Proceedings of the Second IASTED International Conference
European Parallel and Distributed Systems (Euro-PDS'98)
July 1-3, 1998, Vienna, Austria

BROADCAST FOR PEER-TO-PEER NETWORKS *

STEFANO BASAGNI IMRICH CHLAMTAC
Erik Jonsson School of Engineering and Computer Science
The University of Texas at Dallas
2601 N. Floyd Rd. Richardson, TX 75080
E-mail: {basagni,chlamtac}@utdallas.edu

ABSTRACT

Broadcasting (distributing a message from a source to all other nodes) is a fundamental requirement of distributed computing. In this paper we propose deterministic, delivery-guaranteed distributed protocols for solving this problem in wireless networks that have a peer-to-peer (i.e., non-cellular) organization. Deterministic and delivery-guaranteed protocols are defined as solutions which are always executed within an a priori determined period of time, and such that their success does not depend on, or require, collision detection mechanisms. The proposed protocols are distributed in the sense that they can be executed at each node without topology knowledge and, moreover, they do not assume any underlying network architecture. We characterize the broadcast problem as a combinatorial problem for the solution of which we propose here a novel, explicit and completely deterministic method. The derived protocol is proved to complete the broadcast of a message in time that is polylogarithmic in the size n of the network, and we prove that it is optimal for networks with specific topologies.

Keywords: Broadcast Protocols, Distributed Algorithms, Wireless Networks, Mobile Computing.

1 INTRODUCTION

In this paper we are interested in devising deterministic and delivery-guaranteed algorithms which can be used for implementing broadcast primitives in multi-hop wireless (generally, radio) networks. Informally, broadcast is the task initiated by one node, called *source*, that wishes to send a message m to all the nodes in the network. The term multi-hop radio network, of which *peer-to-peer network* is a synonym, refers to a set of geographically dispersed nodes which may be stationary or mobile. One of the basic characteristic of these networks is that several nodes may share the same transmission channel. Thus, selective transmission is impossible: Whenever a node transmits, all of its neighbors (nodes within hearing range) will receive the message, and collision of received message may occur if some transmissions overlap, preventing correct message reception.

The broadcast problem has already been extensively studied for wireless peer-to-peer networks. For instance, in [1] a *deterministic* and *centralized* broadcast protocol is introduced which works in $O(D \log^2 n)$ rounds.[1] From the result proven in [2], such a protocol is optimal for networks with constant diameter. For networks with any (i.e., non-constant) diameter a protocol by Gaber et al. [3] can be applied that works in $O(D + \log^5 n)$ rounds. This protocol is proven to be optimal for networks with $D \in \Omega(\log^5 n)$. Centralized solutions guarantee a bounded delay on the message delivery, but require that each node in the network knows the entire network topology. This is a strong condition, which may be difficult to maintain especially in mobile environments. In [4] a *randomized* distributed broadcast protocol which works in $O(D \log n + \log^2 n)$ rounds was given. In 1993 Kushilevitz et al. [5] proven that such an algorithm is optimal. Randomized solutions can, however, be applied only to non time-dependent applications, i.e., when unbounded delays can be tolerated during the broadcast process. A *deterministic* and *distributed* solution has been recently proposed in [6] for multi-hop networks. In this case, however, the broadcast protocol assumes the existence of an underlying multi-cluster architecture (namely, it needs a hierarchical organization of the nodes), and thus, it is not directly suitable for networks without a cellular organization like peer-to-peer networks.

In this paper we are interested in solutions to the broadcast problem which overcome the above mentioned limitations. The protocols we are looking for should therefore have the following characteristics: they are *deterministic*, so that an a priori known bound on the maximum delay of message transmission can be easily determined. They are *delivery-guaranteed*, in that the correct delivery of a message is always guaranteed within a bounded amount of time, and they should be completely *distributed*, i.e., we assume that the nodes in the network do not have an a priori knowledge of the entire network topology and not

*This work was supported in part by the Army Research Office under contract No. DAAG55-96-1-0382.

[1] In order to create a uniform way of comparing these protocols we represent their complexity as the number of rounds required by the protocol for broadcasting a message m. The parameters used for expressing such a complexity measure are n, the number of nodes in the network, and D, the diameter of the network.

even of their neighbors. The transparency to topology changes that derives from the previous properties makes these protocols the required basis for low-level protocols in all those wireless and mobile situations in which network state and control information has to be efficiently disseminated among all the nodes of the network *without* depending on the network state itself. For instance, they can be used in the case of highly reconfigurable networks, and whenever the rate of change of the network topology does not allow an efficient gathering of topological information, such as in solutions that use minimum spanning trees (see, e.g., [7]).

