

# Mobility Tolerant Broadcast in Mobile Ad Hoc Networks

Pradip K Srimani<sup>1</sup> and Bhabani P Sinha<sup>2</sup>

<sup>1</sup> Department of Computer Science, Clemson University, Clemson, SC 29634-0974

<sup>2</sup> Electronics Unit, Indian Statistical Institute, Calcutta 700 035 India

**Abstract.** A new deterministic broadcast protocol for an ad hoc network is proposed in this paper which avoids re-computation of the transmission schedule, even when the topology of the network changes due to the mobility of the nodes. The basic idea is to use a successive partitioning scheme by representing the identifier of each node (an integer) in an arbitrarily chosen radix system; the protocol then computes the specific time slots in which a particular node should transmit its message. The proposed protocol is simple, easy to implement and needs lesser broadcast time than that in [BBC99].

## 1 Introduction

Mobile ad hoc networks are being increasingly used for military operations, law enforcement, rescue missions, virtual class rooms, and local area networks. A mobile multi-hop network consists of  $n$  identical mobile hosts (nodes) with unique identifiers  $0, \dots, n-1$ . These mobile hosts communicate among each other via a packet radio network. When a node transmits (broadcasts) a message, the nodes in the *coverage area* of the sender can simultaneously receive the message. A node  $i$  is called a *neighbor* of node  $j$  in the network if node  $j$  is in the *coverage area* of node  $i$ . This relationship is time varying since the nodes can and do move.

In this paper we consider the important problem of broadcasting in an ad hoc network; broadcast is defined to be a process where a source node transmits a message to be received by every node in the network (this is different from broadcast at the MAC layer wherein we are just trying to reach all of our one-hop neighbors). In our model [BP97,BBC99], the system consists of multi-hop time-slotted radio networks without collision detection mechanism (although collision detection protocols have been proposed and used in radio networks [LM87,BYGI92]). An important characteristic of this model is that the nodes share the same transmission channel; thus, collision is possible when more than one neighbor transmit at the same time slot (round)<sup>1</sup> and correct message reception is prevented since there is no collision detection mechanism; the broadcast protocol itself should guarantee the reception of messages in presence of possible collisions. The model assumes the ad hoc network to be composed of a set of processors which may be stationary or mobile and they communicate in synchronous time slots or *rounds*. At any given time a node  $i$  can correctly receive a message from one of its neighbors, say  $j$ , iff  $j$  is the *only* neighbor of  $i$  transmitting at that time.

---

<sup>1</sup> we use slot and round interchangeably to mean the same thing.

The broadcast problem in multi-hop networks have been extensively studied; various centralized, distributed, deterministic and randomized algorithms have been proposed [BYGI92,LBA93,CK87,Bas98,PR97]; excellent comparisons of these techniques are given in [BP97,BBC99]. Most of these algorithms do not consider the mobility of the nodes in the sense that they do not account for the dynamic topology of the networks in any cost effective way. The authors in [BBC99] first formulated the requirements of efficient broadcast protocols in presence of node mobility. A broadcast protocol must be: (1) *Mobility Independent* (the broadcast must be correctly completed independent of the knowledge of the identities of the neighbors of a node and the mobility of the nodes), (2) *Deterministic* (an a priori upper bound on the broadcast completion time can be ascertained), (3) *Distributed* (the nodes execute the protocol without the knowledge of the topology of the entire network), (4) *Simple* (computational overhead at each node is minimized). The authors in [BBC99] then proposed a general algorithmic scheme to design broadcast protocol where each node can compute its transmission schedule depending only on global network parameters like  $n$ , the number of nodes in the network,  $\mathcal{D}$ , the diameter of the network, and  $\Delta$ , the degree of the network. They also showed that their protocols are optimal in light of the lower bounds established in [BP97].

In this paper, we propose a new deterministic distributed broadcast protocol that completes the broadcast in less time. In section 2, we introduce the system model [BBC99] and describe the new protocol in section 3 using the successive partitioning scheme. Section 4 compares the new protocol with those in [BBC99] while section 5 concludes the paper.

