1)n < B_c(n) < [c(L_c(n)+1)+1]n$, i.e. $B_c(n) = \Theta(c \cdot L_c(n) \cdot n)$.* ∎

Theorem 6.4 *$B'_c(n) < 2n$ for every $n \geq 1$ and $c \geq 1$.* ∎

We may define the analogous maximal c-broadcast tree where each vertex sends messages at most d times (and hence has outdegree at most cd); the generation sizes are given by the sequence $f^c_d(s) = c \cdot \sum_{i=1}^{d} f^c_d(s-i)$ with growth rate $\lambda \sim (c+1) - c/(c+1)^d$. We may then extend theorem 2.5.

Theorem 6.5 *For every $n \geq 1$, $D_c(n) = \Omega(L_c(n) + \log_{c+1}\log n)$ and $D'_c(n) = \Omega(\log_{c+1}\log n)$.* ∎

The numbering scheme and first construction of section 4 carry through analogously. Further pursuit of the methods of section 5 may yield a construction proving the previous theorem tight as well; we have not pursued this further.

7 Open Problems

A number of other interesting problems suggest themselves for further study. A significant area of problems concerns the design of broadcast schemes for given networks (as opposed to networks designed specifically for the purpose of broadcast). It is known that both determining the broadcast time $b(v, G)$ of an arbitrary vertex v in an arbitrary graph G [SCH] and recognizing a broadcast graph [FHMP] are NP-complete. Consequently, heuristic approaches for the problem of determining a near-optimal broadcast strategy in an arbitrary network were studied in [SW], and exact solutions were provided for special families of graphs, such as trees [P, SCH] and grids [FH]. This line of research seems especially important, as in most cases the designer of a broadcast scheme faces an existing network with a fixed topology. Natural classes of graphs to be considered are families such as regular, planar and bounded-degree graphs.

A related problem is that of distributing distinct pieces of information from *several* originators in the network simultaneously. In its ultimate form, this problem turns into the well-known *gossip problem* (cf. the bibliography of [HHL]). This problem involves n items of information, each initially held in one of the vertices, and the question is what resources (messages, time, edges, etc.) are required to let everyone know everything. This problem assumes a model in which a single message can carry an unlimited amount of information (or at least $O(n)$ bits). More realistic assumptions allow a message to carry no more than $O(\log n)$ bits of information, which makes the intermediate levels of the problem (i.e., with a limited number of originators) interesting in their own right.

Another interesting issue from a theoretical point of view is that of broadcasting on random graphs. The radius of random graphs has been well studied, but it may be worth looking at the broadcast radius $b(G)$ of random graphs. Pure random graphs will not make good broadcast graphs just out of degree constraints, but random graphs may still be useful components (e.g. use random edges instead of Fibonacci edges). More importantly these random graphs may be fault tolerant. One may consider using a random protocol as well.

8 Acknowledgments

We would like to thank Tom Leighton for his encouragement, Art Liestman for commenting on a previous version of the paper and Alex Schäffer for helpful discussions.

References

[A] B. Awerbuch, "Complexity of Network Synchronization," *J. of the ACM*, **v. 32**, pp. 804–823, 1985.

[BD] Ö. Babaoğlu and R. Drummond, "Time-Communication Tradeoffs for Reliable Broadcast," Report TR 85-687, Cornell University, 1985.

[BHLP1] J.-C. Bermond, P. Hell, A.L. Liestman and J.G. Peters, "Broadcasting in Bounded Degree Graphs," Technical Report TR 88-5, Simon Fraser University, Burnaby, Canada, 1988.

[BHLP2] J.-C. Bermond, P. Hell, A.L. Liestman and J.G. Peters, "New Minimum Broadcast Graphs and Sparse Broadcast Graphs," *Discrete Applied Math.*, to appear.

[CL1] S.C. Chau and A.L. Liestman, "Constructing Minimal Broadcast Networks," *J. Combin. Info. & Syst. Sci.*, **v. 10**, No. 1, pp. 110–122, 1985.

[CL2] S.C. Chau and A.L. Liestman, "Constructing Fault-Tolerant Minimal Broadcast Networks," *J. Combin. Info. & Syst. Sci.*, **v. 11**, No. 1, pp. 1–18, 1986.

[F1] A.M. Farley, "Minimal Broadcast Networks," *Networks*, **v. 9**, pp. 313–332, 1979.

[F2] A.M. Farley, "Minimum-Time Line Broadcast Networks," *Networks*, **v. 10**, pp. 59–70, 1980.

[FH] A.M. Farley and S.T. Hedetniemi, "Broadcasting in Grid Graphs," *Proc. 9th Southeastern Conf. Combin., Graph Theory, Comput.*, pp. 275–288, 1978.

[FP] A.M. Farley and A. Proskurowski, "Broadcasting in Trees with Multiple Originators," *SIAM J. Alg. Disc. Meth.*, **v. 2**, pp. 381–386, 1981.

[FHMP] A.M. Farley, S.T. Hedetniemi, S. Mitchell and A. Proskurowski, "Minimum Broadcast Graphs," *Discrete Math.*, **v. 25**, pp. 189–193, 1979.

[Fi] M.J. Fischer, "The Consensus Problem in Unreliable Distributed Systems (A Brief Survey)," Report YALEU/DCS/RR-273, Yale University, 1983.

[GVF] I. Gitman, R. M. Van Slyke and H. Frank, "Routing in Packet-Switching Broadcast Radio Networks," *IEEE Trans. Comm.*, pp. 926–930, 1976.

[HHL] "A Survey of Gossiping and Broadcasting in Communication Networks," S. M. Hedetniemi, S. T. Hedetniemi and A. L. Liestman, *Networks*, **v. 18**, pp. 319–349, 1988.

[HL] P. Hell and A. L. Liestman, "Broadcasting in One Dimension," *Discrete Applied Math.*, **v. 21**, pp. 101–111, 1988.

[K] D. Knuth, *The Art of Computer Programming*, **v. 3**, Addison-Wesley, 1973.

[L] A. L. Liestman, "Fault-Tolerant Broadcast Graphs," *Networks*, **v. 15**, pp. 159–171, 1985.

[LP] A.L. Liestman and J.G. Peters, "Broadcast Networks of Bounded Degree," *SIAM J. Discr. Math.*, **v. 1**, pp. 531–540, 1988.

[MH] S. Mitchell and S.T. Hedetniemi, "A Census of Minimum Broadcast Graphs," *J. Combin., Info. & Syst. Sci.* **v. 5**, pp. 141–151, 1980.

[Pe] D. Peleg, "Tight Bounds on Minimum Broadcast Networks," manuscript, July 1987.

[P] A. Proskurowski, "Minimum Broadcast Trees," *IEEE Trans. on Computers*, **v. 30**, pp. 363–366, 1981.

[RL] D. Richards and A.L. Liestman, "Generalizations of Broadcasting and Gossiping," *Networks*, **v. 18**, pp. 125–138, 1988.

[SW] P. Scheuermann and G. Wu, "Heuristic Algorithms for Broadcasting in Point-to-Point Computer Networks," *IEEE Trans. on Computers*, **v. 33**, pp. 804–811, 1984.

[SCH] P. J. Slater, E. J. Cockayne and S.T. Hedetniemi, "Information Dissemination in Trees," *SIAM J. on Computing*, **v. 10**, pp. 692–701, 1981.

[Wa] X. Wang, $B(18) = 23$, private communication reported in [HHL], April 1986.

[We] D. B. West, "A Class of Solutions to the Gossip Problem, Part I," *Discrete Math.*, **v. 39**, pp. 307–326, 1982.