Recently, in [8] it has been proven that any broadcast protocol which satisfies the required properties stated above needs at least $\Omega(D \log n)$ rounds for terminating successfully. On the other hand, it is easy to obtain a delivery guaranteed broadcast protocol that is deterministic and distributed and which work in $O(Dn)$ rounds. In other words, an exponential gap exists between the upper and the lower bound, leaving room to explore new more efficient protocols. In this paper we show how in many cases this gap can be bridged. After presenting a general scheme for distributed broadcast and a simple algorithm that completes the broadcast in $O(Dn)$ rounds (Section 3), in Section 4 we introduce a family of deterministic, delivery-guaranteed and distributed protocols for broadcast which work in $O(D2^h \log^h n)$ rounds, where h, $1 \leq h < \log n$, is such that 2^{h+1} bounds Δ, the maximum degree of a node in the network. These protocols rely on a new method introduced here (Section 4.1) for solving deterministically the problem of the correct delivery of a message on a multi access channel with no need of topology knowledge, acknowledgments or collision feedback mechanisms. For networks with degree bounded by 2^{h+1}, $h \in O(\frac{\log n}{\log \log n})$, they improve the $O(Dn)$ upper bound. Our characterization permits to prove the tightness of two previously proven lower bounds for the broadcast of a message in peer-to-peer networks with specific topologies. All the protocols presented in this paper do not assume any underlying network architecture.

2 PRELIMINARIES

We model a peer-to-peer network by an undirected graph $G = (V, E)$ in which $V = \{p_1, \ldots, p_n\}$ is the set of (radio) nodes and there is an edge $(p_i, p_j) \in E$ if and only if p_j is in the *hearing range* (namely, can hear the transmissions) of p_i and vice versa. In this case we say that p_i and p_j are neighbors. Due to mobility, the graph can change in time. The set of the neighbors of a node p will be indicated by $\Gamma(p)$ and its cardinality, $\delta(p) = |\Gamma(p)|$, is called the *degree* of p. With $\Delta = \max\{\delta(p) : p \in V\}$ we indicate the maximum *degree* of the network G. The *distance* $d(p_i, p_j)$ between two nodes p_i and p_j, $1 \leq i, j \leq n$, is defined as the length of the shortest path (minimum number of hops) between p_i and p_j. The maximum distance between any pair of nodes is called the *diameter* D of the network. Given the source s of a message, all the nodes p such that $d(s,p) = \ell \leq D$ are said to belong to the ℓth *layer* of the network, $0 \leq \ell \leq D$. Every node in the network is assigned a unique ID which we assume denoted 1 through n.

A *deterministic distributed broadcast protocol for peer-to-peer networks* Π is a protocol which is executed *at each node* in the network in the following way: a) Time of execution is considered to be slotted and the time slots, or *rounds*, are numbered $0, 1, \ldots$ At round 0 a specific node s, called the *source*, transmits a message m; b) In each round a node acts either as a transmitter or as a receiver. A node receives a message m in a specific round if and only if in that round it acts as a receiver and *exactly* one of its neighbors acts as a transmitter. In this case m is the same message transmitted by the neighbor; c) The action of a node in a specific round is deterministically determined by its initial input, i.e., its own ID (my_ID), n, and the degree of the network Δ; d) The broadcast is *completed* at round t if all the nodes have correctly received the message m at one of the rounds $0, 1, \ldots, t$.

Thus, the broadcast proceeds according to a *schedule* $L_\Pi = \langle T_1, \ldots, T_t \rangle$, i.e., according to a list of *transmissions* (*transmission sets*) which specifies for each round i the set of nodes which act as (potential) transmitters, $1 \leq i \leq t$. During the broadcast process, the nodes that in a given round have received a message m are said to be *covered* by the broadcast. The nodes that have not received m are said to be *uncovered*. Given a node p, $\Gamma_c(p)$ ($\Gamma_u(p)$) will indicate its (un)covered neighborhood. Finally, a set H of covered nodes is said to be a *conflicting set* if $\cap_{p \in H} \Gamma_u(p) \neq \emptyset$. In other words, if at least two nodes that have received the message have an uncovered neighbor in common, then they form a conflicting set. In a mobile system, conflicting sets are "dynamic" in the sense that their cardinality and the identity of their nodes can depend on the mobility of the nodes. In the case of peer-to-peer networks with mobile nodes, we assume that at least one covered node remains in the hearing range of any neighboring uncovered node, i.e., we require that any node has the possibility to receive the message. (This does not imply that the network has to be static during the entire broadcast process, but it means that in order for each node to receive the message, the network has always to be *connected*.)

