

# Communicating with Beeps\*

Artur Czumaj<sup>1</sup> and Peter Davies<sup>2</sup>

- 1 Department of Computer Science, Centre for Discrete Mathematics and its Applications (DIMAP), University of Warwick, Warwick, UK  
A.Czumaj@warwick.ac.uk
- 2 Department of Computer Science, Centre for Discrete Mathematics and its Applications (DIMAP), University of Warwick, Warwick, UK  
P.W.Davies@warwick.ac.uk

---

## Abstract

The *beep model* is a very weak communications model in which devices in a network can communicate only via beeps and silence. As a result of its weak assumptions, it has broad applicability to many different implementations of communications networks. This comes at the cost of a restrictive environment for algorithm design.

Despite being only recently introduced, the beep model has received considerable attention, in part due to its relationship with other communication models such as that of ad-hoc radio networks. However, there has been no definitive published result for several fundamental tasks in the model. We aim to rectify this with our paper.

We present algorithms for the tasks of broadcast, gossiping, and multi-broadcast, and also, as intermediary results, means of depth-first search and diameter estimation. Our  $O(D + \log M)$ -time algorithm for broadcasting is a simple formalization of a concept known as beep waves, and is asymptotically optimal. We give an  $O(n \log L)$ -time depth-first search procedure, and show how this can be used as the basis for an  $O(n \log LM)$ -time gossiping algorithm. Finally, we approach the more general problem of multi-broadcast. We differentiate between two variants of this problem: one where nodes must know the origin of all source messages, and another where this information is not required. In the first instance we achieve an algorithm running in time  $O(k \log \frac{LM}{k} + D \log L)$ , and in the second an  $O(k \log \frac{M}{k} + D \log L)$ -time algorithm (or  $O(M + D \log L)$  when  $M \leq k$ ). We then give corresponding lower bounds:  $\Omega(k \log \frac{LM}{k} + D)$  in the case where nodes must know message origins, and  $\Omega(k \log \frac{M}{k} + D)$  and  $\Omega(M + D)$  in the other case, for  $M > k$  and  $M \leq k$  respectively. These lower bounds demonstrate that our algorithms are optimal except for the  $D \log L$  additive term. In these running-time expressions,  $n$  represents network size,  $D$  network diameter,  $L$  range of node labels,  $M$  range of source messages, and  $k$  number of sources.

Our algorithms are all explicit, deterministic, and practical, and give efficient means of communication while making arguably the minimum possible assumptions about the network.

**1998 ACM Subject Classification** C.2.1 Distributed Networks

**Keywords and phrases** Beep model, Communication networks, Broadcasting, Gossiping, Leader election

**Digital Object Identifier** 10.4230/LIPIcs.OPODIS.2015.30

---

\* Research partially supported by the Centre for Discrete Mathematics and its Applications (DIMAP).



## 1 Introduction

The *beep model*, introduced recently by Cornejo and Kuhn [3], is a very weak network communications model in which information can be passed only in the form of a beep or a lack thereof. The model is related to the ad-hoc radio network model, and has been used as a surrogate model in results concerning radio networks with collision detection. As well as attracting study from this angle, the beep model is interesting in its own right because of its generality, simplicity, and wide range of areas where it could be applied.

### 1.1 Model

The network is modeled as an undirected connected graph  $G = (V, E)$ , where vertices in the graph represent devices in the network, and edges represent direct reachability. Time is divided into discrete steps, with a synchronized global clock (though, as in [4], we can extend to the case where only a subset of nodes wake up at time 0 and others must be woken by receiving beeps). In each time-step every node decides whether to *beep* or to *listen*. Nodes which choose to listen in a particular time-step hear a beep if at least one of their neighbors chose to beep, and they cannot distinguish between one neighbor beeping or many. We will assume that nodes have unique labels (IDs), which is essential (at least when considering deterministic algorithms) in order to break symmetry.

We will use the following parameters in analysis of our algorithms:

- $n$  will denote network size, i.e.,  $|V|$ .
- $D$  will denote network diameter, the largest distance between any pair of nodes.
- $L$  will be the range of node labels, i.e., labels will be strings of no more than  $\log L$  bits.
- $M$  will be the range of messages, i.e., messages will be strings of no more than  $\log M$  bits.
- $k$  will be the number of source nodes when considering the multi-broadcast task.

We do not, however, assume that nodes have any prior knowledge of these parameters, nor any other knowledge about the network.

### 1.2 Related Work

There has been a large amount of research focusing on fundamental communication problems in distributed computing, see e.g., [9] and the references therein. The beep model was introduced by Cornejo and Kuhn [3], who used it to design an algorithm for interval coloring. This task is a variant of vertex coloring used in resource allocation problems, and is, in a sense, tailored to the model. In another recent work, Afek et al. [1] presented an algorithm for finding a maximal independent set in the beep model. The beep model is strictly weaker than the model of radio networks with collision detection (see, e.g., [9]), though the aforementioned two results did not approach it from this angle, and so algorithmic results in the former also apply in the latter. This relationship was exploited by Ghaffari and Haeupler [6] to give almost optimal  $O((D + \log n \log \log n) \cdot \min(\log \log n, \log \frac{n}{D}))$ -time randomized algorithm for leader election in radio networks with collision detection. Ghaffari and Haeupler [6] introduces the method of “beep waves” to transmit bit strings, a method which is also employed here for the purpose of broadcast. Ghaffari et al. [5] give a randomized broadcast algorithm in radio networks with collision detection which employs beeping techniques, but, unlike the algorithm of [6], does not entirely translate over to the beep model. A deterministic leader election algorithm in the beep model was given by Förster et al. [4], taking  $O(D \log L)$  time. While a simple binary search approach, like that used in [2] for radio networks, gives the same running time, the method of [4] has the benefit of not requiring prior knowledge of

parameters  $D$  and  $L$ , an advantage which we make use of in our results. In another related work, Gilbert and Newport [7] studied the quantity of computational resources needed to solve specific problems in the beep model.

