On the Time-Complexity of Broadcast in Multi-Hop Radio Networks: An Exponential Gap Between Determinism and Randomization

Reuven Bar-Yehuda *Oded Goldreich* *Alon Itai*
Department of Computer Science
Technion - Israel Institute of Technology
Haifa 32000, ISRAEL

-- --

Proposed running head:

The Complexity of Radio Broadcast

Mailing address:

Oded Goldreich,
Department of Computer Science,
Technion - Israel Institute of Technology,
Haifa 32000, ISRAEL.

Keywords: Multi-hop Radio Communication, Broadcast in Radio Networks, Randomized Protocols in Distributed Computing, Lower Bounds in Distributed Computing.

The second author was partially supported by grant No. 86-00301 from the United States - Israel Binational Science Foundation (BSF), Jerusalem, Israel. A preliminary version of this paper [BGI87] has appeared in the *6th ACM Symp. on Principles of Distributed Computing (PODC).*

ABSTRACT

The time-complexity of deterministic and randomized protocols for achieving broadcast (distributing a message from a source to all other nodes) in arbitrary multi-hop radio networks is investigated. In many such networks, communication takes place in synchronous time-slots. A processor receives a message at a certain time-slot if exactly one if its neighbors transmits at that time-slot. We assume no collision-detection mechanism, i.e., it is not always possible to distinguish the case where no neighbor transmits from the case where several neighbors transmit simultaneously.

We present a randomized protocol that achieves broadcast in time which is optimal up to a logarithmic factor. In particular, with probability $1-\varepsilon$, the protocol achieves broadcast within $O((D+\log n/\varepsilon)\cdot\log n)$ time-slots, where n is the number of processors in the network and D its diameter. On the other hand, we prove a linear lower bound on the deterministic time-complexity of broadcast in this model. Namely, we show that any deterministic broadcast protocol requires $\Theta(n)$ time-slots, even if the network has diameter 3, and n is known to all processors. These two results demonstrate an exponential gap in complexity between randomization and determinism.

1. INTRODUCTION

Channels in computer networks are commonly divided into two categories: point-to-point channels (also known as store-and-forward) and multi-access channels. These categories are very different in nature and each has its advantages and disadvantages making it more suitable to various applications [T81]. The fundamental feature of multi-access channels is that a message placed on the channel is delivered to all stations sharing the channel *if and only if* it is the only message placed on the channel at ‘‘that time’’. If two (or more) messages are placed on the channel (by different stations) at the same time, a collision occurs and none of these messages is delivered. Most works assume that such collision can be detected yet this assumption cannot always be justified (cf. [G85]). In particular, in radio channels, which constitute an important type of multi-access channels, collision is hard to distinguish from the noise that is always present on the channel (cf. [G85]). Furthermore, also in settings where collision is normally detected, it is desirable not to rely on the collision detection mechanism: a communication protocol which does not use collision detection is likely to be more reliable (than one which does use it) since the protocol will not fail in case of undetected collision.

Here we consider multi-access channels without collision detection. As radio transmission constitutes the most popular example of such channels, we carry on our discussion using the terminology of radio networks. In this terminology a single channel is a *single-hop* radio network [A70], whereas a network consisting of several different channels is a *multi-hop* radio network.

A useful (and sometimes unavoidable) paradigm of radio communication is the structuring of communication into time-slots. This paradigm is commonly adopted in the practical design of protocols and hence the use of the paradigm in the theoretical analysis of radio communication is justified [R72] (cf. [G85, sec. IV.A] and [T81, sec. 6.1.2]).

We now present explicitly the model used throughout the paper. The model consists of an arbitrary multi-hop (undirected) network, with processors communicating in synchronous time-slots subject to the following rules. In each time-slot, each processor acts either as a *transmitter* or as a *receiver*. A processor acting as a receiver is said to receive a message in time-slot t if exactly one of its neighbors transmits in time-slot t. The message received is the one transmitted. If more than one neighbor transmits in that time-slot, we say that a *conflict* has occurred. We assume that conflicts (or ‘‘collisions’’) are not detected, hence a processor cannot distinguish the case in which no neighbor transmits from the case in which more than one of its neighbors transmits during that time-slot. The topology of the entire network is not a priori known to the processors. Since communication is synchronous the main difficulty in routing messages, in this model, is the possibility of *conflicts*; that is, situations when several neighbors of a processor transmit simultaneously and (as a result) it receives nothing. This difficulty is aggravated when the processors have no a priori knowledge on the topology of the entire network.

We investigate the complexity of implementing broadcast in the above model. *Broadcast* is a task initiated by a single processor, called the *source*, transmitting a single *message*(1). The goal is to have the message reach all processors in the network. We consider both deterministic and randomized protocols for broadcast and concentrate on their time-complexity (i.e., the number of time-slots required to complete broadcast). Our results demonstrate the advantage of using randomization in the above model.

1.1. Randomized Protocols

We show how conflicts, arising in broadcast protocols, can be resolved quickly by using randomization. In particular, we present a randomized broadcast protocol that always terminates and, with probability $\geq 1-\varepsilon$, succeeds after $O((D+\log n/\varepsilon)\cdot\log n)$ time-slots, where D is the network's diameter (distance between its most distant processors) and n the number of processors. Thus, the complexity is only a logarithmic factor away from the trivial lower bound (i.e., the diameter of the network). The only inputs required by our protocol are the number of processors in the network – n, and the error bound – ε.

1) There is some confusion regarding the term broadcast. In particular, some authors use *broadcast* to mean the task of distributing (many) messages to all processors in a network. A first step in the design of broadcast protocols is the design of protocols which handle correctly the broadcast of a single message (and indeed our paper which handles the single-message broadcast was followed by [BII89] in which broadcasting an arbitrary number of messages was investigated). In the rest of this paper, broadcast will mean the simpler task of single-message broadcast. This convention is in accordance with a significant number of papers.

Our protocol performs almost as well when given instead of the actual number of processors (i.e., n), a "good" upper bound on this number (denoted N). An upper bound polynomial in n yields the same time-complexity, up to a constant factor (since complexity is logarithmic in N).

Our protocol does not use processor IDs, and thus does not require that the processors have distinct IDs (or that they know the identity of their neighbors). Furthermore, a processor is not even required to know the number of its neighbors. This property makes our protocol adaptive to changes in topology which occur throughout the execution, and resilient to non-malicious faults.

The protocol is conceptually simple, and requires a minor amount of local computation. All that is needed is to toss one coin and to increment a counter, at each time-slot.

The basic idea used in the protocol is to resolve potential conflicts by randomly eliminating half of the transmitters. This process of ‘‘cutting by half’’ is repeated each time-slot with the hope that there will exist a time-slot with a single active transmitter. The "cutting by half" process is easily implemented distributedly by letting each processor decide randomly whether to eliminate itself.

1.2. Deterministic Lower Bound

No deterministic protocol can achieve broadcast in radio networks when processors do not have unique IDs. To see why, consider the case where the network consists of n processors arranged so that the source is connected to two nodes of an $(n-1)$-clique. With no IDs, the conflict between these two nodes cannot be resolved deterministically. Thus, the (above mentioned) use of randomness ‘‘beats’’ an impossibility result.