3 A BROADCAST SCHEME

The problem of distributed broadcast as stated in the previous section is that of scheduling, in a deterministic way, the transmissions of the covered nodes in order to guarantee the correct delivery of the message independently of the possibility of collisions and without the need of a collision detection mechanism. In order to solve this problem, here we assume that the time axis is divided into units

called (transmission) *frames*. Each frame is, in turn, divided into rounds, numbered 1 through τ, where $\tau > 0$ is the frame *length*. We assume that the nodes are synchronized on a frame base (namely, we assume that each node has a counter which is set to 1 at the beginning of each frame and that is incremented by 1 with each subsequent round) and that the round length is the same for each node.[2] Each node that either generated or received a message m is allowed to transmit it only in certain rounds in a frame that it calculates by means of the following:

PROCEDURE *Round_Numbers*(n, Δ);
begin
Transm:=*Get_The_Rounds*(n, Δ)
end;

The set Transm $\subseteq \{1, \ldots, \tau\}$ will contain the round numbers in which the node is allowed to transmit the message. By specifying the function *Get_The_Rounds* we can get different broadcast protocols.

As soon as a node either has a message m ready for transmission or receives m, it waits for the beginning of a new frame. At that time, it will start to check when it can transmit m. Specifically, the node will test if the current value of the counter belongs to Transm, and when this is the case the node sends m.

A simple inductive argument allows us to prove that the described scheme achieves the broadcast of m in a layer by layer fashion. Thus, the broadcast is completed in $t \in O(D\tau)$ rounds as soon as we can find a suitable function *Get_The_Rounds* which allows us to prove that m is correctly forwarded from a given layer to the subsequent one in the τ rounds of a frame. What we need is a (deterministic) method to distribute the nodes to the transmission sets in a frame in such a way that it is *always* guaranteed that at least one set will contain only *one* node from *any* conflicting set $H \subseteq V$ (i.e., in the round corresponding to that transmission set, no collision will occur). More than that, we want such a method to generate a schedule that is independent of the *local* current conditions of the network, i.e., such that each node has no need to know its current neighbors to have the guarantee of the correct delivery of the message (complete distributivity). Once given the function *Get_The_Rounds* as just described, we obtain for the broadcast the following desirable properties: 1. *Scalability*: Anytime we want to add a new node to the network, each newly inserted node can issue a broadcast message requesting to update the Transm set of each node according to the new value of n. Each node, then, has only to execute the above procedure *Round_Numbers*. 2. *Parallel broadcast*: More than one message issued by different source nodes can be traversing the network. Indeed, the complete distributivity of the method used to generate the broadcast schedule, guarantees the correct reception of a message sent by *any* neighbor of a given node. These neighbors, of course, may have to send different messages. 3. *Mobility*: Being the schedule independent of the current neighborhood of a node, the topology of the network may change without affecting the broadcast process. Moreover, every node which has moved from an uncovered neighborhood to a covered one during the broadcast, must at some time be the neighbor of a node which has already received the broadcasted message m, and will receive m from it using a *failsafe* recovery procedure such as in [10].

One of the simplest possible broadcast algorithm that meets the previous requirements/properties is obtained using the following:

FUNCTION *Get_The_Rounds*(n, Δ);
begin
output my_ID
end;

Each node is allowed to transmit just once in a frame: when the counter value equals its own ID. This simple method generates a layer to layer schedule for which $\tau = n$. Due to the uniqueness of the nodes IDs, it is clear that at most one node will transmit in a round, so that no collision can ever occur. Each time a new node is added to (or removed from) the network, all that the other nodes need to know is the new value of n, in order to modify the frame length value. Such a broadcast protocol Π has a schedule $L_\Pi = \langle T_1, \ldots, T_t \rangle$ such that each T_j, $1 \leq j \leq t$, is a singleton and $t \in O(Dn)$.