Concurrently with this paper, Hounkanli and Pelc [8] give a  $O(D + \log M)$  time broadcasting algorithm and an  $O(n^2 \log M + nD \log L)$ -time gossiping algorithm in a slightly different model where nodes know network parameters  $n, L, M$  but wake-up at arbitrary different time-steps, rather than simultaneously. To our knowledge there have been no earlier published results for broadcast, gossiping, and multi-broadcast in the model we study.

### 1.3 Our Results

In this paper, we present the following results:

- An optimal  $O(D + \log M)$ -time algorithm for broadcasting a  $\log M$  bit message, developing and formalizing the “beep waves” method of [6].
- An  $O(n \log L)$ -time procedure for performing depth-first search.
- An  $O(n \log LM)$ -time gossiping algorithm based on depth-first search.
- An  $O(D)$ -time procedure for estimating diameter.
- An  $O(k \log \frac{LM}{k} + D \log L)$ -time algorithm for multi-broadcast with provenance (where every node must learn all (source ID, source message) pairs).
- A corresponding  $\Omega(k \log \frac{LM}{k} + D)$  lower bound.
- An algorithm for multi-broadcast without provenance (where every node must learn all unique source messages) taking  $O(k \log \frac{M}{k} + D \log L)$  time when  $M > k$  and  $O(M + D \log L)$  time when  $M \leq k$ .
- A corresponding lower bound of  $\Omega(k \log \frac{M}{k} + D)$  when  $M > k$  and  $\Omega(M + D)$  when  $M \leq k$ .
- These multi-broadcasting algorithms imply  $O(n \log \frac{LM}{n} + D \log L)$  and  $(n \log \frac{M}{n} + D \log L)$ -time gossiping algorithms with and without provenance respectively.

The multi-broadcasting algorithms are our most significant results. The first outperforms the DFS-based gossiping algorithm despite being designed for a more general problem. Furthermore, perhaps surprisingly, the second is faster even than the  $k \log M$  time-steps required for a node to directly transmit or hear the source messages, which might intuitively have appeared to be a lower bound for the problem.

## 2 Broadcasting

Broadcasting is perhaps the most fundamental task in distributed communication models. It assumes that one designated source node has a message (which we will assume to be an integer in the range  $[0, M - 1]$ ) that must be known by all nodes in the network. We achieve optimal an  $O(D + \log M)$ -time algorithm for broadcasting based on the idea of “beep waves.”

### 2.1 Beep Waves

Beep waves were first introduced by Ghaffari and Haeupler [6] as a means of transmitting information in the beep model. Variations of the technique are useful for different circumstances, and here we give a simple formalization tailored to the task of broadcasting from a single source.

The idea is the following: every three time-steps, starting at zero, the source transmits a bit of its message, that is it beeps to represent a  $\mathbf{1}$  or remains silent to represent a  $\mathbf{0}$ . We can encode the message so that it is obvious when the beginning and end are, for example by

duplicating every bit of the message and then placing **10** at the beginning and end. We will denote this coding method  $C$ , and note that for any message  $m$ ,  $|C(m)| \leq 2|m| + 4$ . When we refer to the size of the message, we mean its length in bits, i.e.  $|m| \leq \log M$ . It is easy to see that we can decode to find the original message(s), even if there are several, separated by any number of **0s**. This will become necessary within algorithms for more complex tasks which involve several successive broadcasts.

All non-source nodes, upon hearing a beep in some time-step  $i$ , then relay the beep themselves in time-step  $i + 1$ , unless they themselves beeped in time-step  $i - 1$ .

---

**Algorithm 1** BEEP-WAVE( $s, m(s)$ ) at source  $s$

---

```

for  $i = 1$  to  $|C(m(s))|$  do
  if bit  $C(m(s))_i$  is 1 then
     $s$  beeps in time-step  $3i$ 
  end if
end for

```

---



---

**Algorithm 2** BEEP-WAVE( $s, m(s)$ ) at non-source  $u$

---

```

while end of message not heard do
  if  $u$  hears a beep in time-step  $i$  and did not itself beep in time-step  $i - 1$  then
     $u$  beeps in time-step  $i + 1$ 
    bit  $m(u)_{\lfloor i/3 \rfloor} \leftarrow 1$ 
  end if
end while
output  $C^{-1}(m(u))$ 

```

---

► **Lemma 1.** BEEP-WAVE( $s, m(s)$ ) correctly performs broadcast in time  $O(D + |m(s)|) = O(D + \log M)$ .

**Proof.** Partition all nodes into layers depending on their distance from the source  $s$ , i.e. layer  $L_i = \{v \in V : \text{dist}(v, s) = i\}$ . Every beep emitted by the source is propagated one layer per time-step, reaching all nodes in layer  $i$  after  $i$  time-steps. Nodes in layer  $i$  only ever relay beeps from layer  $i - 1$ , because the only times layer  $i + 1$  beeps are directly after layer  $i$  does. This can be seen by an inductive argument.

Therefore, a node in layer  $i$  receives a beep exactly  $i$  steps after the source transmits one, and so can decode its received bit-string to recover the source's message. ◀

To use our broadcast algorithm, we must have a designated source node, and we must also have a good estimate of  $D$  if we want to know how much time to allow for completion. We do not require nodes to have access to a synchronized global clock when performing BEEP-WAVE; however, non-source nodes must know that they should be behaving in a “beep-forwarding” fashion. If we wish to use broadcast as part of larger algorithms, then we must take care to ensure that each node has the correct behavior during the time period involved.

## 2.2 Lower Bound for Broadcasting

It is straightforward to demonstrate that the  $O(D + \log M)$  running time of BEEP-WAVE is asymptotically optimal:

► **Lemma 2.** *Any algorithm for broadcasting a message  $m(s) \in [0, M - 1]$  must take at least  $c(D + \log M)$  time-steps for some constant  $c$ .*

**Proof.** The message can be any of  $M$  different bit-strings, and so requires  $\log M$  bits to specify. Let  $u$  be the furthest non-source node from the source  $s$ . Since  $u$  can only receive at most one bit of information per time-step (a beep or silence),  $\log M$  time-steps must be required for it to know the message.