A more reasonable model is one where the processors have unique IDs. The impossibility result does not hold in this case, as broadcast can be achieved (e.g. by a DFS-like procedure). However, we show that in this model (of distinct IDs) the use of randomization allows a dramatic improvement. In this case, the improvement is in complexity. We show a lower bound of $\Omega(n)$ for the number of time-slots in a deterministic broadcast protocol running on a network of diameter 3. This should be contrasted with the number of time-slots of the randomized protocol on such networks, which is $O((\log n/\varepsilon)\cdot\log n)$ (terminating with probability $1-\varepsilon$).

1.3. Related Work

In this work we consider ‘‘collision resolution’’ in (*multi-hop*) radio network *without* collision detection mechanisms. By ‘‘collision resolution’’ we mean guaranteeing the receipt of an arbitrary message sent by one of the processors wishing to deliver a message at this stage. A seemingly related problem which has received much attention in the late 70’s and early 80’s is that of ‘‘collision resolution’’ in (*single-hop*) radio network (also known as multi-access channels) *with* collision detection mechanisms (e.g., [C79, H78, TM79, GL83, GW85]). In these works, however, ‘‘collision resolution’’ means guaranteeing the receipt of all messages sent by processors wishing to deliver a message at this stage.

Gaps between the power of determinism and randomization are quite common in distributed computing. In the context of radio networks (with collision detection mechanisms) randomization is used in practice to resolve conflicts (cf. [A70, T81]). The key role of randomization in that context was demonstrated in [GW85] that contrasted with (for example) [GL83] yields a gap between the power of determinism and randomization. The gap is essentially a multiplicative factor which is logarithmic in the number of processors sharing the channel. The gap we demonstrate is between the power of determinism and randomization in the context of radio communication without collision detection mechanisms. We show that the time required by deterministic procedures is exponential in the time required by randomized ones.

Our lower bound argument introduces a combinatorial game which seems similar to ‘‘group testing’’, a problem that has been used in the context of multi-access channels (cf. [W85]). However, to the best of our knowledge, research on group testing concentrates on monotone feedback functions and the average cost with respect to specific instance distribution.

Our protocol can be thought of as consisting of a distributed algorithm for finding a broadcast schedule (i.e., an assignment of processors to be transmitters and receivers in specified time-slots) and a trivial protocol using the schedule. It is thus interesting to contrast our results with the results known for the time complexity of centralized algorithms for finding broadcast schedules. Chlamtac and Kutten [CK85] showed that, given a network and a designated source, finding an optimal broadcast schedule (i.e., a schedule which uses the minimum number of time-slots) is NP-Hard. Chlamtac and Weinstein [CW87] presented a polynomial-time (centralized) algorithm for constructing a broadcast schedule which uses $O(D\log^2 n)$ time-slots. This centralized algorithm can be implemented in a distributed system assuming the availability of special control channels, but the number of control messages sent may be quadratic in the number of nodes of the network [W87].

Finally, it is interesting to note that Bar-Yehuda, Israeli and Itai, building on the ideas presented in our protocol, have developed efficient protocols for broadcasting multiple messages and point-to-point routing of messages in multi-hop radio networks [BII89].

Organization

In Section 2 we present our randomized broadcast protocol. In Section 3 we prove a linear lower bound on the deterministic time complexity of broadcast. Our conclusions appear in Section 4.

2. RANDOMIZED PROTOCOLS

Throughout this section, n denotes the actual number of processors in the network, N denotes an a priori known upper bound on n, and Δ an a priori known upper bound on the maximum degree in the network (both bounds are a priori known to the source).

The basis for all our protocols is a randomized procedure, called *Decay*, which resolves conflicts among the transmitting neighbors of a receiver by randomly eliminating half of them at each time-slot.

2.1. The Basic Transmission Protocol – Decay

The intuition behind the *Decay* procedure is as follows: A processor receives a message in a certain time-slot if and only if exactly one of its neighbors acts as a transmitter during this time-slot. Thus, in order to guarantee that a message is received, one must coordinate the neighbors so that exactly one of them transmits. As we will see in Section 3, coordinating neighbors by deterministic means is highly inefficient, since the "coordination channels" are subject to exactly the same difficulties. Thus, we abandon the desire of achieving deterministic coordination, and turn for help to randomization procedures. Suppose $d \le \Delta$ processors compete for a time-slot in which exactly one of them sends a message. Simultaneously, they all start a game of coin flips. At each time-slot, on the average half of the remaining processors remove their candidacy. We will show that, with constant probability, before all processors remove their candidacy there exists a time-slot with exactly one candidate.

We now present a precise description of the procedure as executed by each processor.

```
procedure Decay $(k, m)$;
repeat at most $k$ times (but at least once!)
        send $m$ to all neighbors;
        set $coin \leftarrow 0$ or 1 with equal probability.
 until $coin = 0$.
```

By using elementary probabilistic arguments, we get

Theorem 1: Let y be a vertex of G. Also let $d \ge 2$ neighbors of y execute *Decay* during the time interval $[0, k)$ and they all start the execution at $Time=0$. Then $P(k,d)$, the probability that y receives a message by $Time=k$, satisfies:

(i) $\lim_{k \to \infty} P(k,d) \ge 2/3$;

(ii) for $k \ge 2\lceil \log d \rceil$, $P(k,d) > 1/2$.

Convention: Throughout the paper all logarithms are to base 2.

Proof: Clearly, for fixed d, $P(k,d)$ is a nondecreasing function of k; since it is also bounded by 1, it converges. Let $P(\infty,d)$ denote that limit.

(i) Clearly, $P(\infty,0)=0$ and $P(\infty,1)=1$. Also for each $d\geq 2$, we get the recurrence

$$P(\infty,d)=\sum_{i=0}^{d}\binom{d}{i}2^{-d}P(\infty,i)=\sum_{i=1}^{d}\binom{d}{i}2^{-d}P(\infty,i). \qquad (1)$$

We proceed by induction on $d\geq 2$.

<u>*Induction Basis*:</u> By (1),

$$P(\infty,2)=\frac{2\cdot\frac{1}{4}}{1-\frac{1}{4}}=\tfrac{2}{3}.$$

<u>*Induction Step*:</u> Let $d>2$, and assume that $P(\infty,i)\geq\frac{2}{3}$, for all $i<d$. By (1), we get

$$P(\infty,d)\cdot(2^d-1)=\binom{d}{1}P(\infty,1)+\sum_{i=2}^{d-1}\binom{d}{i}P(\infty,i).$$

By the induction hypothesis and $P(\infty,1)=1$,

$$P(\infty,d)\cdot(2^d-1)\geq 1\cdot\binom{d}{1}+\tfrac{2}{3}\sum_{i=2}^{d-1}\binom{d}{i}>\tfrac{2}{3}\cdot(2^d-1) \qquad [d>2]$$

and (i) follows.

(ii) <u>*Case* $d\leq 5$:</u> by inspection.