## 2 System Model & Previous Work

A multi-hop ad hoc radio network is modeled by an undirected graph  $G = (V, E)$  where  $V = \{0, 1, \dots, n - 1\}$  is the set of computing nodes and  $E$  is the set of bidirectional edges (an edge exists between two nodes iff they are in the hearing range of each other). The set of neighbors of node  $i$  is denoted by  $N(i)$  and  $\Delta$ , the degree of the network is defined as  $\Delta = \max_{i \in V} |N(i)|$ . The diameter of the network  $\mathcal{D}$  is defined to be  $\mathcal{D} = \max_{i, j \in V} d(i, j)$  where  $d(i, j)$  is defined to be the number of hops between the two nodes  $i$  and  $j$ . There is one distinguished node in the network, called the source  $s$  (which is the initiator of the broadcast message); any node  $i$ ,  $0 \leq d(s, i) = \ell \leq \mathcal{D}$  is said to belong to the *layer*  $\ell$  of the network. A distributed deterministic broadcast protocol is executed at each node and it should have the following characteristics:

- Execution time is discrete; the time axis is divided into *frames*, each frame being made up of  $\tau$  *rounds* (numbered 0 through  $\tau - 1$ , where  $\tau$  is the frame length). The source node  $s$  transmits a message  $m$  before the start of any frame.
- In each round, any node is either a transmitter or a receiver. A node cannot transmit a message unless it has received it. Before receiving the message, every node is set to *receive mode*, and after receiving the message, every node is set always to the *transmit mode*. A node can receive the message  $m$  iff at any round the node acts as a receiver and exactly one of its neighbors transmits the message.

- The transmission schedule of any node  $i$  is a priori computed deterministically by using  $n$ , the identifier (ID) of the node  $i$ , and  $\Delta$  of the network. For this, every node executes the following general protocol, where  $my\_id$  is the identifier of the node.

**Protocol at node  $i$ :**

$find\_my\_slots(my\_id, n, \Delta)$

- The broadcast is complete at round  $t$  of a frame  $f$ , iff all the nodes have been informed of the message  $m$  by round  $t$  of frame  $f$ .

We make the following assumptions about the system of nodes in the ad hoc network [BBC99].

1. Nodes are synchronized on a slot or round basis – each node has a counter which is set to 0 at the beginning of each frame and is incremented by 1 at each subsequent round.
2. When a node receives a message  $m$ , it waits for the beginning of a new frame. At that time, the counter is incremented at each time slot or round and the node transmits according to its pre-determined transmitting slots in the frame.
3. The nodes which have received the message during the broadcast process are said to be *covered* by the broadcast, and those which have not yet received the message are called *uncovered*. At any phase of the broadcast, the sets of covered and uncovered neighbors of a given node  $id$  will be denoted by  $N_c(id)$  and  $N_u(id)$ , respectively. A set  $C$  of covered nodes is termed as a **conflicting set** if there is at least one neighbor common to all the nodes in  $C$  that has not yet received the message. It is assumed that at least one node from a conflicting set remains in the hearing range of any neighboring uncovered node. Also, the network never gets disconnected.

*Remark 1.*

1. The above scheme does the broadcast in a layer by layer fashion, i.e., all nodes at layer  $\ell$ ,  $1 \leq \ell \leq \mathcal{D} - 1$ , become informed of the message  $m$  before any node at layer  $\ell + 1$ .
2. The broadcast is complete in at most  $\mathcal{D} \times \tau$  rounds.
3. The procedure to compute the transmission slots for each of the nodes should be independent of the identity of the current neighbors of the node.

### 3 Proposed Approach

Each node transmits (broadcasts) messages only in some specific time slots. We need to specify these time slots for each node so as to guarantee that no matter what the network topology is, eventually every node would receive the broadcast message from only one single node during at least one such time slot. Every node identifies these transmission time slots by executing the protocol  $find\_my\_slots(my\_id, n, \Delta)$  which is done in the following way. The set of nodes  $V$  is partitioned in some disjoint blocks following some rule so that any given pair of nodes will be partitioned in two different blocks (call this as

level 1 partitioning of  $V$ ). If  $\Delta$ , the maximum number of neighbors of a node is 2, then we associate a time slot (round) to each of these partition blocks, meaning thereby that every node in a block will transmit its message during its assigned time slot or round. This guarantees that a given pair of neighbors of a node will transmit at two different rounds, and thus, we are done for  $\Delta = 2$ . If, however,  $\Delta \geq 2$ , then each of the above partition blocks (having size smaller than  $n$ ) is further partitioned in disjoint blocks following the same rule (call this as level 2 partitioning of  $V$ ). If  $\Delta \leq 7$ , then we would show that associating a unique time slot to each of these partition blocks generated after level 2 partitioning of  $V$  would guarantee that there will be at least one time slot during which only one of the neighbors of a given node will be transmitting. In general, if  $\lfloor \log_2 \Delta \rfloor = h$ , then we successively generate level  $i$  ( $2 \leq i \leq h$ ) partitions of  $V$  from the blocks of level  $i - 1$  partition, assign a unique time slot to each of the generated blocks after level  $h$  partition. We would show, in what follows, that this guarantees at least one round in each frame during which every node would have only one of its  $\Delta$  neighbors transmitting the message.