Information can only be propagated through the network at most one adjacency layer per time-step, since non-adjacent nodes have no means of communication. Therefore at least  $\text{dist}(u, s)$  time-steps are required for any information to reach  $u$ . Since  $\text{dist}(u, s) \geq \frac{D}{2}$ , the total number of time-steps required to inform  $u$  of the message is at least  $\max(\frac{D}{2}, \log M) \geq \frac{1}{2}(\frac{D}{2} + \log M) \geq \frac{1}{4}(D + \log M)$  ◀

### 3 Leader Election

If we wish to use broadcast as part of a more complex algorithm, we must be sure that we have a single source who wishes to send a message. To ensure this, we can perform the task of leader election.

Leader election enables all nodes to agree on the ID of one particular node. In our applications, we will always choose the node with the highest ID in the entire network. More generally, though, leader election can be used on any subset of nodes, whenever each holds some integer value, to find the participating node with the highest (or lowest) such value. The values need not even be unique, since if multiple nodes hold the target value, we can pick out one by performing leader election again on their IDs. Leader election, particularly when used in this way, is sometimes also referred to as Find Max.

We wish to be able to perform leader election in  $O(D \log L)$  time. We note that there is a straightforward way to do this: we can perform a binary search for the highest ID, iterating through the bits of the IDs and having all nodes who are still “in the running” for leader, and who have a 1 in the current position, broadcast. While we cannot use our previous broadcast procedure with multiple sources, since these nodes need only transmit a single bit we can still use beep-waves to ensure that the network hears *something*. This is sufficient for all nodes to determine whether any have a 1 in the current position. A similar method to this was used to perform leader election in radio networks in [2].

However, there is a problem with this approach: we would need a common linear upper bound on  $D$  and a polynomial upper bound on  $L$  to correctly perform it. Since we do not assume this knowledge, we instead make use of a result of Förster et al. [4] (paraphrased):

► **Theorem 3.** *There is an algorithm ELECTLEADER which performs leader election in time  $O(D \log L)$  without prior knowledge of  $D$  or  $L$ .*

Upon completion, all nodes have knowledge of the highest ID, and can therefore use this as  $L$  in future operations. Förster et al. extend their algorithm to function when only some subset of nodes wake up at time 0, removing the assumption of synchronous wake-up. Since we employ it as a subroutine at the beginning of all our forthcoming algorithms, they can also forgo this assumption.

### 4 Network Traversal

We may wish to perform operations which require an organized exploration of the entire network. For this purpose, we give a procedure for depth-first search, and an application to the task of gossiping.

## 4.1 Depth-First Search

Depth first search is performed here in fundamentally the usual way. There is a network-wide “token,” i.e., only one node is the ‘active’ node at any one time. This node checks for unexplored neighbors, passes the token to one if any exist, or sends it back to its parent if not. Here we also wish to pass round a counter, incremented upon reaching each new unexplored node, so that nodes know the order in which they were explored.

To detect unexplored neighbors we use a process much like the binary-search method mentioned for leader election, in which nodes iteratively agree on each bit of an ID. Here, though, we do not need to broadcast to the whole network between every step, since the nodes involved are all adjacent to the current active node. To organize this process we need predetermined constant-size control messages; this can easily be achieved by using any sensible system of code-words.

To apply Algorithm 4 we must first have a designated leader node. This leader is the parameter  $v$  taken as input in our description of the algorithm.

---

### Algorithm 3 DEPTH-FIRST SEARCH( $v, x, count$ )

---

```

number( $x$ )  $\leftarrow$  count
loop
   $x$  transmits “child-acknowledge” message
  unvisited neighbors beep
  if  $x$  received no beep then
    break loop
  end if
   $x$  transmits “child-search” message
  for  $i = 1$  to  $\log L$  do
    unvisited nodes still in running for highest ID beep if  $i^{th}$  bit of ID is 1
     $x$  transmits “acknowledge 1” message if it heard a beep, “acknowledge 0” otherwise
    if “acknowledge 1” received, nodes with  $i^{th}$  bit 0 drop out
  end for
   $y \leftarrow$  ID highest unvisited neighbor
   $x$  transmits the message ( $y, count + 1$ )
   $count \leftarrow$  DEPTH-FIRST SEARCH( $v, y, count + 1$ )
end loop
if  $v = x$  then
  terminate procedure
else
  return  $count$  (by transmitting back to parent)
end if

```

---

► **Lemma 4.** DEPTH-FIRST SEARCH( $v, v, 1$ ) correctly performs depth-first search within  $O(n \log L)$  time-steps.

**Proof.** Each round of the “child-search” loop ensures that the current token node and its unexplored children all agree on a bit of the target ID, and since only one node has the token there will be no interference from the rest of the network. Thus, after  $\log L$  such rounds, the ID of the next node to explore is agreed upon. This process must be performed  $n$  times to explore the entire graph, taking  $n \log L$  total time-steps. We must also consider the cost of

passing *count*, which is  $O(\log n)$  time-steps each time the token moves, and so  $O(n \log n)$  in total. Since  $L \geq n$ , total running time is  $O(n \log L)$ . ◀

## 4.2 DFS-Based Gossiping

An application of our depth-first search algorithm is to the task of gossiping. The premise of gossiping is that every node has a message which must become known to the entire network. We can achieve this in  $O(n \log LM)$  time by first electing a leader, then performing depth-first search, and finally having each node broadcast its message in the order in which it was explored by the DFS.

This last broadcast stage is not quite as straightforward as it may seem, since in general  $n$  different consecutive broadcasts would take  $O(n(D + \log M))$  time, exceeding our desired  $O(n \log LM)$  running time. However, since we can encode messages so that it is obvious when they start and end (without affecting asymptotic size), and we also know that transmissions during BEEP-WAVE move exactly one distance layer per time-step and never move backwards, we can pipeline the broadcasts. That is, once a node's message, in its entirety, has been heard by the next node in the ordering, that next node can immediately begin its own broadcast without waiting for the previous message to reach the entire network. The waves of beeps will not interfere with each other, since the start of the new message cannot reach any node quicker than the end of the old message, and the behavior of all other nodes does not change, so they do not need to know the precise time-step when the new node starts broadcasting.