<u>*Case* $d\geq 6$:</u> Consider runs of the procedure *Decay* without a time bound (i.e., $k=\infty$). Let $T_{t,d}$ be a Boolean random variable assigned *True* if and only if all the neighbors of y terminate by Time t; and R_d be a Boolean random variable assigned *True* if and only if y received a message at finite time.

$$\begin{aligned}
P(k,d) &\geq Pr(R_d\wedge T_{k,d})\\
&=1-Pr(\overline{R_d})-Pr(\overline{T_{k,d}})\\
&\geq P(\infty,d)-d\cdot 2^{-k}\\
&\geq \tfrac{2}{3}-d\cdot d^{-2} && [\text{by (i) and } k\geq 2\log d]\\
&\geq \tfrac{1}{2} && [d\geq 6]
\end{aligned}$$

□

The expected time of the algorithm depends on the probability that *coin* = 0. Here, this probability is set to be one half. An analysis of the merits of using other probabilities was carried out by Hofri [H87].

2.2. The Broadcast Protocol

The broadcast protocol makes several calls to $Decay(k,m)$. In order to obtain the desired probability of Theorem 1 (ii), the parameter k should be at least $2\log d$, where d is the number of neighbors sending a message to a node. Since d is not known, we choose $k = 2\lceil \log \Delta \rceil$ (recall that Δ was defined to be an upper bound on the indegree). Theorem 1 also requires that all participants start executing *Decay* at the same time-slot. Therefore, we start *Decay* only at integer multiples of $2\lceil \log \Delta \rceil$ (i.e., we synchronize the initialization of the various versions of *Decay*).

```
procedure Broadcast;
k := 2⌈logΔ⌉ ;
t := 2⌈log(N/ε)⌉ ;
Wait until receiving a message, say m;
do t times
    Wait until (Time mod k) = 0 ;
    Decay(k,m) ;
od
```

A network is said to execute the *Broadcast_scheme* if some processor, denoted s, transmits an initial message and each processor executes the above *Broadcast* procedure[2]. The following lemma demonstrates the effectiveness of *Broadcast_scheme*, albeit in a crude way. It states that, with very high probability, the communication activity in the network does not die out before all processors receive the message.

2) We distinguish between *Broadcast* which is a program to be executed by each processor and the *Broadcast_scheme* which is a distributed protocol augmented by an initialization assumption (namely that a single processor initiates the execution of the protocol by sending a single message).

Lemma 2: If a network executes *Broadcast_scheme* then:

$$Pr(\text{ All nodes receive } m\ ) \geq 1 - \varepsilon.$$

Proof:

Pr (not all nodes received the message m)

$= Pr(\exists v \neq s$ such that v did not receive m and one of v's neighbors received m)

$\leq \sum_{v \neq s} Pr(\ v$ did not receive m and one of v's neighbors received m)

$\leq \sum_{v \neq s} Pr(\ v$ did not receive $m \mid$ one of v's neighbors received m)

$\leq n \cdot (\frac{1}{2})^{\lceil \log(N/\varepsilon) \rceil} \leq n \cdot (\varepsilon/N) \leq \varepsilon.$ □

The above Lemma bounds from below the probability that broadcast is successful in the network, but does not implicitly address the question of when this happens (i.e., after how many time-slots). An obvious upper bound of $O(Dk \cdot \log(n/\varepsilon))$ can be obtained from the proof of the Lemma. A much sharper bound on the number of time-slots required for broadcast is given by Lemma 3 below.

Notation: For $0 < \varepsilon \leq 1$, let

$$M(\varepsilon) = \sqrt{\log \frac{n}{\varepsilon}} \quad \text{and} \quad T(\varepsilon) = 2 \cdot D + 5M(\varepsilon) \cdot Max(\sqrt{D}, M(\varepsilon)) \tag{2}$$

We abbreviate $M(\varepsilon)$ by M and $T(\varepsilon)$ by T.

Lemma 3: Consider an execution of a modified *Broadcast_scheme* in which the main loop is not timed-out after $\lceil \log(n/\varepsilon) \rceil$ repetitions, but rather is executed indefinitely (starting at $Time = 0$). Then for all $0 < \varepsilon \leq 1$, the following hold

(i) Let T_v be a random variable denoting the time by which processor v receives the message m. Then for $\forall v \in V$,

$$Pr(\ T_v > 2\lceil \log\Delta \rceil \cdot T(\varepsilon)\ ) < \frac{\varepsilon}{n}$$

(ii) Let $T_{fin} = \max_v T_v$. Then

$$Pr(\ T_{fin} \leq 2\lceil \log\Delta \rceil \cdot T(\varepsilon)\ ) > 1 - \varepsilon.$$

The bound provided by Lemma 3 contains two additive terms: the first represents the diameter of the network and the second represents delays caused by conflicts (which are rare yet exist).

Proof: Following is a sketch of the proof of Part (i), from which Part (ii) easily follows. Consider a node $v \in V$. Let the random variable $Dist_i$ be the length of a shortest path from the set of nodes which have received the message m at phase i to v (each phase takes $2\lceil \log\Delta \rceil$ time-slots). Since at $Time = 0$ the source has m,

$$Dist_0 \le D \tag{3}$$

From (ii) of Theorem 1 we get

$$Pr\,(Dist_i - Dist_{i+1} = 1 \mid Dist_i \ne 0) \ge \tfrac{1}{2} \tag{4}$$

Now, $Pr\,(\,Dist_{T(\varepsilon)} > 0\,)$ is the probability that v has not received the message m by time $T(\varepsilon)\cdot 2\lceil \log\Delta \rceil$. On the other hand,

$$\begin{aligned} &Pr(\,Dist_{T(\varepsilon)} > 0\,) \\ &\quad = Pr\,(\sum_{i=0}^{T(\varepsilon)-1} (Dist_i - Dist_{i+1}) < Dist_0) \\ &\quad \le Pr\,(\sum_{i=0}^{T(\varepsilon)-1} (Dist_i - Dist_{i+1}) < D)\,. && \text{[by (3)]} \end{aligned}$$

Define a 0-1 random variable $\chi_i = Dist_i - Dist_{i+1}$. By (4), $Pr\,(\chi_i = 1) \ge \tfrac{1}{2}$. Thus, the above expression corresponds to the probability that the sum of such $T(\varepsilon)$ variables does not exceed D. Using the Chernoff bound [ES74, p. 18] this probability is

$$\begin{aligned} &< \exp\left(-(1 - \frac{2D}{T(\varepsilon)})^2 \cdot \frac{T(\varepsilon)}{4} \right) \\ &\le \exp\left(-\frac{25M^2}{4} \cdot \frac{Max\{D, M^2\}}{2D + 5M \cdot Max\{\sqrt{D}, M\}} \right) && \text{[substitute for } T\text{]} \\ &\le \exp\left(-\frac{25M^2}{4} \cdot \frac{1}{2+5} \right) \le 2^{-M^2} \\ &= \frac{\varepsilon}{n} && \text{[substitute for } M\text{]} \end{aligned}$$

This concludes the proof of (i). □

Combining Lemmas 2 and 3, we get

Theorem 4: Let $T = 2D + 5\max\{\sqrt{D}, \sqrt{\log(n/\varepsilon)}\}\cdot\sqrt{\log(n/\varepsilon)}$. Assume that *Broadcast_scheme* starts at $Time = 0$ then, with probability $\geq 1-2\varepsilon$, by time $2\lceil \log \Delta \rceil \cdot T$ all nodes received the message. Furthermore, with probability $\geq 1-2\varepsilon$, all the nodes have terminated by time $2\lceil \log \Delta \rceil \cdot (T + \lceil \log(N/\varepsilon) \rceil)$.