### 3.1 Successive Partitioning Scheme

The successive partitioning scheme is based on radix encoding of the node IDs in the set  $V$  (integers 0 through  $n - 1$ ). We assume a radix  $r$ ,  $r \geq 2$  and we assume that  $n = r^m$ , for some integer  $m$ , to simplify the discussions. Each node  $id \in V = \{0, 1, 2, \dots, n - 1\}$ , is converted into an  $m$ -digit code in radix  $r$  number system as  $id = d_{m-1} d_{m-2} \dots d_0$ , where  $0 \leq d_i \leq r - 1$ , for all  $i$ ,  $0 \leq i \leq m - 1$ . Consider the  $d_i$  values of each node for a specific  $i$ ; the set  $V$  is partitioned into  $r$  disjoint blocks  $B_i(j)$ ,  $0 \leq j \leq r - 1$ , i.e.,  $d_i(id) = j \Rightarrow id \in B_i(j)$ . Since there are  $m$  different digits ( $m$  different values of  $i$ ), we get  $m$  different partitions of  $V$ , each into  $r$  different blocks. Based on the value of  $i$ -th digit in each  $id \in V$ , we partition  $V$  in  $r$  different disjoint blocks. Thus, if  $d_i(id) = j$ ,  $j = 0, 1, \dots, r - 1$ , then we place  $id$  in the block  $B_i(j)$  of the partition. Note that  $V$  can be partitioned in  $m$  such ways, each induced by one digit position  $i$ ,  $0 \leq i \leq m - 1$ . The total number of blocks generated by these partitions is  $rm$ . A given  $id$  value is a member of exactly  $m$  of these blocks.

**Definition 1.** *The blocks  $B_i(j)$ ,  $0 \leq i \leq m - 1$ ,  $0 \leq j \leq r - 1$  will be called the blocks of level-1 partitioning, and are denoted by  $P(1, k)$ ,  $k = 0, 1, \dots, rm - 1$ , where  $P(1, k) = B_i(j)$  if  $k = ri + j$ . Each block  $P(1, k)$ ,  $0 \leq k \leq rm - 1$ , has exactly  $n/r$  elements (nodes). Henceforth, we use only the  $P$  notation to denote the partition blocks; the  $B$  notation was introduced to show the intuitive physical significance of the partitioning blocks.*

*Example 1.* Let  $n = 64$  and  $r = 4$  (See Figure 1). Hence,  $m = 3$ , i.e., each of the 64  $id$  values will be encoded in a 3-digit code in radix 4 system. The 0-th digit position (least significant digit) induces the partition of  $V$  in four blocks, e.g.,  $B_0(0)$ ,  $B_0(1)$ ,  $B_0(2)$  and  $B_0(3)$ , where  $B_0(0) = (0, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60)$ ,  $B_0(1) = (1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 61)$ ,  $B_0(2) = (2, 6, 10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50, 54, 58, 62)$ , and  $B_0(3) = (3, 7, 11, 15, 19, 23, 27, 31, 35, 39, 43, 47, 51, 55, 59, 63)$ . Similarly, blocks  $B_1(0)$ ,  $B_1(1)$ ,  $B_1(2)$  and  $B_1(3)$  (induced by the