In the following section we propose a deterministic distributed broadcast algorithm which maintains the delivery-guaranteed property. This algorithm completes the broadcast in polylogarithmic time, and in *sparse* networks (i.e., in networks with a "small" or with a constant maximum degree Δ) it has to be preferred to the linear protocol just described.

4 POLYLOGARITHMIC BROADCAST

As noticed in Section 3, the problem of the correct forwarding of a message between any two consecutive layers of a wireless network is that of distributing the nodes to the transmission sets in a frame in such a way that in at least a transmission set there are no two nodes from the same conflicting set. Here we present a novel method (*division method*) that, given a non empty set of integers P (the nodes' IDs), distributes the elements of *any* non empty $R \subseteq P$ (a conflicting set) in a family $\mathcal{F}$ of τ subsets of P (the layer to layer schedule; each set corresponds to a different round) in such a way that the delivery-guaranteed property required for the broadcast is always obtained. Put differently, so that there exists at least a (transmission) set $F \in \mathcal{F}$ such that $|F \cap R| = 1$ (in the round corresponding to F only one node from R transmits—in that round the delivery is guaranteed). The method is completely deterministic and constructive and it can be executed *at each node* p in order to get the numbers of the rounds in which p is allowed to transmit.

[2] The synchronization of the nodes can be achieved, for instance, by using GPS—Global Positioning System [9].

Each node needs only to know global information, such as n and Δ.

In the remaining part of this section we first describe the division method, then we present a family of broadcast algorithms based on the method and prove their correctness. For the sake of simplicity, from now on we consider n and $|P|$ powers of 2. We also assume here that only one message m is traversing the network. All logarithms are to be considered to be base 2.

4.1 THE DIVISION METHOD

Consider a non empty set of integers P. In this section we describe a general method for deriving from P a family of subsets of P that *hits* any non empty $R \subseteq P$, i.e., a family $\mathcal{F} \subseteq 2^P$ such that there exists at least a set $F \in \mathcal{F}$ for which $|F \cap R| = 1$. In the following, with the operator $\longrightarrow$ we will partition a set of integers I into two subsets I_1 and I_2 with the same cardinality.

The method is based on the following procedure that given a set of integers P, $|P| \geq 2$, divides P into $2\log|P|$ distinct sets.

```
PROCEDURE Divide(I);
begin
  I ⟶ o_1^1, e_1^1;
  T^1 := o_1^1;
  T^2 := e_1^1;
  for i := 2 to log|I| do
    begin
    T^{2i-1} := T^{2i} := ∅;
    for j := 1 to 2^{i-2} do
      begin
      o_j^{i-1} ⟶ o_{2j-1}^i, e_{2j-1}^i;
      e_j^{i-1} ⟶ o_{2j}^i, e_{2j}^i;
      T^{2i-1} := T^{2i-1} ∪ o_{2j-1}^i ∪ o_{2j}^i;
      T^{2i} := T^{2i} ∪ e_{2j-1}^i ∪ e_{2j}^i
      end
    end
end;
```

The functioning of the *Divide* procedure is explained in the following example.

EXAMPLE 1. Consider $P = \{1, \ldots, 16\}$. The following is the output of the *Divide* procedure called on P. Starting from

$$P = \{\underbrace{1,2,3,4,5,6,7,8}_{o_1^1}, \underbrace{9,10,11,12,13,14,15,16}_{e_1^1}\}$$

we will have:

$$T^1 = \{\underbrace{1,2,3,4}_{o_1^2}, \underbrace{5,6,7,8}_{e_1^2}\} \quad T^2 = \{\underbrace{9,10,11,12}_{o_2^2}, \underbrace{13,14,15,16}_{e_2^2}\}$$

$$T^3 = \{\underbrace{1,2}_{o_1^3}, \underbrace{3,4}_{e_1^3}, \underbrace{9,10}_{o_2^3}, \underbrace{11,12}_{e_2^3}\} \quad T^4 = \{\underbrace{5,6}_{o_3^3}, \underbrace{7,8}_{e_3^3}, \underbrace{13,14}_{o_4^3}, \underbrace{15,16}_{e_4^3}\}$$

$$T^5 = \{\underbrace{1}_{o_1^4}, \underbrace{2}_{e_1^4}, \underbrace{5}_{o_2^4}, \underbrace{6}_{e_2^4}, \underbrace{9}_{o_3^4}, \underbrace{10}_{e_3^4}, \underbrace{13}_{o_4^4}, \underbrace{14}_{e_4^4}\}$$