Remark: Theorem 4 remains valid also in the case that Broadcast is initiated by a non-empty set of processors at the same time (i.e., $Time = 0$) with the same initial message. Namely, redefine *Broadcast_scheme* so that at $Time = 0$ a non-empty subset of the processors have received ("from an external source") copies of the same initial message. Then, with probability $\geq 1-2\varepsilon$, all the processors have received a copy of the initial message and terminated by time $2(T + \lceil \log(N/\varepsilon) \rceil)\cdot\log\lceil \Delta \rceil$. In case *Broadcast_scheme* is initiated by a subset of the processors having arbitrary (i.e., not necessarily identical) messages then, with high probability, each processor terminates getting at least one of these messages.

Additional Properties of our Broadcast Protocol:

1) *Simplicity and Fast Local Computation* – In each time slot each processor does a constant amount of local computation.

2) *Message complexity* – Each processor is active for $\lceil \log(N/\varepsilon) \rceil$ consecutive phases and the average number of transmissions per phase ≤ 2. Thus the expected number of transmissions is bounded by $2n\cdot\lceil \log(N/\varepsilon) \rceil$.

3) *Adaptiveness to Changing Topology and Fault Resilience* – Our protocol is resilient to some changes in the topology of the network. For example, edges may be added or deleted at any time, provided that the network of unchanged edges remains connected. This corresponds to fail/stop failure of edges, thus demonstrating the resilence to some non-malicious failures.

4) *Directed Networks* – Our protocol does not use acknowledgements. Thus it may be applied even when the communication links are not symmetric, i.e., the fact that

processor v can transmit to u does not imply that u can transmit to v. (The appropriate network model is, therefore, a directed graph.) In real life this situation occurs, for instance, when v has a stronger transmitter than u.

2.3. Other Applications of Decay

We first describe an application of Decay to *Breadth First Search (BFS)* defined as follows: given a root r, mark all nodes v by integer $dist(r,v)$ denoting the distance from r to v. BFS can be used for the construction of shortest (i.e., ‘‘minimum hop’’) routing paths in the network.

Before presenting our BFS algorithm, let us note that the paths induced by the *Broadcast_scheme* presented above are unlikely to form a BFS tree. This fact is a consequence of the independence of the randomized events corresponding to a successful receipt of a message at various processors neighbouring the same transmitter (or transmitters of equal distance from s). In fact, though the expected phase number in which a processor receives the message sent by s equals twice the distance between the processor and s, the variance in the value of this random variable is non-negligible. We overcome the difficulty by ‘‘slowing down’’ so to force it to progress "layer by layer". We define each phase to be of length $\lceil \log(N/\varepsilon) \rceil$ times the duration of *Decay*. (I.e., each phase takes $2\lceil \log\Delta \rceil \lceil \log(N/\varepsilon) \rceil$ time slots.) The distance from r is equal to the number of phases from the start until the message was first received. This can be done simply by having r send the start time along with the message. Thus, without loss of generality, we can assume that the protocol started at $Time = 0$.

```
procedure $BFS_v$;
$k := 2\lceil \log\Delta \rceil$ ;
Wait until receiving a message, say $m$;
$Distance_v := \lfloor Time/(k\lceil \log(N/\varepsilon) \rceil) \rfloor$ ;
do $\lceil \log(N/\varepsilon) \rceil$ times
    Wait until ($Time$ mod $k\lceil \log(N/\varepsilon) \rceil) = 0$ ;
    $Decay(k,m)$ ;
od
```

If r starts the algorithm at $Time = 0$ and the only transmissions in the network are those of procedure *BFS* then

$$Pr(\forall v \in V \ \ Distance_v = dist(r,v)) \geq 1 - \varepsilon.$$

The proof is identical to that of Lemma 2. Thus, with probability $\geq 1 - \varepsilon$ the number of consecutive time slots required by the BFS algorithm is $2D\lceil \log\Delta \rceil \lceil \log \frac{N}{\varepsilon} \rceil$.

In the preliminary version of this paper [BGI87], we have stated an application of our broadcast scheme to achieve leader election in arbitrary multi-hop radio networks. That protocol can be viewed as an emulation of Willard's protocol, for electing a leader in a single-hop radio network with collision detection [W86], on an arbitrary multi-hop radio network without collision detection. This emulation is in fact independent of the specific protocol and has appeared in [BGI89].

Finally, we note that *Decay* plays a central role in the efficient protocols for the broadcast and point-to-point routing of messages in multi-hop radio networks presented in [BII89].

3. A DETERMINISTIC LOWER BOUND

Before presenting our lower bound, we formally present the problem of broadcast in radio networks.

Definition 1: (broadcast protocols): A *broadcast protocol for radio networks* is a multi-processor protocol the execution of which proceeds in time-slots (numbered $0,1,2,...$) as follows.

1) In the initial time-slot, referred to as *time-slot 0*, a specific processor, called the *source*, transmits a message, called the *initial message* (or *the message*).

2) In each time-slot, including time-slot 0, each processor either acts as a transmitter or acts as a receiver or is inactive.

3) A processor receives a message in a specific time-slot if and only if it acts as a receiver in this time-slot and *exactly* one of its neighbors acts as a transmitter in that time-slot. (The message received in this case is the message transmitted by that neighbor.)

4) The action of a processor in a specific time-slot is determined as a function of its initial input (which consists of its own ID and the IDs of its neighbors), and the (sequence of) messages that it has received in previous time-slots. Thus, without loss of generality, all processors have identical copies of the same program.

5) A processor may act as a transmitter in a time-slot only if it has received a message in a previous time-slot (i.e., there are no ‘‘spontaneous’’ actions). (As we will see in subsection 3.5, this condition can be omitted).

6) The broadcast is *completed* at time-slot t, if all processors have received the initial message at one of the time-slots $0, 1, \cdots, t$.

A broadcast protocol Π for radio networks is *correct* for the class C if for every $G(V,E) \in C$ and any assignment of IDs, ϕ, to the nodes of G there exists an integer t such that Π completes broadcast at time-slot t when executed in the graph G with the ID assignment ϕ.

In our lower bound argument, we consider an *arbitrary* deterministic broadcast protocol and its executions on members of a particular class of networks denoted C_n. Clearly, the lower bound holds for protocols running on any class of networks containing C_n. All networks in C_n have exactly $n+2$ processors, and thus we can think that the protocol gets the number of processors as input.

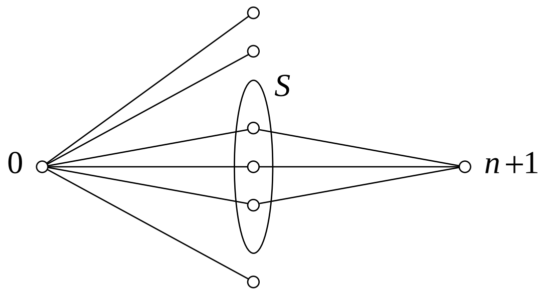
3.1. The Networks Used in our Argument

A (generic) member of C_n will be denoted as G_S, where S is a $\underline{\text{non–empty}}$ subset of $\{1,2,...,n\}$. The processors in this network have IDs denoted 0 through $n+1$, and are associated with nodes 0 through $n+1$. The structure of the network G_S constitutes a graph with vertex-set $\{0,1,2,...,n+1\}$ and edge-set $E_1 \cup E_2$, where

$$E_1 = \{(0,i) : 1 \le i \le n\}$$
$$E_2 = \{(i,n+1) : i \in S\}$$

The nodes are organized in three layers. The first layer consists of node 0, called the *source*. The second layer contains nodes 1 through n, these nodes will be the receivers of the initial transmission. The third layer consists of the last node $n+1$, which is adjacent to the nodes in

S (see figure).