next significant digit position) are given by  $B_1(0) = (0, 1, 2, 3, 16, 17, 18, 19, 32, 33, 34, 35, 48, 49, 50, 51)$ ,  $B_1(1) = (4, 5, 6, 7, 20, 21, 22, 23, 36, 37, 38, 39, 52, 53, 54, 55)$ ,  $B_1(2) = (8, 9, 10, 11, 24, 25, 26, 27, 40, 41, 42, 43, 56, 57, 58, 59)$ , and  $B_1(3) = (12, 13, 14, 15, 28, 29, 30, 31, 44, 45, 46, 47, 60, 61, 62, 63)$ . Similarly,  $B_2(0) = (0, 1, 2, \dots, 15)$ ,  $B_2(1) = (16, 17, 18, \dots, 31)$ ,  $B_2(2) = (32, 33, 34, \dots, 47)$ , and  $B_2(3) = (48, 49, 50, \dots, 63)$ . Thus, the level-1 partitioning is computed as  $P(1,0) = B_0(0)$ ,  $P(1,1) = B_0(1)$ ,  $P(1,2) = B_0(2)$ ,  $P(1,3) = B_0(3)$ ,  $P(1,4) = B_1(0)$ ,  $P(1,5) = B_1(1)$ ,  $P(1,6) = B_1(2)$ ,  $P(1,7) = B_1(3)$ ,  $P(1,8) = B_2(0)$ ,  $P(1,9) = B_2(1)$ ,  $P(1,10) = B_2(2)$ ,  $P(1,11) = B_2(3)$ .  $\square$

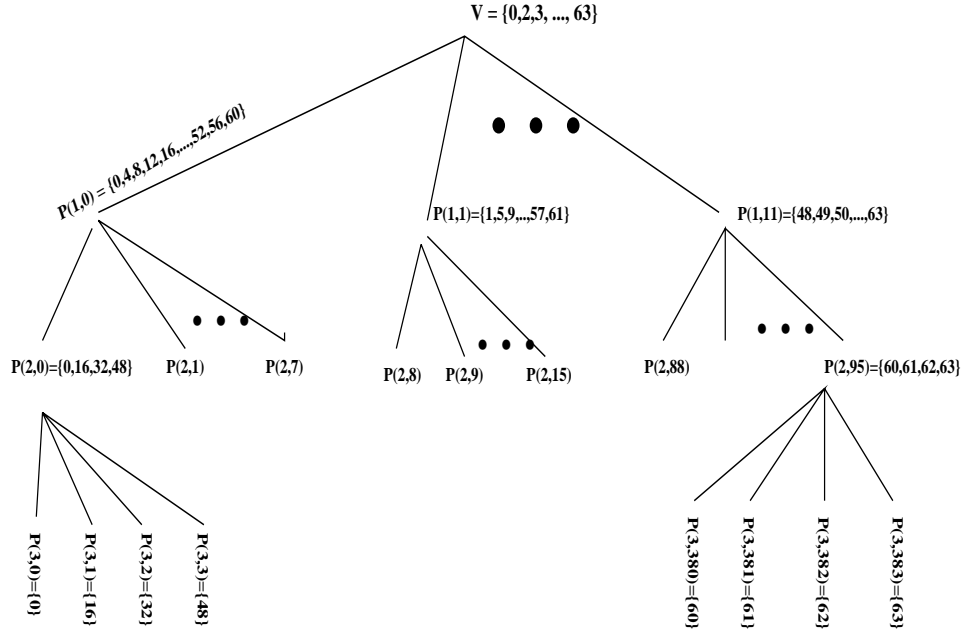


Fig. 1. Partition Tree for  $n = 64$  and  $r = 4$

**Lemma 1.** For an arbitrary subset  $R$  of the set  $V$ , there exists at least one  $P(1, k)$ ,  $0 \leq k \leq rm - 1$ , which contains at most  $\lfloor |R|/2 \rfloor$  elements of  $R$ .

*Proof.* Consider the  $m$ -digit radix  $r$  codes of the elements in  $R$ . Since all these codes are distinct, there exists at least one digit position which consists of at least two different values  $v_1$  and  $v_2$ ,  $0 \leq v_1, v_2 \leq r - 1$  in all these codes. This means that the elements of  $R$  will be placed in at least two different blocks of the partitions at level 1. One of these blocks must contain at most  $\lfloor |R|/2 \rfloor$  elements.

**Lemma 2.** Each element in any arbitrary block  $P(1, k)$ ,  $0 \leq k \leq rm - 1$  of level-1 partition can be uniquely re-coded by using only  $m - 1$  digits in radix  $r$  number system.