$$T^6 = \{\underbrace{3}_{o_5^4}, \underbrace{4}_{e_5^4}, \underbrace{7}_{o_6^4}, \underbrace{8}_{e_6^4}, \underbrace{11}_{o_7^4}, \underbrace{12}_{e_7^4}, \underbrace{15}_{o_8^4}, \underbrace{16}_{e_8^4}\}$$

$$T^7 = \{1,3,5,7,9,11,13,15\} \quad T^8 = \{2,4,6,8,10,12,14,16\}.$$

It is easy to verify that for each i, $1 \leq i \leq \log|P|$, (a) $T^{2i-1} \cap T^{2i} = \emptyset$, $T^{2i-1} \cup T^{2i} = P$, (b) $|T^{2i-1}| = |T^{2i}| = |P|/2$. This implies that after h, $1 \leq h \leq \log|P|$, calls of the *Divide* procedure every time on a set of its output, the last output will be sets with cardinality $\frac{|P|}{2^h}$. Furthermore, (c) for each j, $1 \leq j \leq 2^{i-1}$, is $|o_j^i| = |P|/2^i (= |e_j^i|)$. Another useful property of the *Divide* procedure is stated in the following lemma.

Lemma 1 *Given a subset* R *of a set of integers* P, $|R| \geq 2$, *there always exists an* i, $1 \leq i \leq \log|P|$, *such that* R *is partitioned by the procedure call Divide*(P) *into two non empty subsets* R_1 *and* R_2 *such that* $R_1 \subseteq T^{2i-1}$ *and* $R_2 \subseteq T^{2i}$.

Proof We show that as far as $R \subseteq T^{2i-1}(T^{2i})$, $1 \leq i \leq \log|P| - 1$, then there exists a j, $1 \leq j \leq 2^{i-1}$, such that $R \subseteq o_j^i$ (e_j^i) and that as soon as this is no longer true we have the thesis. We proceed by induction on the number i of subsequent partitions of the set P. Let $R \subseteq o_1^1 = T^1$ (the case $R \subseteq e_1^1 = T^2$ is symmetric. The case in which $(R_1 =)$ $R \cap T^1 \neq \emptyset \neq R \cap T^2 (= R_2)$ is obvious). The base case of induction (i = 1) is then trivial. Now, suppose that for a j, $1 \leq j \leq 2^{i-3}$, we have $R \subseteq o_j^{i-1}$ (the case with $R \subseteq e_j^{i-1}$ is analogous). We know that in the next iteration of the main loop in the *Divide* procedure we will have:

$$o_j^{i-1} \longrightarrow o_{2j-1}^i, e_{2j-1}^i,$$

and we have the following two cases: either $R \subseteq o_{2j-1}^i$ (e_{2j-1}^i), or R is partitioned into two non empty subsets R_1 and R_2 such that $R_1 \subseteq o_{2j-1}^i$ and $R_2 \subseteq e_{2j-1}^i$ which implies the thesis ($R_1 \subseteq T^{2i-1}$ and $R_2 \subseteq T^{2i}$).

Notice that the second case always occurs when

$$|o_j^{i-1}| \geq R \geq (|o_j^{i-1}|/2) = |o_{2j-1}^i|$$

(see (c), above), whence the thesis into $\log|P|$ steps. •

Consider now the following function (where $c \geq 1$ is an integer variable that takes values $\leq |I|$ and $h \in \{0, \ldots, \log c\}$):

```
FUNCTION Smash(I);
begin
  if |I| = c/2^h
    then output I
    else
      begin
      Divide(I);
      for j := 1 to 2log|I| do Smash(T^j)
      end
end;
```