The problem of broadcast, in networks of C_n, thus reduces to reaching the node of the third layer, called the *sink*. The difficulty stems from the fact that the partition of the second layer (i.e., *S*) is not known a priori.

3.2. Reduction to a Combinatorial Game

In this subsection we reduce the problem of broadcast to a simple combinatorial game, called the *hitting game*. The reduction is in three stages. In the first two stages we simplify the problem by restricting and strengthening broadcast protocols in a manner which does not effect the lower bound, while in the third stage an abstract broadcast problem is reformulated as a hitting game.

We first (slightly) restrict the broadcast protocol operating on the networks of the class C_n. This restriction does not change the asymptotic complexity of broadcast in C_n.

Definition 2: (restricted broadcast protocols): A broadcast protocol Π for the class C_n is called *restricted* if, for every graph $G_S \in C_n$ and every time-slot *i*, in the *i*-th time-slot of the execution of Π on G_S either the source is active or the sink is active, but not both.

Lemma 5: If there is a broadcast protocol which terminates within *t* time-slots on every network in C_n, then there exists a restricted broadcast protocol which terminates within $2t$ time-slots on every network in C_n.

The proof is given in Appendix A1.

To further simplify the analysis, we will consider only time-slots in which either the source or the sink acts as a receiver. The effect of the other time-slots will be achieved by assuming that all processors of the second layer have gained knowledge of each transmission received by the source or sink immediately after it has occurred. This and other simplifying assumption are present in the following abstract communication model which certainly is *not intended* to be a real model of radio communication. Nevertheless, we will show that the complexity of broadcast in the model of Definition 2 is bounded below by the complexity of broadcast in the abstract model. Before describing the abstract model we present the following concept which relates to processors in a network $G_S \in C_n$.

Definition 3: The *S-indicator* (the *indicator*) of a second layer processor $p \in \{1,2,...,n\}$, denoted χ_p^S, is a bit which equals 1 if and only if $p \in S$.

Definition 4: (abstract broadcast protocols): An *abstract* broadcast protocol for C_n is a multi-processor protocol which proceeds in rounds (numbered $1,2,...$) as follows.

1) In each round, only processors of the second layer may act as transmitters, and either the source or the sink (but not both) may act as receivers. All messages sent consist merely of the transmitter's ID and its S-indicator. That is, each message transmitted by processor p consists of the pair (p, χ_p^S).

2) A processor (sink or source) receives a message in a specific round if and only if it acts as a receiver in this round and *exactly* one of its neighbors acts as a transmitter in that round. A round is called *successful* if the processor acting as a receiver (i.e., either the source or the sink) has received a message.

3) At the end of the round, all processors of the second layer know whether the round has been successful. Furthermore, in case the round was successful these processor know the contents of the message which was received.

4) The action of a processor in a specific round is determined as a function of its initial input (which consists of its own ID and the IDs of its neighbors), and the sequence of pairs (t,M), where t is a previous successful round and M is the message received in that round.

5) The broadcast is *completed* once the processor indicator in the message received in a successful round equals 1. (I.e., the broadcast is completed in the first round in which a message sent by a processor in S is received.)

It follows that the processor indicator in the messages in all successful rounds preceding the last one is 0.

Lemma 6: If there is a restricted broadcast protocol which terminates within t time-slots on every network in C_n, then there exists an abstract broadcast protocol which terminates within t rounds on every network in C_n.

The proof is given in Appendix A2.

Notation: Let $\overline{S}$ denote the set $\{1,2,...,n\}-S$.

Let us now take a closer look at the execution of the abstract protocol Π. Let H_{i-1} be the common knowledge of the history rounds 1 through $i-1$. This history consists of the sequence of successful rounds and the corresponding successful transmitters. Namely $H_{i-1}=P_1,P_2,...,P_{i-1}$, where P_k is a special symbol (say -1) in case round k is not successful, and otherwise P_k is the ID of the processor the message of which is received in round k.

Processor $p \in \{1,2,...,n\}$ decides whether to transmit in round i as a function of its initial input (which in turn is determined by its ID, p, and its S-indicator χ_p^S) and the history H_{i-1}. Let us denote this predicate by π; namely, the p-th processor acts as transmitter in round i if and only if $\pi(p,\chi_p^S,H_{i-1})=1$. Without loss of generality, $\pi: \{1,2,...,n\}\times\{0,1\}\times\{-1,1,2,...,n\}^* \rightarrow \{0,1\}$.

The set of transmitters in round i is denoted by T_i. The following equalities are easily verified.

$$\begin{aligned} T_i &= \{p : \pi(p,\chi_p^S,H_{i-1})=1\} \\ &= \{p\in S : \pi(p,1,H_{i-1})=1\} \cup \{p\in \overline{S} : \pi(p,0,H_{i-1})=1\} \\ &= \left(\{p:\pi(p,1,H_{i-1})=1\}\cap S\right) \cup \left(\{p:\pi(p,0,H_{i-1})=1\}\cap \overline{S}\right) \end{aligned}$$

Let $T_i^{(\sigma)} = \{p:\pi(p,\sigma,H_{i-1})=1\}$, for $\sigma \in \{0,1\}$. Then

$$T_i = (T_i^{(1)} \cap S) \cup (T_i^{(0)} \cap \overline{S})$$

Recall that round i is successful if and only if $|T_i \cap S| = 1$. In formulating the following combinatorial game, which captures the structure of abstract broadcast protocols, we will use a relaxed condition.

Definition 5: (The n-th *hitting game*): The n-th *hitting game* is a combinatorial game played by two parties, called the *explorer* and the *referee*. The game is played on a non-empty set $S \subseteq \{1,2,...,n\}$, known only to the referee. The explorer's task is to "hit" an element of S. The game proceeds in moves. In the i-th move the explorer, based on the consequence of his previous moves, specifies a set M_i. If $M_i \cap S$ is a singleton then the referee reveals it to the explorer, and the game is terminated. If $M_i \cap \overline{S}$ is a singleton then the referee reveals it to the explorer without terminating the game. Otherwise (i.e., both $M_i \cap S$ and $M_i \cap \overline{S}$ are not singletons) the referee says ‘‘nothing’’. We stress that the actions of the referee are completely determined by the explorer's moves and the set S. We say that the explorer *won the game in t moves* if the game was terminated at the t-th move (when the referee handed an element of S to the explorer). We say that the explorer has a *t-move winning strategy* if, no matter what S is, the explorer wins within t moves.

Remark: An explorer strategy determines each move of the explorer as a function of the current history of the game. In fact, it suffices to consider the consequences of (i.e., referee's answer to) the previous moves of the explorer.