---

### Algorithm 4 GOSSIP( $m(V)$ )

---

```

 $v \leftarrow \text{ELECTLEADER}$ 
perform DEPTH-FIRST SEARCH( $v, v, 1$ )
for  $i = 1$  to  $n$  do
  let  $u$  be such that  $\text{number}(u) = i$ 
  BEEP-WAVE( $u, m(u)$ )
end for

```

---

► **Theorem 5.** GOSSIP correctly performs gossiping within  $O(n \log LM)$  time-steps.

**Proof.** Performing ELECTLEADER and DEPTH-FIRST SEARCH takes  $O(n \log L)$  time in total. Upon completion, each node knows its ordering in the DFS tree. Nodes then broadcast in order, and a node can begin broadcasting immediately after hearing the end of the message from its predecessor. The total time taken for all  $n$  broadcasts is then  $O(\sum_{i=1}^n (\log M + \text{dist}(i, i+1)))$ . Since this sum of distances is no greater than the distance traveled when traversing the DFS tree, this expression is  $O(n \log M)$ . Therefore total running time is  $O(n \log L + n \log M) = O(n \log LM)$ . ◀

## 5 Auxiliary Procedures

We next define some procedures for useful auxiliary tasks, which we will need for our multi-broadcast algorithms, but are also general enough to be useful elsewhere. Specifically, we give protocols for:

- Diameter estimation, i.e. allowing all nodes to calculate a common linear upper bound on  $D$ .
- Message collection, where a designated leader node receives the logical **OR**-superimposition of bit-strings from several source nodes.



- Message length determination, i.e. providing all nodes with the size of the largest of a set of bit-strings from source nodes.

## 5.1 Diameter Estimation

Our model assumes that nodes do not have access to any of the network parameters. In algorithms for complex tasks, we generally wish to start with a leader election phase, and this provides all nodes with knowledge of  $L$ . However, if we also wish to know the value of  $D$ , we must perform an extra task for this purpose.

Our diameter estimation procedure (Algorithm 5) works as follows: we take as input a leader node to co-ordinate the process. An initial beep from the leader propagates through the network. Having received this beep, nodes beep to acknowledge their existence back to the leader; a modularity restriction on when nodes can transmit ensures that these beeps only travel backwards through the layers. While the initial beep from the leader is still reaching further nodes, acknowledgment beeps will continue to return through the network every three time-steps. Once all nodes have been reached, this pattern will cease, and the leader will know that the diameter of the graph is no greater than the current time-step value. All of the other nodes have also ceased transmission, and so an application of BEEP-WAVE can safely be used to broadcast the diameter estimate.

We split the algorithm into two parts, one performed by the leader, and one performed by all non-leader nodes, since their behavior is quite different.

---

**Algorithm 5** ESTIMATEDIAMETER( $v$ ) at leader  $v$

---

```

 $v$  beeps in time-step 1
let  $i$  be the first time-step (greater than 2) in which  $v$  has not received a beep for 3 previous
time-steps
 $\tilde{D} \leftarrow i$ 
perform BEEP-WAVE( $v, \tilde{D}$ )
output  $\tilde{D}$ 

```

---



---

**Algorithm 6** ESTIMATEDIAMETER( $v$ ) at non-leader  $u$

---

```

let  $j$  be the first time-step in which  $u$  receives a beep
 $u$  beeps in the next time-step which is equivalent to  $(-j) \bmod 3$ 
while  $u$  has heard a beep in the last 3 time-steps do
    any beep  $u$  hears in a time-step equivalent to  $(2 - j) \bmod 3$ , it relays in the next
time-step
end while
 $\tilde{D} \leftarrow$  BEEP-WAVE( $v, \tilde{D}$ )
output  $\tilde{D}$ 

```

---

► **Lemma 6.** ESTIMATEDIAMETER( $v$ ) correctly broadcasts an estimate  $\tilde{D}$  satisfying  $D \leq \tilde{D} \leq 2D + 7$ , and terminates within  $O(D)$  time-steps.

**Proof.** Let  $D'$  be the distance from the leader to the furthest node. Then,  $D \geq D' \geq D/2$ . The leader emits a beep in time-step 1 which travels to this furthest node by time-step  $D' + 1$ . After at most 3 more time-steps, the node transmits its acknowledgment beep, which travels back to the leader in a further  $D'$  time-steps. After another 3 time-steps, the leader knows



that it has received the final acknowledgment beep, and takes the current time-step  $i$  as its diameter estimate. Since  $2D' \leq i \leq 2D' + 7$ , the estimate  $\tilde{D}$  which is broadcast to the network satisfies  $D \leq \tilde{D} \leq 2D + 7$ . Running time of the estimation phase is no more than  $2D + 7$  time-steps, and the final broadcast takes  $O(D + \log D) = O(D)$  time.  $\blacktriangleleft$

Since we are only interested in asymptotic behavior, we will assume, for ease of notation, that having performed ESTIMATEDIAMETER as part of a more complex algorithm we can then make use of the exact value of  $D$ .

## 5.2 Message Collection

We next introduce a sub-procedure (Algorithm 7) which will allow the leader to collect messages  $m(S)$  from a set of sources  $S$ , receiving an **OR**-superimposition of all the messages. This works similarly to the usual beep-waves procedure, except that nodes use their distance from the leader (inferred by the time taken to receive the initial BEEP-WAVE( $v, \mathbf{1}$ )) to ensure that the waves only travel towards the source, and all messages arrive at the same time. We must have an input parameter  $p$  giving an upper bound on the length of messages, so that nodes know when the procedure is finished, and we assume that we have already performed ESTIMATEDIAMETER and so can make use of  $D$ .

