

Randomized Broadcasting in Radio Networks, 1992; Reuven Bar-Yehuda, Oded Goldreich, Alon Itai

Entry editor: Alon Itai

Department of Computer Science, Technion - IIT, Haifa, Israel

email: itai@cs.technion.ac.il

URL: <http://www.cs.technion.ac.il/~itai/>

September 13, 2007

SYNONYMS: Multi-hop radio networks, ad hoc networks.

INDEX TERMS: Radio broadcast, randomized distributed algorithms, collision detection.

1 PROBLEM DEFINITION

The paper investigates deterministic and randomized protocols for achieving broadcast (distributing a message from a source to all other nodes) in arbitrary multi-hop synchronous radio networks.

The model consists of an arbitrary (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 that time-slot t if exactly one of its neighbors transmits in that time-slot. The message received is the one transmitted. If more than one neighbor transmits in that time-slot, a *conflict* occurs. In this case the receiver may either get a message from one of the transmitting neighbors or get no message. It is assumed that conflicts (or “collisions”) are not detected, hence a processor cannot distinguish the case in which no neighbor transmits from the case in which one or more of its neighbors transmits during that time-slot. The processors are not required to have ID’s nor do they know their neighbors, in particular the processors do not know the topology of the network.

The only inputs required by the protocol are the number of processors in the network – n , Δ – an a priori known upper bound on the maximum degree in the network and the error bound – ϵ . (All bounds are a priori known to the algorithm.)

Broadcast is a task initiated by a single processor, called the *source*, transmitting a single *message*. The goal is to have the message reach all processors in the network.

2 KEY RESULTS

The main result is a randomized protocol that achieves broadcast in time which is optimal up to a logarithmic factor. In particular, with probability $1 - \epsilon$, the protocol achieves broadcast within $O((D + \log n/\epsilon) \cdot \log n)$ time-slots.

On the other hand, a linear lower bound on the deterministic time-complexity of broadcast is proved. Namely, any deterministic broadcast protocol requires $\Omega(n)$ 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.

2.1 Randomized Protocols

2.1.1 The procedure *Decay*

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. It will be shown that if all neighbors of a receiver follow the elimination procedure then with positive probability there exists a time slot in which exactly one neighbor transmits.

What follows is a description of the procedure for sending a message m , that is executed by each processor after receiving m :

```
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, one can prove:

Theorem 2.1 *Let y be a vertex of G . Also let $d \geq 2$ neighbors of y execute *Decay* during the time interval $[0, k)$ and assume that they all start the execution at Time = 0. Then $P(k, d)$, the probability that y receives a message by Time = k , satisfies:*

- (i) $\lim_{k \rightarrow \infty} P(k, d) \geq \frac{2}{3}$;
- (ii) for $k \geq 2\lceil \log d \rceil$, $P(k, d) > \frac{1}{2}$.

All logarithms are to base 2.

The expected termination 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 [4].

2.1.2 The Broadcast Protocol

The broadcast protocol makes several calls to $Decay(k, m)$. By Theorem 2.1 (ii), to ensure that the probability of a processor y receiving the message be at least $1/2$, the parameter k should be at least $2 \log d$ (where d is the number of neighbors sending a message to y). Since d is not known, it was chosen as $k = 2 \lceil \log \Delta \rceil$ (recall that Δ was defined to be an upper bound on the in-degree). Theorem 2.1 also requires that all participants start executing $Decay$ at the same time-slot. Therefore, $Decay$ is initiated only at integer multiples of $2 \lceil \log \Delta \rceil$.