Lemma 7: If there is an abstract broadcast protocol which terminates within t rounds on every network in C_n, then there exists a $2t$-move winning strategy for the n-th hitting game.

The proof is given in Appendix A3.

Combining Lemma 5, 6 and 7, we get

Proposition 8: Let $T(n)$ be the deterministic time-complexity of broadcast on networks in C_n, and let $G(n)$ be the number of steps required to win the n-th hitting game. Then

$$T(n) \geq \frac{1}{4} \cdot G(n)$$

Remark: A more careful reduction yields $T(n) \geq G(n)/2$. The essential ideas appear in Appendix A4.

3.3. A Lower Bound on Hitting Games

In this subsection we prove a linear lower bound on the number of moves required to win the n-th hitting game. We do this by presenting an ‘‘adversary’’ procedure for determining, for each explorer strategy of $n/2$ moves, a (non-empty) S which foils this strategy. Furthermore, we will show that for every strategy of less than $n/2$ moves there exists a set S so that the referee answers $\emptyset$ to all non-singleton moves. Clearly, the referee answers all singleton moves with the singleton itself, and it goes without saying that these moves are not in S. We stress that for such a set S, the referee’s answers are determine solely by the explorer’s moves, and thus the explorer gains no information from these answers. Hence, the problem of finding sets which foil all explorer strategies reduces to the problem of finding sets which foil all (‘‘oblivious’’) strategies which do not depend on the referee’s answers to the previous moves.

We start by constructing a set S which foils an *oblivious* strategy (for the explorer). An arbitrary oblivious strategy consists of a fixed sequence of moves. Given a sequence of t $(\leq n/2)$ moves $M_1, M_2, ..., M_t$ we construct a (non-empty) set S that contains no singleton moves and for all non-singleton moves M_i both $|M_i \cap S| \neq 1$ and $|M_i \cap \overline{S}| \neq 1$. An equivalent condition is that for all i, the set $M_i \cap S$ is not a singleton and the set $M_i \cap \overline{S}$ is a singleton iff M_i is a singleton.

The construction of the set S proceeds as follows. We start with $S = \{1, 2, ..., n\}$. First we omit all singleton moves from S, and their elements from all other moves. New *residual* moves are created. If any of them is a singleton it is omitted from S too (and residual moves are updated again). This process guarantees that no move has a singleton intersection with S. Recall, that we need also to guarantee that except for singleton moves, no other moves have singleton intersection with $\overline{S}$. To this end, we remove another element from S each time a non-singleton move is updated for the first time (guaranteeing that the intersection of this move with $\overline{S}$ has cardinality ≥ 2).

```
procedure find_set;
input: $M_1, M_2, ..., M_t$;
```

```
initialization: $S \leftarrow \{1,2,\ldots,n\}$;
while ($\exists i$ s.t. $|M_i \cap S| = 1$) do begin
    let $x$ denote the single element of $M_i \cap S$;
    set $S \leftarrow S - \{x\}$;
    while ($\exists j$ s.t. $|M_j \cap S| = |M_j| - 1 > 0$)
    do begin
        pick (arbitrary) $p \in M_j \cap S$;
        set $S \leftarrow S - \{p\}$;
    end;
end;
output: $S$;
```

First, we show that if the procedure has output a non-empty S then S is consistent with the input moves. Namely,

Lemma 9: Let the M_i's be as in the procedure, and suppose that the procedure outputs S. Then, for every i ($1 \le i \le t$):

1) The set $M_i \cap S$ is not a singleton.

2) $M_i \cap \overline{S}$ is a singleton if and only if M_i is a singleton.

Proof: We consider two cases:

Case 1: $M_i \cap S$ is empty. Condition (1) holds trivially. To see that condition (2) holds note that in this case $M_i \subseteq \overline{S}$ and $|M_i \cap \overline{S}| = |M_i|$ follows.

Case 2: $M_i \cap S$ is not empty. By the condition of the outer **while** loop, $|M_i \cap S| \neq 1$ and condition (1) holds. Since $|M_i| \ge |M_i \cap S| \notin \{0,1\}$, condition (2) requires showing that $|M_i \cap \overline{S}| \neq 1$. We consider two subcases.

Subcase 2.1: $M_i = M_i \cap S$. In this subcase, $M_i \cap \overline{S} = \emptyset$ and the claim follows .

Subcase 2.2: $M_i \neq M_i \cap S$. In this subcase, $1 \le |M_i \cap \overline{S}| = |M_i - (M_i \cap S)|$. By the condition of the inner **while** loop, $|M_i \cap S| \neq |M_i| - 1$ and therefore $|M_i \cap \overline{S}| \neq 1$. □.

Next, we show that the procedure terminates outputting a non-empty S. Namely,

Lemma 10: If $t \leq n/2$ then the above procedure outputs $S \neq \emptyset$.

Proof: We prove the lemma by considering the decrease in $|S|$ throughout the execution of the procedure. Elements are omitted from S in two cases:

1) When $M_i \cap S$ becomes a singleton, then during the execution of the outer **while** loop, it is omitted from S. In this case we charge this element to move i.

2) When $M_j \cap S$ first decreases, then during the execution of the inner **while** loop, one of its elements is omitted from S. In this case we charge this element to move j.

The above charging rule certainly satisfies the following two claims:

Claim 1: Each element omitted from S is charged to some move. (Proof: by definition of the charging rule.)

Claim 2: Each move is charged at most twice. Furthermore, a singleton move is charged exactly once and a (non-empty) non-singleton move is charged at most twice. (Proof: each move is charged at most once by case (1) and at most once by case (2).)

This gives a bound of $2t$ on the number of elements omitted (and charged). To get a slightly sharper bound note that either there are no singleton moves (and then no elements are omitted) or there exists a singleton move (which is of course charged exactly once). The total charge is thus $2(t-1)+1 \leq n-1$, and $S \neq \emptyset$. □

The lower bound on the hitting game now follows. Given an explorer strategy, we consider the moves it induces supposing that all previous non-singleton moves were answered $\emptyset$. Combining Lemma 9, and 10, we get

Proposition 11: Let $G(n)$ be the number of steps required to win the n-th hitting game. Then

$$G(n) > \frac{n}{2}$$

3.4. Summary

Combining Propositions 8 and 11, we get

Theorem 12: There exists no deterministic broadcast protocol which terminates in less than $n/8$ time-slots on any network in C_n.

We have proved a $\Omega(n)$ lower bound on the time-complexity of deterministic radio broadcast (on arbitrary networks of n processors). This bound is tight, as it is easy to see that one may reach all n processors in a network within $2n$ time-slots, by having the current transmitter traverse the network in a Depth-First-Search manner. On the other hand, the gap between the deterministic and randomized time-complexity of radio broadcast is striking, as we have

Corollary 13: There exists a family of n-processor networks for which the (constant-error) randomized time-complexity of radio broadcast is $O(\log n)$, whereas the deterministic time-complexity is $\Omega(n)$.