---

**Algorithm 7** COLLECTMESSAGES( $v, S, m(S), p$ ) at source  $s \in S \setminus \{v\}$

---

```

perform BEEP-WAVE( $v, \mathbf{1}$ )
for  $i = 1$  to  $p$  do
  if bit  $m(s)_i$  is 1 then
     $s$  beeps in time-step  $3i + D - \text{dist}(s)$ 
  end if
end for

```

---



---

**Algorithm 8** COLLECTMESSAGES( $v, S, m(S), p$ ) at  $u \notin S \setminus \{v\}$

---

```

perform BEEP-WAVE( $v, \mathbf{1}$ )
for  $j = 0$  to  $3p + D$  do
  if  $u$  hears a beep in time-step  $j$  such that  $j \equiv 2 + D - \text{dist}(u) \pmod{3}$  then
     $u$  beeps in time-step  $j + 1$ 
    if  $u = v$  then
      bit  $m(u)_{\lfloor j/3 \rfloor} \leftarrow 1$ 
    end if
  end if
end for
output  $m(v)$ 

```

---

We note that even if leader  $v$  is itself a source, it should perform the non-source behavior. Since it already knows its own message, it can superimpose it with the string it receives manually upon termination of the procedure.

► **Lemma 7.** COLLECTMESSAGES( $v, S, m(S), p$ ) correctly informs  $v$  of the **OR**-superimposition of  $m(S)$  within  $O(D + p)$  time-steps.

**Proof.** The modularity restriction on relaying beeps ensures that beep-waves only travel towards the  $v$ , and the starting times for sources ensure that bits in the same position

arrive simultaneously. Thus,  $v$  hears a **1** in a position iff one of the sources messages contained a **1** in the same position. After  $D + 3p$  time-steps it must then have received the **OR**-superimposition of  $m(S)$ . ◀

### 5.3 Message Length Determination

One issue with using COLLECTMESSAGES is the necessity of prior knowledge of a common upper bound on message size. We give a simple method of obtaining this bound (Algorithm 9).

We perform COLLECTMESSAGES using strings which are as long as the messages we actually want to collect, but consist of entirely **1**s. The superimposition of these strings is a **1**-string of equal length to the longest message. Since the leader will be able to tell that this string has ended when it hears the substring **10**, the procedure can be terminated even without an upper bound for the COLLECTMESSAGES call.

---

#### Algorithm 9 GETMESSAGELENGTH( $v, S, m(S)$ )

---

perform  $p \leftarrow \text{COLLECTMESSAGES}(v, S, \mathbf{1}^{m(S)}, \infty)$ , terminating when  $v$  hears the substring **10**  
 perform BEEP-WAVE( $v, |p|$ )  
 output  $|p|$

---

► **Lemma 8.** GETMESSAGELENGTH( $v, S, m(S)$ ) correctly informs all nodes of  $p = \max_{s \in S} |m(s)|$  within  $O(D + p)$  time-steps.

**Proof.** COLLECTMESSAGES will terminate after  $D + 3p$  steps, since  $v$  will hear the final **1** and then a **0**. All other nodes will be inactive and so BEEP-WAVE( $v, |p|$ ) will successfully inform the network of  $p$  (nodes will be aware that the COLLECTMESSAGES phase is over and so perform BEEP-WAVE correctly, since they either heard a string of contiguous **1**s and then a **0** during COLLECTMESSAGES, or silence for more than  $D$  time-steps).

Running time is  $O(D + p)$  for COLLECTMESSAGES and  $O(D + \log p)$  for BEEP-WAVE, giving  $O(D + p)$  total. ◀

## 6 Multi-Broadcast

We are now ready to approach the most general of the communication tasks we will consider, that of multi-broadcast. As in gossiping, multiple source nodes have messages which must become known to the entire network. However, rather than all nodes being sources, only those belonging to some unknown subset are. We denote the number of sources as  $k$ , but this value is not known to the network.

There are two slightly different variants of the problem we consider: *multi-broadcast with provenance*, where the network must become aware of all (source ID, source message) pairs, and *multi-broadcast without provenance*, where the IDs need not be known. Since we do not assume that messages are unique, we also allow in the case without provenance that only one copy of each distinct message must be output. That is, nodes need not be aware of how many sources held each message.

### 6.1 Multi-Broadcast With Provenance

We first present an algorithm for multi-broadcast with provenance, where all nodes must be made aware of not only the source messages, but also the IDs of the sources they originated from.

The idea of the algorithm is essentially to conduct  $k$  simultaneous binary searches to allow a leader to ascertain the IDs of all sources. The process consists of  $\log L$  rounds, one for each bit of the IDs. Each node will maintain a list of known prefixes of source IDs, and we aim to preserve the invariant that, after round  $i$ , all nodes know the first  $i$  bits of every source ID. We denote the number of distinct known prefixes at the start of round  $i$  by  $k_i$ .

At the start of round  $i$ , sources know  $k_i$  distinct  $i - 1$ -bit ID prefixes (note  $k_i$  may be less than  $k$ , since some IDs may share prefixes), and they will each construct a  $2k_i$ -bit string in which each bit corresponds to a particular  $i$ -bit prefix. Specifically, if we denote the known prefixes in lexicographical order by  $(p_1, p_2, \dots, p_{k_i})$ , then bit  $2j$  in the new string will represent the prefix  $p_j\mathbf{0}$ , and bit  $2j + 1$  will represent  $p_j\mathbf{1}$ . Each source constructs its string by placing a  $\mathbf{1}$  in the position corresponding to its own ID's  $i$ -bit prefix, and  $\mathbf{0}$  in all others. We will denote the string constructed in this manner by source  $s$  in round  $i$  by  $Z_{s,i}$ .

Performing COLLECTMESSAGES with these strings ensures that the leader receives the **OR**-superimposition, which informs it of all  $i$ -bit prefixes of source IDs (since it is aware of which prefix each position corresponds to). It then broadcasts this back out to the network via the standard beep wave procedure, and thus the invariant is fulfilled round  $i$ . After  $\log L$  rounds, the IDs of all sources are known in entirety by all nodes. We then perform one final COLLECTMESSAGES procedure, this time to collate all of the messages the sources wish to broadcast to the network. We construct a  $k \log M$ -bit string in which the  $j^{\text{th}}$  block of  $\log M$  bits corresponds to the message of the  $j^{\text{th}}$  source (in lexicographical order of ID). Each source individually fills in its own message in the appropriate block, leaving all other bits as  $\mathbf{0}$ . We denote the string constructed in this manner by source  $s$  as  $\tilde{m}_s$ . Performing COLLECTMESSAGES on these strings ensures that the full string of messages arrives at the leader, who then broadcasts it back out to the network.