```

procedure Broadcast;
   $k = 2 \lceil \log \Delta \rceil$ ;
   $t = 2 \lceil \log(N/\epsilon) \rceil$ ;
  Wait until receiving a message, say  $m$ ;
  do  $t$  times {
    Wait until  $(Time \bmod k) = 0$  ;
     $Decay(k, m)$  ;
  }

```

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.

Theorem 2.2 *Let $T = 2D + 5 \max\{\sqrt{D}, \sqrt{\log(n/\epsilon)}\} \cdot \sqrt{\log(n/\epsilon)}$. Assume that *Broadcast_scheme* starts at $Time = 0$. Then, with probability $\geq 1 - 2\epsilon$, by time $2 \lceil \log \Delta \rceil \cdot T$ all nodes will receive the message. Furthermore, with probability $\geq 1 - 2\epsilon$, all the nodes will terminate by time $2 \lceil \log \Delta \rceil \cdot (T + \lceil \log(N/\epsilon) \rceil)$.*

The bound provided by Theorem 2.2 contains two additive terms: the first represents the diameter of the network and the second represents delays caused by conflicts (which are rare, yet they exist).

2.1.3 Additional Properties of the Broadcast Protocol:

- **Processor IDs** – The 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 the protocol adaptive to changes in topology which occur throughout the execution, and resilient to non-malicious faults.
- **Knowing the size of the network** – The 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).
- **Conflict detection** – The algorithm and its complexity remain valid even if no messages can be received when a conflict occurs.

- **Simplicity and fast local computation** – In each time slot each processor performs a constant amount of local computation.
- **Message complexity** – Each processor is active for $\lceil \log(N/\epsilon) \rceil$ consecutive phases and the average number of transmissions per phase is at most 2. Thus the expected number of transmissions of the entire network is bounded by $2n \cdot \lceil \log(N/\epsilon) \rceil$.
- **Adaptiveness to changing topology and fault resilience** – The 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 resilience to some non-malicious failures.
- **Directed networks** – The 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 .

2.2 A Lower Bound on Deterministic Algorithms

For deterministic algorithms one can show a lower bound: for every n there exist a family of n -node networks such that every deterministic broadcast scheme requires $\Omega(n)$ time. For every **non-empty** subset $S \subseteq \{1, 2, \dots, n\}$, consider the following network G_S (Figure 1).

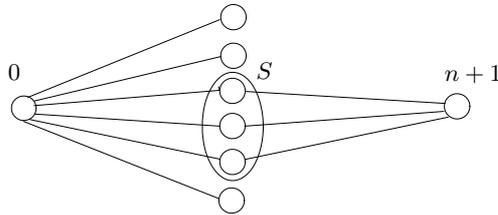


Figure 1: The network used for the lower bound

Node 0 is the *source* and node $n + 1$ the *sink*. The source initiates the message and the problem of broadcast in G_S is to reach the sink. The difficulty stems from the fact that the partition of the middle layer (i.e., S) is not known a priori. The following theorem can be proved by a series of reductions to a certain “hitting game”:

Theorem 2.3 *Every deterministic broadcast protocol that is correct for all n -node networks requires time $\Omega(n)$.*

In [2] there was some confusion concerning the broadcast model. In that paper it was erroneously claimed that the lower bound holds also when a collision is indistinguishable from the absence of transmission. Kowalski and Pelc [5] disproved this claim by showing how to broadcast in logarithmic time on all networks of type G_S .

3 APPLICATIONS

The procedure *Decay* has been used to resolve contention in radio and cellular phone networks.

4 CROSS REFERENCE

031,221,222,317.

5 RECOMMENDED READING

Subsequent papers showed the optimality of the randomized algorithm:

- Alon et al. [1] showed the existence of a family of radius-2 networks on n vertices for which any broadcast schedule requires at least $\Omega(\log^2 n)$ time slots.
- Kushilevitz and Mansour [7] showed that for any randomized broadcast protocol there exists a network in which the expected time to broadcast a message is $\Omega(D \log(N/D))$.
- Bruschi and Del Pinto [3] showed that for any deterministic distributed broadcast algorithm, any n and $D \leq n/2$ there exists a network with n nodes and diameter D such that the time needed for broadcast is $\Omega(D \log n)$.
- Kowalski and Pelc [6] discussed networks in which collisions are indistinguishable from the absence of transmission. They showed an $\Omega(n \log n / \log(n/D))$ lower bound and an $O(n \log n)$ upper bound. For this model, they also showed an $O(D \log n + \log^2 n)$ randomized algorithm, thus matching the lower bound of [1] and improving the bound of [2] for graphs for which $D = \theta(n / \log n)$.

References

- [1] Noga Alon, Amotz Bar-Noy, Nathan Linial, and David Peleg. A lower bound for radio broadcast. *J. Comput. Syst. Sci.*, 43(2):290–298, 1991.
- [2] Reuven Bar-Yehuda, Oded Goldreich, and Alon Itai. On the time-complexity of broadcast in multi-hop radio networks: An exponential gap between determinism and randomization. *J. Comput. Syst. Sci.*, 45(1):104–126, 1992.
- [3] Danilo Bruschi and Massimiliano Del Pinto. Lower bounds for the broadcast problem in mobile radio networks. *Distrib. Comput.*, 10(3):129–135, 1997.
- [4] Micha Hofri. A feedback-less distributed broadcast algorithm for multihop radio networks with time-varying structure. In *Computer Performance and Reliability*, pages 353–368, 1987.

- [5] Dariusz R. Kowalski and Andrzej Pelc. Deterministic broadcasting time in radio networks of unknown topology. In *FOCS '02: Proceedings of the 43rd Symposium on Foundations of Computer Science*, pages 63–72, Washington, DC, USA, 2002. IEEE Computer Society.
- [6] Dariusz R. Kowalski and Andrzej Pelc. Broadcasting in undirected ad hoc radio networks. *Distributed Computing*, 18(1):43–57, 2005.
- [7] Eyal Kushilevitz and Yishay Mansour. An $\Omega(d \log(n/d))$ lower bound for broadcast in radio networks. In *PODC*, pages 65–74, 1993.