EXAMPLE 2. Consider $P = \{1, \ldots, 16\}$ and $c = |P| = 16$. Then, when $h = 0$, the function call *Smash*(P) returns P as output. When $h = 1$ the same call will output the 8 sets as in Example 1. When $h = 2$, we obtain the 48 sets output by the *Divide* procedure successively called on the sets $T^1, \ldots, T^8$. For example, *Divide*(T^1)

gives the 6 sets $T^1 = \{1,2,3,4\}, T^2 = \{5,6,7,8\}, T^3 = \{1,2,5,6\}, T^4 = \{3,4,7,8\}, T^5 = \{1,3,5,7\}, T^6 = \{2,4,6,8\}$; *Divide*$(T^3)$ gives the 6 sets $T^1 = \{1,2,3,4\}, T^2 = \{9,10,11,12\}, T^3 = \{1,2,9,10\}, T^4 = \{3,4,11,12\}, T^5 = \{1,3,9,11\}, T^6 = \{2,4,10,12\}$, and *Divide*$(T^4)$ gives the 6 sets $T^1 = \{5,6,7,8\}, T^2 = \{13,14,15,16\}, T^3 = \{5,6,13,14\}, T^4 = \{7,8,15,16\}, T^5 = \{5,7,13,15\}$ and $T^6 = \{6,8,14,16\}$. ◇

It is easy to verify (see also (b) above) that, in general, for a set of integers P and $c = |P|$, the function call *Smash*(P) outputs $\tau = 2^h \prod_{j=0}^{h-1} \log \frac{|P|}{2^j}$ sets, each one with cardinality $\frac{|P|}{2^h}$, $0 \leq h \leq \log |P|$.[3] Furthermore, when $c = |P|$, the depth of the recursion is h. It is worth noticing that h also indicates the number of subsequent applications of the *Divide* procedure to subsequently halved sets.

We show now that given a non empty set of integers P and $c = |P|$, for each h, $0 \leq h \leq \log |P|$, the function call *Smash*(P) returns a family of subsets of P that hits any non empty set $R \subseteq P$ such that $|R| < 2^{h+1}$.

Theorem 1 *Given a non empty set of integers* P *and the integer* $c = |P|$*, for each* h*,* $0 \leq h \leq \log |P|$*, the Smash function called on* P *returns a family* $\mathcal{F} \subseteq 2^P$ *such that for any non empty set* $R \subseteq P$*,* $|R| < 2^{h+1}$*,* $\mathcal{F}$ *hits* R.

Proof Easily obtained by induction on h, the depth of the recursive calls of the *Smash* function on subsequently halved sets. •

4.2 THE PROTOCOL

The *Get_The_Rounds* function used by each node (by means of the *Round_Numbers* procedure, see Section 3) uses a slightly modified version of the *Smash* function described in the previous section. Instead of returning a set, the following *Find_Rounds* procedure divides the set $P = \{1, \ldots, n\}$ of nodes' IDs into $\tau = 2^h \prod_{j=0}^{h-1} \log \frac{n}{2^j}$ sets, $1 \leq h < \log n$, and checks if the ID of the node that executes the *Round_Numbers* procedure (my_ID) belongs to the resulting sets.

```
FUNCTION Get_The_Rounds(n, Δ);
begin
  c := n;
  h = ⌈log Δ⌉;
  o := 2^{h-1} ∏_{j=1}^{h-1} log n/2^j;
  Temp := ∅;
  Find_Rounds({1, ..., n}, 0, 0);
  output Temp
end;
where
PROCEDURE Find_Rounds(I, x, y);
begin
  if |I| = c/2^h
    then if my_ID ∈ I then
           Temp := Temp ∪ {o(x − 1) + y}
    else begin
      Divide(I);
      if x = 0
        then for j := 1 to 2 log |I|
          do Find_Rounds(T^j, j, 1)
        else for j := 1 to 2 log |I|
          do Find_Rounds(T^j, x, j)
    end
end;
```

[3] When $h = 0$ we stipulate that $\prod_{j=0}^{h-1} \log \frac{|P|}{2^j} = 1$.

EXAMPLE 3. Suppose that node p such that my_ID$= 2$ runs the procedure *Round_Numbers*(16). Then, when $h = 1$, the previous *Get_The_Rounds* function will output the set $\{1,3,5,8\}$, i.e., during a frame of length $\tau = 8$, node p is allowed to transmit the message in rounds $1,2,3$ and 8. Indeed, if we consider $n = 16$, the transmission sets $T_1, \ldots, T_8$ of a frame are the 8 sets output by the *Divide* procedure called on $\{1, \ldots, 16\}$. More precisely: $T_i = T^i$, $1 \leq i \leq 8$ (Example 1). When $h = 2$, we have Transm $= \{1,3,6,7,9,12,13,15,18,43,45,47\}$, i.e., during a frame of length $\tau = 48$, node p is allowed to transmit the message in rounds $1,3,6,7,9,12,13,15,18,43,45$ and 47. The transmission sets $T_1, \ldots, T_{48}$ of the frame are the 48 sets output by the *Divide* procedure called on the sets $T_1, \ldots, T_8$ (Example 2). ◇

Now we prove that the *Get_The_Rounds* function described in this section allows a delivery-guaranteed forwarding of a message m between any two consecutive layers.