---

**Algorithm 10** MULTI-BROADCAST WITH PROVENANCE( $S, m(S)$ )
 

---

```

 $v \leftarrow \text{ELECTLEADER}$ 
 $D \leftarrow \text{ESTIMATEDIAMETER}(v)$ 
 $\log M \leftarrow \text{GETMESSAGELENGTH}(v, S, m(S))$ 
for  $i = 1$  to  $\log L$  do
     $Z_i \leftarrow \text{COLLECTMESSAGES}(v, S, Z_{S,i}, 2k_i)$ 
    perform BEEP-WAVE( $v, Z_i$ )
end for
 $\tilde{m} \leftarrow \text{COLLECTMESSAGES}(v, S, \tilde{m}_S, k \cdot \log M)$ 
perform BEEP-WAVE( $v, \tilde{m}$ )
  
```

---

► **Theorem 9.** MULTI-BROADCAST WITH PROVENANCE( $S, m(S)$ ) *correctly performs multi-broadcast with provenance within  $O(k \log \frac{LM}{k} + D \log L)$  time-steps.*

**Proof.** The three sub-procedure calls in initial “set-up” phase take a total of  $O(D \log L + \log M)$  time-steps, and provide a leader node and knowledge of  $D$  and  $\log M$ .

Round  $i$  of the main loop of the algorithm takes  $O(D + k_i)$  time, since it consists of performing COLLECTMESSAGES on strings of length  $O(k_i)$ , and then BEEP-WAVE on a string of the same length. Furthermore, since the number of known prefixes at most doubles each round,  $k_i \leq 2^{i-1}$ . Hence, there exists some constant  $c$  such that total time for the loop is

bounded by:

$$\begin{aligned}
\sum_{i=1}^{\log L} c(D + k_i) &= cD \log L + c \left( \sum_{i=1}^{\log k} k_i + \sum_{i=\log k+1}^{\log L} k_i \right) \\
&\leq cD \log L + c \left( \sum_{i=1}^{\log k} 2^{i-1} + \sum_{i=\log k+1}^{\log L} k \right) \\
&\leq cD \log L + c(k + k(\log L - \log k)) \\
&= O(D \log L + k \log \frac{L}{k}) .
\end{aligned}$$

The final call to COLLECTMESSAGES then takes a further  $O(D + k \log M)$  time, and so total running time is  $O(D \log L + k \log \frac{L}{k} + k \log M) = O(k \log \frac{LM}{k} + D \log L)$

Correctness follows since each round of the loop informs the leader of the next bit in each ID prefix, and it then broadcasts this information to the network. After  $\log L$  rounds, all nodes know all source IDs and each source  $s$  can correctly construct its string  $\tilde{m}_s$ . The **OR**-superimposition of these strings, broadcast to all nodes, is a list of messages in source ID order, which fulfills the goal of the algorithm. ◀

We remark that this result yields an algorithm for gossiping with running time  $O(n \log \frac{LM}{n} + D \log L)$ , slightly improving over the  $O(n \log LM)$  running time of Algorithm 4.

## 6.2 Multi-Broadcast Without Provenance

It may be the case that we do not need to know where messages originated from, or the number of duplicate messages; for example when using short control messages instructing all nodes to perform some action, for which provenance might be irrelevant. For this reason, we also study the variant of multi-broadcast where nodes need only know one copy of each unique source message, and no source IDs.

The main difference in concept for our multi-broadcast without provenance algorithm (Algorithm 11) is that the concurrent binary searches are performed on the bits of the source messages rather than the IDs. However, this turns out to be slower when  $k$  is smaller than  $D$ , and so we first run Algorithm 10, curtailing it when our number  $k_i$  of known ID prefixes (which is a lower bound for  $k$ ) exceeds  $D$ , in order to efficiently deal with these cases.

If  $k \leq D$  then the call to Algorithm 10 will complete multi-broadcast (meeting the requirements for the case without provenance, since they are strictly weaker than those with provenance). Otherwise, we move onto performing binary searches on the bits of the message. This functions in much the same way as in Algorithm 10, except that we do not need the final COLLECTMESSAGES and BEEP-WAVE stage since the network is already aware of all source messages upon completion of the main loop. We will use  $\tilde{k}_i$  to denote the number of  $i - 1$ -bit message prefixes known to nodes at the start of round  $i$  of the for loop, and  $\tilde{Z}_{s,i}$  to be the string constructed by source  $s$  in round  $i$  by placing a **1** in the position corresponding to the  $i$ -bit prefix of its message and **0** in all others.

► **Theorem 10.** MULTI-BROADCAST WITHOUT PROVENANCE( $S, m(S)$ ) correctly performs multi-broadcast without provenance within  $O(k \log \frac{M}{k} + D \log L)$  time-steps if  $k < M$ , and  $O(M + D \log L)$  time-steps if  $k \geq M$ .

**Proof.** By the same argument as for Theorem 10, each round of the main loop informs all nodes of the next bit in each message prefix. Therefore, after  $\log M$  rounds we have performed multi-broadcast without provenance.

**Algorithm 11** MULTI-BROADCAST WITHOUT PROVENANCE( $S, m(S)$ )

---

```

perform MULTI-BROADCAST WITH PROVENANCE( $S, m(S)$ ) until  $k_i > D$ 
if it did not complete then
  for  $i = 1$  to  $\log M$  do
     $\tilde{Z}_i \leftarrow \text{COLLECTMESSAGES}(v, S, \tilde{Z}_{S,i}, 2\tilde{k}_i)$ 
    perform BEEP-WAVE( $v, \tilde{Z}_i$ )
  end for
end if

```

---

We separate the running-time proof into four cases:

1.  $k \leq D$  and  $k < M$ .
2.  $k \leq D$  and  $k \geq M$ .
3.  $k > D$  and  $k < M$ .
4.  $k > D$  and  $k \geq M$ .