Proof: Consider the family C_{n-2} defined above. The deterministic lower bound is by Theorem 12. Using the protocol of Section 2, for the randomized upper bound, the Corollary follows. □

3.5. Extension to Spontaneous Transmission

Throughout the entire section we have assumed that, except for the source, no processor transmits before receiving a message. If this assumption does not hold there exist a three round broadcast protocol for the network class C_n. In round 0 the source transmits as usual, in round 1 the sink spontaneously "awakes" and transmits the smallest among its neighbors ID, and in round 2 this processor transmits and the broadcast is completed. Fortunately, a slightly more complicated network class admits a lower bound similar to the one proven in Theorem 12.

The new network class, denoted C_n^*, consists of graphs denoted as $G_{S,R}$, where S and R are $\underline{\text{non-empty}}$ subset of $\{1,2,...,n\}$ and $\{n+1,n+2,...,2n\}$, respectively. The network consists of $2n+1$ processors having IDs denoted 0 through $2n$. The structure of the network $G_{S,R}$ constitutes a graph with vertex-set $\{0,1,2,...,2n\}$ and edge-set $E_1 \cup E_2$, where

$$E_1 = \{(0,i) : 1 \le i \le n\}$$

$$E_2 = \{(i,j) : i \in S\,,\ j \in R\,\}$$

As before, the nodes are organized in three layers. Node 0 is called the *source*, and the nodes R are called the sinks. The problem of broadcast, in C_n^*, consists of reaching all sinks (which is as difficult as reaching one of them!). An alternative formulation defines broadcast as completed once a message is received through any of the links in E_2. The reader may easily verify that the arguments we have used still apply with respect to such a transmission being in any of the two possible directions.

4. CONCLUDING REMARKS

We have shown how conflicts, arising in broadcast protocols, can be resolved quickly by using randomization. This point is further pursuit in our emulation of single-hop radio network with collision detection on multi-hop radio networks without collision detection [BGI89].

The exponential gap between the deterministic and randomized complexities in this model, is believed to be another strong indication to the importance of randomization for distributed applications.

Collision Detection: Sometimes it is reasonable to assume that a processor can detect collisions: i.e, distinguish between the case that zero or more than one neighbor transmits. Our randomized protocol achieve almost optimal behavior without resorting to collision detection. However, our lower bound on deterministic protocols no longer holds. In particular, one can broadcast in C_n using 4 time-slots. An interesting open problem is to find matching lower and upper bounds for deterministic broadcast protocols which use collision detection.

ACKNOWLEDGEMENTS

It is a pleasure to thank Shay Kutten, Nati Linial and Avi Wigderson for many helpful discussions. We also wish to thank the referees for their useful comments.

APPENDICES to Subsection 3.2

Appendix A1: Proof of Lemma 5

Let Π be a broadcast protocol such as in the hypothesis. We construct a restricted broadcast protocol, Π', which simulates Π as follows. The i-th time-slot of Π is simulated by the $2i-1$-st and $2i$-th time-slots of Π'. In the $2i-1$-st time-slot of Π', the sink is inactive while all other processors (i.e., the source and the processors of the second layer) act as in the i-th time-slot of Π. In the $2i$-th time-slot of Π', the source is inactive while all other processors (i.e., the sink and the processors of the second layer) act as in the i-th time-slot of Π. After the $2i$-th time-slot each processor considers the messages it has received in the $2i-1$-st and $2i$-th time-slots. If it has received messages in both time-slots, the processor ignores these messages and records that it has received no messages in simulating the i-th time-slot. (This may occur only for processors in the set S.) Otherwise, the processor records the message (possibly none!) it has received as the message received in the simulation of the i-th time-slot. Clearly, the message (possibly none) recorded by each processor after time-slot $2i$ equals the message received by the very same processor in the i-th time-slot of the execution of Π. Thus, Π' completes broadcast within $2t$ time-slots on every network in C_n, and the Lemma follows.

Appendix A2: Proof of Lemma 6

Let Π be a restricted broadcast protocol such as in the hypothesis. Recall that without loss of generality all processors have identical copies of the same local program. It follows that, without loss of generality, all messages sent by a processor $p \in \{1,2,...,n\}$ contain only the pair (p, χ_p^S) and the sequence of all messages received by processor p in previous rounds.

We now construct an abstract broadcast protocol, Π', which simulates the above protocol Π. The i-th time-slot of Π is simulated by the i-th round of Π' as follows. The processors of the second layer which are active as transmitters in the i-th time-slot of Π are active as transmitters in the i-th round of Π'. If either the source or the sink is active as receiver in the i-th time-slot of Π then it is active as receiver in the i-th round of Π'. In all other cases (a processor of the second layer which does not act as transmitter or the sink or source which is not active as receiver) the processor is inactive. The messages transmitted contain only the transmitter's ID and indicator and in case of success the transmitter's ID is immediately known to all processors of the second layer.

We need to verify that the processors of the abstract protocol have the knowledge required to simulate the corresponding processors in the restricted protocol Π. First note that by the termination condition, the sink does not transmit during the execution of Π, since the sink may transmit only after receiving a message (and at this point Π terminates). Since Π must work for all initial messages we may consider its execution with some standard message, which may be incorporated into the protocol. Thus, there is no need to send the standard message and the protocol Π terminates when some message reach the sink. Also, the first transmission in Π (i.e., time-slot 0 in which the source transmits), does not add any information and may be omitted. Omitting all the other transmissions of the source also does not decrease the information available to the processors of the second layer. This is the case since after the i-th time-slot the source only knows its initial input (which is a priori known to all processors), and the list of all previous successful rounds and the corresponding transmitters (which is known to also to all processors of the second layer - by definition of the abstract model). A similar argument shows that no information is lost when omitting from the messages of the processors of the second layer everything but the ID of the transmitter and its indicator.

The Lemma follows by noting that if Π terminates in the t-th time-slot then Π' terminates in the t-th round.

Appendix A3: Proof of Lemma 7

Let Π be an abstract broadcast protocol as in the hypothesis, and π be the corresponding predicate determining whether to transmit or not. We construct a 2*t*-move winning strategy for the *n*-th hitting game as follows. The *i*-th round of Π is simulated by the $2i-1$-st and 2*i*-th moves of the game.

For $M \subseteq \{1,2,...,n\}$, let $ref_S(M)$ denotes the referee's answer to the move *M*. That is, $ref_S(M)$ equals $M \cap S$ if $M \cap S$ is a singleton, and equals the empty set (denoted $\emptyset$) otherwise. The first move of the game consists of the set $T_1^{(1)}$ (= $\{p : \pi(p, 1, \emptyset) = 1\}$) and the second move consists of the set $T_1^{(0)}$ (= $\{p : \pi(p, 0, \emptyset) = 1\}$). If either set has a singleton intersection with *S* then the game is terminated. Otherwise, let $R_1 \leftarrow g(ref_S(T_1^{(1)}), ref_S(T_1^{(0)}))$, where $g(A,B)$ equals p if $\{p\} = A \cup B$ and equals -1 if $|A \cup B| \neq 1$. For $\sigma \in \{0,1\}$, the $2i-\sigma$-th move is the set $T_i^{(\sigma)} = \{p : \pi(p, \sigma, R_{i-1}) = 1\}$. Note that the explorer can compute his moves! If either (the $2i-1$-st or 2*i*-th) moves has a singleton intersection with *S* then the game is terminated. Otherwise, let $R_i \leftarrow R_{i-1}, g(ref_S(T_1^{(1)}), ref_S(T_1^{(1)}))$, where *g* is as above.