Proposition 1 *Consider a wireless network with maximum degree* $\Delta = 2^{h+1} - 1$*,* $1 \leq h < \log n$*, in which each node* p *executes the Round_Numbers procedure with the previous Get_The_Rounds function. Then, in a frame of length* $\tau = 2^h \prod_{j=0}^{h-1} \log \frac{n}{2^j}$*, the message* m *is correctly forwarded between any two consecutive layers.*

Proof The proof that any conflicting set $R \subset P$ is distributed among the transmission sets of a frame in such a way that no collision occurs in at least one round relies on Theorem 1, noticing that the two integer parameters x and y used by the *Find_Rounds* procedure do not interfere with the division method: they just deal with the problem of the correct attribution of the rounds to the node p. As for this last problem, we note that as soon as the *Get_The_Rounds* function calls the *Find_Rounds* procedure with the actual parameters $P = \{1, \ldots, n\}$, $x = 0$ and $y = 0$, the **else** branch of the outermost **if** is executed ($c = n$ and for each h, $1 \leq h < \log n$, $n \neq \frac{n}{2^h}$). Then, being $x = 0$, the **then** branch of the innermost **if** is executed (this is the unique time in the whole recursive execution of the procedure) and the *Find_Rounds* procedure is recursively called on the $2 \log n$ sets returned by the *Divide*$(\{1, \ldots, n\})$ call. If $h = 1$, each one of the $2 \log n$ calls of the *Find_Rounds* procedure (*Find_Rounds*$(T^j, j, 1)$, $1 \leq j \leq 2 \log n$) executes the **then** branch of the outermost **if** and if the membership condition is satisfied, the set Temp (it will be the set Transm of the node p) is updated with a round number. More precisely, in this case the round number is $o(x-1)+y = x-1+1 = x$, and being $x = j$, $1 \leq j \leq 2 \log n$, each one of the $2 \log n$ rounds is correctly assigned.[4] If $h > 1$,

[4] Notice that, when $h = 1$, we have $o = 1$ provided that we

then each one of the $2\log n$ calls of the *Find_Rounds* procedure executes again the **else** branch of the outermost **if**, but this time the condition of the innermost **if** is no longer verified ($x = j$, $1 \leq j \leq 2\log n$) and the successive recursive calls of the *Find_Rounds* procedure are of the form *Find_Rounds* (T^j, x, j), $1 \leq j \leq 2\log|I|$ and $1 \leq x \leq 2\log n$. After $h-1$ such recursive calls, the *Find_Rounds* procedure finally executes the **then** branch of the outermost **if**, and if the membership condition is verified the set Temp is updated. As noticed in Section 4.1, each one of the initial set output by the first call of the *Divide* procedure is in turn divided into $o = 2^{h-1}\prod_{j=1}^{h-1}\log\frac{n}{2^j}$ transmission sets whose round number is $o(x-1)+y$, $1 \leq x \leq 2\log n$ and $1 \leq y \leq 2\log\frac{n}{2^{h-1}}$. The total number of rounds needed is thus $\tau = o2\log n = 2^h\prod_{j=0}^{h-1}\log\frac{n}{2^j}$.

According to the general scheme of Section 3, this section is summed up by the following:

Theorem 2 *The algorithm obtained using the previous Get_The_Rounds function completes the broadcast in* $t \in O(D2^h\log^h n)$ *rounds in wireless networks with maximum degree* $\Delta = 2^{h+1} - 1$, $1 \leq h < \log n$.

Due to the general lower bound on deterministic distributed broadcast protocol in (mobile) wireless networks proven in [8], when $h = 1$ (namely, for wireless networks with maximum degree $\Delta = 3$) the bound proven in Theorem 2 is tight. Furthermore, due to the lower bound on the broadcast of a message in wireless networks with constant diameter presented in [2], when $h = 2$ (namely, for wireless networks with maximum degree $\Delta = 7$) and D is a constant, the presented algorithm is optimal.