**Case 1:  $k \leq D$  and  $k < M$ .** For the  $k \leq D$  case, the number of unique  $i$ -bit source ID prefixes  $k_i$  will never exceed  $D$  (since it is bounded above by  $k$ ), and so the call to MULTI-BROADCAST WITH PROVENANCE will successfully perform multi-broadcast (with provenance, and therefore also without) in  $O(k \log \frac{LM}{k} + D \log L) = O(k \log L + k \log \frac{M}{k} + D \log L) = O(k \log \frac{M}{k} + D \log L)$  time-steps.

**Case 2:  $k \leq D$  and  $k \geq M$ .** As above, the call to MULTI-BROADCAST WITH PROVENANCE will successfully perform multi-broadcast in  $O(k \log \frac{LM}{k} + D \log L) = O(k \log L + D \log L) = O(D \log L)$  time-steps.

**Case 3:  $k > D$  and  $k < M$ .** Since  $k > D$ , the call will not complete multi-broadcast, but its “set-up” phase will provide a leader  $v$  and knowledge of  $D$  and  $\log M$ , so these steps are not duplicated in our description of Algorithm 11. Each round of the main loop then informs every node of the next bit in each unique message prefix, and so after  $\log M$  rounds we are done.

Let  $t$  be the round of the loop at which the call to MULTI-BROADCAST WITH PROVENANCE terminates. Running time for the call is then bounded above (for some constant  $c$ ) by

$$cD \log L + \sum_{i=1}^t c(D+k_i) \leq cD \log L + \sum_{i=1}^t 2cD \leq cD \log L + \sum_{i=1}^{\log L} 2cD = 3cD \log L = O(D \log L) ,$$

where the first inequality is due to the fact that  $k_i \leq D$  until termination.

Running time for the main loop of Algorithm 11 is bounded above (again for some constant  $c$ ) by:

$$\begin{aligned}
\sum_{i=1}^{\log M} c(D + \tilde{k}_i) &= cD \log M + c \left( \sum_{i=1}^{\log k} \tilde{k}_i + \sum_{i=\log k+1}^{\log M} \tilde{k}_i \right) \\
&\leq cD \log M + c \left( \sum_{i=1}^{\log k} 2^{i-1} + \sum_{i=\log k+1}^{\log M} k \right) \\
&\leq cD \log M + c(k + k(\log M - \log k)) = O(D \log M + k \log \frac{M}{k}) .
\end{aligned}$$

Total time is therefore

$$\begin{aligned} O(D \log L + D \log M + k \log \frac{M}{k}) &= O(D \log L + D \log \frac{M}{k} + D \log k + k \log \frac{M}{k}) \\ &= O(k \log \frac{M}{k} + D \log L) , \end{aligned}$$

where the last expression holds since  $D \log k \leq D \log L$  and  $D \log \frac{M}{k} \leq k \log \frac{M}{k}$ .

**Case 4:  $k > D$  and  $k \geq M$ .** The call to MULTI-BROADCAST WITH PROVENANCE will fail and take  $O(D \log L)$  time as before. Running time for the main loop of Algorithm 11 is now bounded by:

$$\begin{aligned} \sum_{i=1}^{\log M} c(D + \tilde{k}_i) &= cD \log M + c \sum_{i=1}^{\log M} \tilde{k}_i \leq cD \log M + c \sum_{i=1}^{\log M} 2^{i-1} \\ &\leq cD \log M + cM = O(D \log M + M) . \end{aligned}$$

Since  $M \leq k \leq L$ , total running time is  $O(M + D \log L)$ .

### Combining cases

When  $M > k$  total running time is  $O(k \log \frac{M}{k} + D \log L)$ , and when  $M \leq k$ , total running time is  $O(M + D \log L)$ .  $\blacktriangleleft$

It may seem unintuitive that Algorithm 11 achieves multi-broadcast in fewer than the  $k \log M$  time-steps required for a single node to directly transmit or hear the messages, since this might seem to be a natural lower bound. The improvement stems from implicit compression of the messages within the algorithm's method. We next prove lower bounds that match our algorithmic results, modulo the  $D \log L$  additive term.

## 6.3 Lower Bounds

In this section we give lower bounds for the problem of multi-broadcasting. The proofs of these bounds assume that  $k$  is greater than 1; the  $k = 1$  case follows from the lower bounds given for broadcasting.

► **Theorem 11.** *Any algorithm achieving multi-broadcast with provenance must require  $\Omega(k \log \frac{LM}{k} + D)$  time-steps.*

**Proof.**  $D$  is an obvious lower bound, since information can only be propagated via beeps one adjacency layer per time-step. Hence, if nodes  $u$  and  $v$  are the furthest pair in the network,  $D$  time-steps are required for  $v$  to receive any information from  $u$ . When  $u$  is a source, this means that at least  $D$  time-steps are required before  $v$  can know its message, which is necessary for multi-broadcast to be successful.

Consider any node  $w$ . By the end of the multi-broadcast algorithm,  $w$  must be aware of all (source ID, source message) pairs. It may be that  $w$  is itself a source, and already knows its own pair, but it must still learn all  $k - 1$  others. The number of possibilities for this  $k - 1$ -size set of pairs is  $\binom{L-1}{k-1} \cdot M^{k-1}$ , since IDs must be unique but messages need not be. The number of bits required to distinguish one particular case is at least:

$$\begin{aligned} \log \left( \binom{L-1}{k-1} \cdot M^{k-1} \right) &\geq \log \left( \frac{(L-1)M}{k-1} \right)^{k-1} = (k-1) \log \left( \frac{(L-1)M}{k-1} \right) \\ &\geq \frac{k}{2} \log \left( M \sqrt{\frac{L}{k}} \right) \geq \frac{1}{4} k \log \frac{LM}{k} . \end{aligned}$$

Here we used the inequality  $\frac{L-1}{k-1} \geq \sqrt{\frac{L}{k}}$ , which is true whenever  $L \geq k > 1$ .