*Proof.* Let  $j = \lfloor k/r \rfloor$ . It follows from the definition that the block  $P(1, k)$  is induced by  $j$ -th digit,  $d_j$ , ( $0 \leq j \leq m-1$ ) of the radix- $r$  codes for the elements in  $V$ . Consider an element  $id \in P(1, k)$ . Let the  $m$ -digit code for  $id$  be  $d_{m-1}d_{m-2} \cdots d_{j+1}d_j d_{j-1} \cdots d_0$ . Since  $\forall id \in P(1, k)$ , the value of  $d_j$  is same, it does not play any role in distinguishing the elements in  $P(1, k)$ . Thus, if we have a new  $(m-1)$ -digit code for each  $id$  in  $P(1, k)$  given by  $d_{m-1}d_{m-2} \cdots d_{j+1}d_{j-1} \cdots d_0$ , by removing the  $j$ -th digit in each code, then these  $m-1$ -digit codes are sufficient to uniquely identify the elements in  $P(1, k)$ . It follows from this process of re-coding that each element  $id \in P(1, k)$  is mapped to a unique element  $id' \in V_1 = \{0, 1, 2, \dots, n/r - 1\}$  by the following one-to-one and onto mapping:  $id' = (\lfloor \alpha/r^{j+1} \rfloor) \times r^j + \alpha \text{ MOD } r^j$ , where  $\alpha = id - d_j \times r^j$ .

*Example 2.* In Example 1 ( $n = 64$  and  $r = 4$ ), consider  $P(1, 7) = (12, 13, 14, 15, 28, 29, 30, 31, 44, 45, 46, 47, 60, 61, 62, 63)$ . We have  $j = \lfloor 7/4 \rfloor = 1$ . Considering the  $m$ -digit radix- $r$  codes of each of these  $ids$  in  $P(1, 7)$ , we see that  $d_1 = 3$  for all these ids. Hence the set of  $\alpha$  values for these elements are  $\{0, 1, 2, 3, 16, 17, 18, 19, 32, 33, 34, 35, 48, 49, 50, 51\}$ . The set of  $id'$  values for these elements is  $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$ , which can be uniquely identified by 2-digit radix-4 codes as follows: 12: (0 0), 13: (0 1), 14: (0 2), 15: (0 3), 28: (1 0), 29: (1 1), 30: (1 2), 31: (1 3), 44: (2 0), 45: (2 1), 46: (2 2), 47: (2 3), 60: (3 0), 61: (3 1), 62: (3 2), 63: (3 3).  $\square$

Consider any one of the blocks  $P(1, k)$ ,  $0 \leq k \leq rm - 1$ , where each element (integer) has a unique  $r$ -radix code using  $(m-1)$  digits,  $d_{m-2}d_{m-3} \cdots d_1d_0$ . We apply the partitioning process on each of these blocks using the digits similar to the level-1 partitioning we applied to the entire set  $V$ . We call this level-2 partitioning.

**Definition 2.** *If we partition the elements of  $P(1, i)$ ,  $0 \leq i \leq rm - 1$ , into  $r$  blocks in  $(m-1)$  different ways, each induced by a digit of the new  $(m-1)$ -digit code, the new blocks are called the blocks of level 2 partitioning. If a level-2 block is obtained from  $P(1, k)$  induced by its  $j$ -th digit code  $d_j$ ,  $0 \leq j \leq m-2$ , such that  $d_j = \beta$ ,  $0 \leq \beta \leq r-1$ , we denote this block by  $P(2, k)$ , where  $k = r \times (m-1) \times i + r \times j + \beta$ .*

*Example 3.* Consider the previous example again. We have  $r(m-1) = 8$ . Hence the blocks from level-2 partitioning are given as  $P(2, 56) = (12, 28, 44, 60)$ ;  $P(2, 57) = (13, 29, 45, 61)$ ;  $P(2, 58) = (14, 30, 46, 62)$ ;  $P(2, 59) = (15, 31, 47, 63)$ ;  $P(2, 60) = (12, 13, 14, 15)$ ;  $P(2, 61) = (28, 29, 30, 31)$ ;  $P(2, 62) = (44, 45, 46, 47)$ ;  $P(2, 63) = (60, 61, 62, 63)$ .  $\square$

*Remark 2.*

1. Each block  $P(1, k)$  of level-1 partition generates  $r(m-1)$  blocks of level-2 partition. The total number of blocks in level-2 partition is given by  $r^2m(m-1)$ .
2. Each level-2 block,  $P(2, k)$ ,  $0 \leq k \leq r^2m(m-1) - 1$  is of size  $n/r^2$ .
3. An arbitrary node  $id \in V$  will belong to exactly  $m(m-1)$  blocks of level-2 partition.