When $h = \log n - 1$, i.e., in wireless networks with the maximum possible degree, the above presented algorithm completes the broadcast in $O(Dn\log^{\log n} n)$ rounds, so that the $O(Dn)$ algorithm presented in Section 3 is to be preferred. In general, in wireless networks with $\Delta = 2^{h+1} - 1$, the above family is to be used when $h \in O(\frac{\log n}{\log\log n})$. Indeed, for each integer h, $0 \leq h < \frac{\log n}{\log\log n+1}$, is $(2\log n)^h < n$: starting from $(2\log n)^h < n$ we have $\log(2\log n)^h < \log n$, $h(\log 2\log n) < \log n$, $h(\log\log n + \log 2) < \log n$, $h(\log\log n + 1) < \log n$, and thus $h < \frac{\log n}{\log\log n+1}$. Therefore, being $(2\log n)^h \geq \tau = 2^h\prod_{j=0}^{h-1}\log\frac{n}{2^j}$, $0 \leq h \leq \log n$ (the proof is easily obtained by induction on h), for wireless networks with maximum degree $\Delta = 2^{\lfloor\frac{\log n}{\log\log n+1}\rfloor+1} - 1$, the broadcast algorithms presented in this section is faster than the $O(Dn)$ one.

5 CONCLUSIONS

In this paper we have presented two different algorithms for the deterministic, delivery-guaranteed and distributed broadcast of a message in peer-to-peer networks. The first one is a simple algorithm that completes the broadcast in $O(Dn)$ rounds. The second algorithm is in fact a family of protocols which work in $O(D2^h\log^h n)$ rounds, where h, $1 \leq h < \log n$, is such that 2^{h+1} bounds Δ, the maximum degree of (a node in) the network. When $\Delta = 3$ we obtained an upper bound that matches the general lower bound presented in [8]. Moreover, due to the result in [2] on constant diameter networks, when $\Delta = 7$ and the diameter of the network is a constant, our algorithm is optimal. Obtaining these results became possible by characterizing the broadcast problem as a combinatorial problem for the solution of which we proposed a novel, explicit and completely deterministic method. For networks with degree bounded by 2^{h+1}, $h \in O(\frac{\log n}{\log\log n})$, the corresponding algorithms improve the $O(Dn)$ upper bound. Moreover, our solution does not assume *any* underlying network architecture thus being suitable for pure *ad hoc* networks.

define $\prod_{j=1}^{h-1}\log\frac{n}{2^j} = 1$ (see also Note 3).

REFERENCES

[1] I. Chlamtac and O. Weinstein. The wave expansion approach to broadcasting in multihop radio networks. *IEEE Transactions on Communications*, 39(9):426–433, 1991.

[2] N. Alon, A. Bar-Noy, N. Linial, and D. Peleg. A lower bound for radio broadcast. *Journal of Computer and System Sciences*, 43(2):290–298, October 1991.

[3] I. Gaber and Y. Mansour. Broadcast in radio networks. In *Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 577–585, San Francisco, CA, 22–24 January 1995.

[4] R. Bar-Yehuda, O. Goldreich, and A. Itai. On the time-complexity of broadcast in multi-hop radio networks: an exponential gap between determinism and randomization. *Journal of Computer and System Sciences*, 45(1):104–126, August 1992.

[5] E. Kushilevitz and Y. Mansour. An $\Omega(D\log n/D)$ lower bound for broadcast in radio networks. In *Proceedings of the 12th Annual ACM Symposium on Principles of Distributed Computation*, pages 65–74, Ithaca, New York, 1993.

[6] E. Pagani and G. P. Rossi. Reliable broadcast in mobile multi hop packet networks. In *Proceedings of the The Third Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom'97)*, pages 34–42, Budapest, Hungary, 26-30 September 1997.

[7] I. Chlamtac and S. Kutten. Tree-based broadcasting in multihop radio networks. *IEEE Transactions on Computers*, C-36(10):1209–1223, 1987.

[8] D. Bruschi and M. Del Pinto. Lower bounds for the broadcast problem in mobile radio networks. *Distributed Computing*, 10(3):129–135, April 1997.

[9] E. D. Kaplan, editor. *Understanding GPS : principles and applications*. Artech House, Boston, MA, 1996.

[10] A. Segall and M. Sidi. A failsafe distributed routing protocol for minimum delay routing. *IEEE Transactions on Communications*, Com-29(5):689–695, May 1981.