Since a node can only receive at most one bit of information per time-step,  $\frac{1}{4}k \log \frac{LM}{k}$  time-steps are required for it to receive all of the information necessary for multi-broadcast with provenance. Therefore, total time required is at least  $\max(D, \frac{1}{4}k \log \frac{LM}{k}) \geq \frac{1}{8}(D + k \log \frac{LM}{k}) = \Omega(k \log \frac{LM}{k} + D)$ . ◀

► **Theorem 12.** *Any algorithm achieving multi-broadcast without provenance must require  $\Omega(k \log \frac{M}{k} + D)$  time-steps if  $M > k$ , or  $\Omega(M + D)$  if  $M \leq k$ . We assume that  $M > 1$ .*

**Proof.** As before,  $D$  is an obvious lower bound.

We will again consider the amount of information a node  $w$  must receive for multi-broadcast to be successful. Node  $w$  must become aware of the set of all source messages, and it starts with knowledge of at most 1 (its own, if it is a source). The number of possibilities for the remaining messages is at least  $\sum_{i=0}^{k-1} \binom{M-1}{i}$ , since any subset of size at most  $k-1$  of the remaining message space is possible (messages need not be unique, so there need not be exactly  $k-1$  other messages). To distinguish a particular case, the number of bits of information  $w$  must receive is at least  $\log \left( \sum_{i=0}^{k-1} \binom{M-1}{i} \right)$ .

If  $M > k$ , then we have the following:

$$\begin{aligned} \log \left( \sum_{i=0}^{k-1} \binom{M-1}{i} \right) &\geq \log \binom{M-1}{k-1} \geq \log \left( \frac{M-1}{k-1} \right)^{k-1} \geq (k-1) \log \left( \sqrt{\frac{M}{k}} \right) \\ &\geq \frac{1}{4}k \log \left( \frac{M}{k} \right). \end{aligned}$$

Similarly to the proof of Theorem 11, we used that  $\frac{M-1}{k-1} \geq \sqrt{\frac{M}{k}}$  for  $M \geq k > 1$ .

If  $M \leq k$ , then we have the following:

$$\log \left( \sum_{i=0}^{k-1} \binom{M-1}{i} \right) = \log(2^{M-1}) = M-1 \geq \frac{M}{2}.$$

So, if  $M > k$  then  $w$  must receive at least  $\frac{1}{4}k \log \frac{M}{k}$  bits, and therefore  $\max(D, \frac{1}{4}k \log \frac{M}{k}) \geq \frac{1}{8}(D + k \log \frac{M}{k}) = \Omega(k \log \frac{M}{k} + D)$  time-steps are required.

If  $M \leq k$ , then  $w$  must receive at least  $\frac{M}{2}$  bits, and therefore  $\max(D, \frac{M}{2}) \geq \frac{1}{4}(D + M) = \Omega(M + D)$  time-steps are required. ◀

The assumption that  $M > 1$  is essential; in the case that  $M = 1$  and  $k = n$ , all nodes know the only possible message already, and a multi-broadcast without provenance algorithm can terminate immediately rather than requiring the  $\Omega(D)$  time-steps the lower bound would imply.

## 7 Conclusion

The beep model is interesting as a widely applicable model that requires very little of communications devices, and can be applied even where restrictive circumstances frustrate communication under more complex models. Furthermore, it is an interesting technical challenge to design efficient algorithms while making the minimum possible assumptions about the network. In this paper we have given deterministic algorithms for several fundamental communications tasks in the beep model. The model is still young, however, and there are many remaining avenues for fruitful research.



The most pressing question our work here raises is whether the  $D \log L$  additive term in the running time of our multi-broadcasting algorithms can be reduced to  $D$ , or whether the algorithms are in fact optimal in all cases. Since  $D \log L$  is the cost of leader election, a fundamental starting point for our algorithms, any improvement would have to begin here. However, a faster leader election algorithm alone would not be sufficient to improve multi-broadcast time, since it is not a bottleneck in our algorithms; they also require  $D \log L$  time elsewhere. This may suggest that  $D \log L$  is indeed a lower bound, and again, it may be wisest to focus on leader election to prove this. Multi-broadcast with provenance and constant  $M$  is at least as hard as leader election from a set of  $k$  candidates, since after performing the multi-broadcast with the candidates as sources, their IDs (and in particular the highest ID) are known to all nodes.

A different possible focus for further research is to determine to what extent randomization can help. The leader election algorithm of [6], taking  $O((D + \log n \log \log n) \cdot \min(\log \log n, \log \frac{n}{D}))$  time and succeeding with high probability, demonstrates that improvements over deterministic algorithms can be made. It seems likely that randomization could also be of use in algorithms for multi-broadcast, though as mentioned, simply employing this randomized leader election algorithm rather than the deterministic one does not reduce the asymptotic running time of our multi-broadcast algorithms.

---

## References

- 1 Y. Afek, N. Alon, Z. Bar-Joseph, A. Cornejo, B. Haeupler, and F. Kuhn. Beeping a maximal independent set. In *Proceedings of the 25th International Symposium on Distributed Computing (DISC)*, pages 32–50, 2011.
- 2 M. Chrobak, L. Gąsieniec, and W. Rytter. Fast broadcasting and gossiping in radio networks. *Journal of Algorithms*, 43(2):177–189, 2002.
- 3 A. Cornejo and F. Kuhn. Deploying wireless networks with beeps. In *Proceedings of the 24th International Symposium on Distributed Computing (DISC)*, pages 148–262, 2010.
- 4 K.-T. Förster, J. Seidel, and R. Wattenhofer. Deterministic leader election in multi-hop beeping networks. In *Proceedings of the 28th International Symposium on Distributed Computing (DISC)*, pages 212–226, 2014.
- 5 M. Ghaffari, B. Haeupler, , and M. Khabbazi. Randomized broadcast in radio networks with collision detection. In *Proceedings of the 32nd Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 325–334, 2013.
- 6 M. Ghaffari and B. Haeupler. Near optimal leader election in multi-hop radio networks. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 748–766, 2013.
- 7 S. Gilbert and C. Newport. The computational power of beeps. In *Proceedings of the 29th International Symposium on Distributed Computing (DISC)*, pages 31–46, 2015.
- 8 K. Hounkanli and A. Pelc. Deterministic broadcasting and gossiping with beeps. In arxiv 1508.06460, 2015.
- 9 D. Peleg. Time-efficient broadcasting in radio networks: A review. In *Proceedings of the 4th International Conference on Distributed Computing and Internet Technology (ICDCIT)*, pages 1–18, 2007.