**Lemma 3.** *Given any subset  $R$  of  $V$  such that  $|R| \leq 2n/r$ , there exists a block  $P(2, k)$ ,  $0 \leq k \leq r^2m(m-1) - 1$  which contains at most  $\lfloor |R|/4 \rfloor$  elements of  $R$ .*

*Proof.* By Lemma 1, there exists a partition  $P(1, k)$ ,  $0 \leq k \leq rm - 1$  which contains at most  $\lfloor |R|/2 \rfloor \leq n/r$  elements of  $R$ . This level-1 block  $P(1, k)$  is now subjected to level-2 partitioning and by similar arguments as in the proof of Lemma 1, we get a level-2 block of size  $\lfloor |R|/4 \rfloor$ .

The partitioning process can now be generalized. We generate blocks of level- $(i+1)$  partition from the blocks of level- $i$  partition,  $i \geq 1$ , as follows. For each element  $id \in P(i, k)$ , we compute  $j = \lfloor k/r \rfloor$  and  $\alpha = id - d_j \times r^j$ , where  $d_{m-i-1} \dots d_0$  is the  $(m-i)$ -digit radix- $r$  code of the element  $id$ . We then map  $id$  to  $id' = (\lfloor \alpha/r^{j+1} \rfloor) \times r^j + \alpha \text{ MOD } r^j$ , so that all the elements in  $P(i, k)$  are now mapped to a set of integers  $\{0, 1, \dots, r^{m-i} - 1\}$  via a one-to-one and onto mapping. These  $id'$  values, corresponding to the elements of  $P(i, k)$ , are used to re-code the elements of  $P(i, k)$  into  $(m-i-1)$ -digit radix- $r$  numbers to generate the blocks of level- $(i+1)$  partition. This successive partitioning scheme is illustrated by a *partition tree* in which  $V$  is the root with blocks  $P(1, k)$  as its children, and at every successive level  $i$ ,  $P(i, k')$  is a child of  $P(i-1, k)$ ,  $1 < i \leq m$ , iff  $P(i, k')$  is obtained from the block  $P(i-1, k)$ . An example partition tree for  $r = 4$  and  $n = 64$  is shown in Figure 1. Note that at level  $i$  (root considered to be at level  $i = 0$ ), we get the blocks  $P(i, k)$ ,  $0 \leq k \leq r^i \times m \times (m-1) \times \dots \times (m-i+1) - 1$  corresponding to level- $i$  partitioning. As before, blocks  $P(i+1, k')$  obtained from  $P(i, k)$ ,  $i \geq 1$ , are numbered with  $k'$  from  $r(m-i)k$  to  $r(m-i)(k+1) - 1$ . The partitioning process is formalized as shown in Figure 2; note that we assume the availability of a simple primitive (procedure) *radix\_code(value, q, r, D)* where *value*,  $q$  and  $r$ , are integer inputs, and  $D$  is a  $q$ -dimensional output array of integers; it converts the integer “*value*” ( $0 \leq \text{value} \leq r^q - 1$ ) to a  $q$ -digit radix- $r$  number and stores the digits in locations  $D(0)$  through  $D(q)$  (from least significant digit to most significant digit).

**Lemma 4.** *Given an arbitrary subset  $R$  of  $V$  such that  $|R| \leq n \times (\frac{2}{r})^i$  for some  $i, i \geq 0$ , there exists a block  $P(i+1, k)$ , at level- $(i+1)$  partitioning, that contains at most  $\lfloor |R|/2^{i+1} \rfloor$  elements of  $R$ .*

*Proof.* Since  $|R| \leq n/(r/2)^i, \forall i \geq 0$ , we can apply level-1 partitioning on elements of  $R$  to get at least one block, say  $R_s$ , of size at most  $(n/2)/(r/2)^i$ . If  $i \geq 1$ , then  $(n/2)/(r/2)^i \leq n/r$ , (for  $r \geq 2$ ) and hence, we can apply level-2 partitioning on block  $R_s$ . We use induction on  $j, 1 \leq j \leq i+1$  to show that it is possible to successively apply level-1 through level- $(i+1)$  partitioning on the smallest block at each step. The induction hypothesis is true for  $j = 1$  and  $j = 2$ . Assume it holds up to level- $j$  partitioning. Hence, after level  $j$  partitions ( $j < i+1$ ), we get at least one block, say  $R'_s$ , which contains at most  $|R|/2^j$  elements of  $R$ . Now  $|R'_s| \leq 2^{i-j}n/r^i \leq n/r^j$ , since  $2^{i-j} \leq r^{i-j}$ . It is now possible to apply level  $(j+1)$  partitions on  $R'_s$ . Thus, the induction hypothesis is verified to be true for all values of  $j$  up to  $i+1$ .