For every *i*, if the game is not terminated within 2*i* moves, when playing according to the above strategy on the set *S*, then the sequence R_i computed by the explorer corresponds to the history sequence H_i in the execution of the protocol Π on the graph G_S. If the protocol Π is completed at the *i*-th round, when executed on the graph G_S, then $|T_i \cap S| = 1$ and furthermore $|T_i^{(1)} \cap S| = 1$. It follows that, when playing according to the above strategy on the set *S*, the game terminates no later than after the $2i-1$-st move. Thus, the above strategy constitutes a $2t-1$ moves winning strategy for the *n*-th hitting game, and the lemma follows.

Appendix A4: Essential Ideas for a more Careful Reduction

This appendix refers to the reduction of the time-complexity of broadcast to the hitting game. The reduction presented in section 3.2 yields $T(n) \geq G(n)/4$. A careful modification of that reduction yields $T(n) \geq G(n)/2$. Following are the essential ideas:

1) In the restricted broadcast protocol generated by the proof of Lemma 5, the source is active only in odd time-slots, while the sink is active only during even time-slots. Furthermore, the same processors of the second layer are active in the $2i-1$-st and 2*i*-th

time slots, for every i.

2) One can modify the abstract broadcast protocol produced by the proof of Lemma 6 so that all processors in $\overline{S}$ are inactive during each even round, while all processors in S are inactive during each odd round. (The first modification is obvious since processors in $\overline{S}$ transmitting during an even round have no effect. For the second modification note that if the message sent by a processor in S is received by the source in round $2i-1$ then it will be received by the sink in round $2i$, and therefore cancelling the first transmission only delays termination by 1 round.)

3) It follows that $T_i^{(0)}$ is empty for even i, while $T_i^{(1)}$ is empty for odd i. Using this fact, Lemma 7 can be strengthened to yield that an abstract broadcast protocol which terminates within t rounds on every network in C_n implies a t-move winning strategy for the n-th hitting game. In the proof of the modified Lemma, let the i-th move of the explorer consist of $T_i^{(0)}$ for odd i and $T_i^{(1)}$ for even i.

REFERENCES

[A70] Abramson, N., "The Aloha system – Another approach for computer communications", in Proc. Fall Joint Computer Conf., AFIPS Press, Vol. 37, (1970).

[BGI87] R. Bar-Yehuda, O. Goldreich, A. Itai, ‘‘On the Time-Complexity of Broadcast in Radio Networks: An Exponential Gap Between Determinism and Randomization’’, *6th ACM Symp. on Principles of Distributed Computing (PODC)*, 1987, pp. 98-108.

[BGI89] R. Bar-Yehuda, O. Goldreich, A. Itai, "Efficient Emulation of Single-Hop Radio network with Collision Detection on Multi-Hop Radio network without Collision Detection", *3rd International Workshop*, Nice, France, (proceedings), Lecture Notes in Computer Science, Vol. 392, Springer Verlag, 1989, pp. 24-32.

[BII89] R. Bar-Yehuda, A. Israeli, A. Itai, "Multiple Communication in Multi-Hop Radio Networks", *8th ACM Symp. on Principles of Distributed Computing (PODC)*, 1989, pp. 329-338, to appear in SIAM Journal on Computing.

[C79] Capetanakis, J., "Generalized TDMA: The multi-accessing tree protocol", *IEEE Trans. Commun.*, Vol. COM-27, 1479-1484, (1979).

[CK85] Chlamtac, I. and Kutten, S., "On Broadcasting in Radio Networks- Problem Analysis and Protocol Design", *IEEE Transactions on Communications,* December (1985), Vol COM-33, No. 12.

[CW87] Chlamtac, I. and Weinstein O., "The wave expansion approach to broadcasting in multihop radio networks", *INFOCOM* (April 1987).

[DIX80] Digital-Intel-Xerox, "The Ethernet data link layer and physical layer specification 1.0" (Sept. 1980).

[ES74] Erdos, P., and J. Spencer, *Probabilistic Methods in Combinatorics*, Academic Press, (1974).

[EGMT] Even, S., Goldreich, O., Moran S. and Tong, P., "On the NP Completeness of Certain Network Testing Problems", *Networks*, Vol. 14, (1984), 1-24.

[G85] Gallager, R., "A perspective on multiaccess channels", *IEEE Trans. on Inf. Theory* special issue on Random Access Communications, Vol. IT-31 (1985), 124-142.

[GVF76] Gitman, I., Van Slyke, R.M. and Frank, H., "Routing in Packet-Switching Broadcast Radio Networks", *IEEE Transactions on Communications,* (August 1976), 926-930.

[GL83] Greenberg, A.G., and Ladner, R.E., "Estimating the Multiplicity of Conflicts in Multi-access Channels", *24th FOCS*, 1983, pp. 383-392.

[GW85] Greenberg, A.G., and Winograd, S., "A Lower Bound on the Time Needed in Worst Case to Resolve Conflicts Deterministically in Multi-access Channels", *JACM*, Vol. 32, No. 3, July 1985, pp. 589-596.

[H78] Hayes, J.F, "An adaptive technique for local distribution", *IEEE Trans. on Commun.*, Vol. COM-26, (1978), 1178-1186.

[H87] Hofri, M., "A Feedback-less Distributed Broadcast Algorithm for Multihop Radio Networks with Time-varying Structure", *2nd ACM Intr. MCPR Workshop*, Rome (May 1987). Also available as TR-451, Computer Science Dept., Technion, Haifa, Israel (March 1987).

[KG85] Komlós, J. and Greenberg A.G., "An asymptotically nonadaptive algorithm for conflict resolution in multiple-access channels", *IEEE Trans. on Inf. Theory*, Vol. IT-31 (1985), 302-306.

[R76] M.O. Rabin, "Probabilistic algorithms", Proc. Symp. on New Directions and Recent Results in Algorithms and Complexity, J.F. Traub ed., Academic Press, (1976).

[R72] L.G. Roberts, "Aloha packet system with and without slots and capture", ASS Notes 8, Advanced Research Projects Agency, Network Information Center, Stanford Research Institute, Stanford, June 1972.

[T81] A.C. Tanenbaum, *Computer Networks*, Prentice-Hall, Inc., 1981.

[T85] Tsybakov, B.S., "Survey on USSR contribution to Random Multi-access communications", *IEEE Trans. on Inf. Theory* special issue on Random Access Communications, Vol. IT-31 (1985), 143-165.

[TM79] Tsybakov, B.S. and Mikhailov, V.A., "Free synchronous packet access in a broadcast channel with feedback", *Prob. Inform. Th.*, Vol. 14, 259-280, (1979).

[W87] Weinstein O., "The wave expansion approach to broadcasting in multihop radio networks", M.Sc. Thesis, Computer Science Dept., Technion, Haifa, Israel, (1987).

[W86] Willard D.E., "Log-logarithmic selection resolution protocols in a multiple access channels", SIAM J. on Comput. Vol. 15, 468-477, (1986).

[W85] Wolf, J.K., "Born Again Group Testing: Multi-access communications", *IEEE Trans. on Inf. Theory* special issue on Random Access Communications, Vol. IT-31 (1985), 185-191.