**Theorem 1.** *Given any arbitrary subset  $R$  of  $V$ ,  $2^h \leq |R| \leq 2^{h+1} - 1, h \geq 0$  and  $n \geq r^h(2^{h+1} - 1)/2^h$ , there exists at least one block of level- $j$  partitioning,  $j \leq h$ , which contains only one element of  $R$ .*

*Proof.* It follows from the previous discussions that every time we apply the partitioning scheme corresponding to a specific level on a given set of elements, there will be at least

```

Procedure find_partition (value, i, h, offset, Flag) ;
  begin
    radix_code(value, m - i + 1, r, D);
    for j := 0 to m - i do
      begin
        temp ← r × (j - 1) + D(j);
        set Flag(i, offset + temp) ← 1;
        if i ≤ h - 1 then
          begin
            value ← value - D(j) × rj;
            value ← (value DIV rj+1) × rj + value MOD rj;
            offset ← r × (m - i) × temp;
            find_partition(value, i + 1, h, offset, Flag);
          end;
        end
      end
    end
  end procedure;

```

**Fig. 2.** Partitioning Algorithm

one block with no more than half the number of elements in the given set. Given that  $2^h \leq |R| \leq 2^{h+1} - 1$ , it follows that  $h$  many levels of partitioning are sufficient. From Lemma 4, we see that for  $h$  successive levels of partitioning to be applied to the smallest block derived from the previous partitioning step,  $|R| \leq 2^h n / r^h$ , i.e.,  $2^{h+1} - 1 \leq 2^h n / r^h$ . Hence the theorem.

*Example 4.* Let  $n = 64$ ,  $r = 4$  and  $R = \{0, 1, 2, 4, 5, 6, 16, 17, 18, 20, 22, 32, 33, 34, 36\}$ . Since there are 15 elements in  $R$ ,  $h = 3$  for this example. Applying level-1 partitioning, the partition blocks derived from  $R$  are : (0, 4, 16, 20, 32, 36), (1, 5, 17, 33), (2, 6, 18, 22, 34); (0, 1, 2, 16, 17, 18, 32, 33, 34), (4, 5, 6, 20, 22, 36); (0, 1, 2, 4, 5, 6), (16, 17, 18, 20, 22), (32, 33, 34, 36). The smallest cardinality of all these blocks is 4. Consider a smallest block, say (32, 33, 34, 36). The elements in this block will first be mapped as  $32 \rightarrow 0$ ,  $33 \rightarrow 1$ ,  $34 \rightarrow 2$ , and  $36 \rightarrow 4$ . Then these will be recoded in radix 4 system as follows: 32: (0 0), 33: (0 1), 34: (0 2), 36: (1 0). Level-2 partitioning then generates the following blocks from (32, 33, 34, 36): (32, 36), (33), (34); (32, 33, 34), (36). Thus, another level of partitioning is not even necessary to generate a singleton block from  $R$ . □

### 3.2 Transmission Schedule

We use the successive partitioning scheme, developed in the previous section, to design the transmission schedule of an arbitrary node in an ad hoc network. Consider a given node with identification number *my\_id*; the node knows only its own identification number and  $n$ , total number of nodes in the network and  $\Delta$ , the degree of the network. Given  $\Delta$ , we compute  $h$  such that  $2^h \leq \Delta < 2^{h+1}$ . The node then computes the

blocks of the level- $h$  partitioning,  $P(h, k)$ ,  $0 \leq k \leq r^h \frac{m!}{(m-h)!} - 1$ . In each frame, the node with identification number  $my\_id$  will transmit during a time slot or round  $k$  iff  $my\_id \in P(h, k)$ . The details of the pseudo-code for the protocol to be used at each node to compute its own transmission slot or round is given in Figure 3. Note that  $Flag[]$  is a two dimensional array and  $T[]$  is a one-dimensional array;  $Flag(i, j)$  is set to 1 iff  $my\_id$  value of a node belongs to the block  $P(i, j)$ , and  $T(k)$  is set to 1 iff  $Flag(h, k)$  is set to 1.  $Flag$  and  $T$  are both initialized to all zero values. The variable  $offset$  is used for properly numbering the partition blocks at any level.

```

procedure compute_slot (Flag, h, T);
  begin
    for  $k := 0$  to  $[r^h \times m \times (m-1) \times \dots \times (m-h+1) - 1]$  do
      if  $Flag(h, k) = 1$  then  $T(k) \leftarrow 1$ ;
    end
  end procedure;
procedure find_my_slots(my_id, n,  $\Delta$ );
  begin
     $h = \lfloor \log_2 \Delta \rfloor$ ;
     $offset \leftarrow 0$ ;
     $Flag \leftarrow 0$ ; /*initializes the two-dimensional array  $Flag(i, k)$  */
     $T \leftarrow 0$ ; /*initializes the one-dimensional array  $T(i)$  to all zero values */
    find_partition(my_id, 1, h, offset, Flag);
    compute_slot(Flag, h, T);
  end procedure

/* code to be executed by each node having my_id as its identifier */

begin
   $h = \lfloor \log_2 \Delta \rfloor$ ;
   $max\_slot\_number \leftarrow r^h \times m \times (m-1) \times \dots \times (m-h+1) - 1$ 
  find_my_slots(my_id, n,  $\Delta$ );
  for frame_number := 1 to  $\mathcal{D} - 1$  do
    for  $i := 0$  to  $max\_slot\_number$  do
      if  $T(i) = 1$  then transmit the message;
    end
  end.

```

**Fig. 3.** Transmission protocol at each node

**Theorem 2.** Consider a node  $x$  which has not received the message at the beginning of a frame and there is at least one node  $y$ , that has the message, among the set  $R$  of neighbors of node  $x$ . During this frame, there exists at least one time slot  $T(k)$ ,  $0 \leq k \leq r^h \frac{m!}{(m-h)!} - 1$ , when exactly one of the informed neighbors of node  $x$  will transmit the message and node  $x$  will receive it correctly (without collision).

*Proof.* Let  $R$  be the set of neighbors of node  $x$  that are informed (i.e., already have the message to transmit) at the beginning of the frame. Clearly,  $|R| \leq \Delta$ . Consider the

blocks  $P(h, k)$ ,  $0 \leq k \leq r^h \frac{m!}{(m-h)!} - 1$ , generated by level- $h$  partitioning of the vertex set  $V$ ; by Theorem 1, there exists at least one  $k$  such that block  $P(h, k)$  contains only one element of the set  $R$ . So, during time slot  $T(k)$ , exactly one node from the set  $R$  will transmit and node  $x$  will correctly receive the message.

Consider the entire broadcast process. The source node  $s$  transmits the message. All its neighbors, set to *receive mode*, receive the message from  $s$ . Suppose the frames start after this. That means, all nodes of layer 1 have the message received at the beginning of frame 1. These nodes will then be set to *transmit mode* so that they would now transmit the message in their respective time slots of frame 1 as specified above. Since any node in layer 2 will have at most  $\Delta$  neighbors in layer 1, it follows that after the completion of frame 1 transmissions, all nodes in layer 2 will receive the message. This process continues for the successive layers so that at the beginning of frame  $i$ ,  $i \geq 2$ , all nodes in layer  $i$  have received the message successfully and they are ready to transmit the message during frame  $i$  at their respective time slots. By theorem 2, it follows that all nodes in layer  $i + 1$  will receive the message at the end of frame  $i$ . Thus, at the end of frame  $\mathcal{D} - 1$ , all nodes of the layer  $\mathcal{D}$  of the network will receive the message, and the broadcast is complete. Hence, we have the following result.

**Theorem 3.** *The broadcast process completes in  $(\mathcal{D} - 1)$  frames consisting of a total of  $(\mathcal{D} - 1) \times \tau$  rounds (where  $\tau = r^h \frac{m!}{(m-h)!}$ ), or in  $\mathcal{O}(\mathcal{D}r^h \log_r^h n)$  time.*

*Remark 3.* In light of the  $\Omega(\mathcal{D} \log n)$  lower bound [BP97] for deterministic distributed broadcast protocol in mobile multi-hop networks, our bound as given in Theorem 3 is tight when  $h = 1$ , i.e., for networks with  $\Delta = 3$ .

*Remark 4.* It is to be noted that for radix  $r = 2$  the frame length needed by our protocol (Theorem 3) is exactly the same as that needed by the protocol *Division* of [BBC99].

**Theorem 4.** *Our protocol is resilient to node mobility as long as mobility assumption (section 2, assumption 3) is valid.*

*Proof.* By assumption, when the nodes are mobile, the network never gets disconnected, and at least one node from a conflicting set always remains in the hearing range of any neighboring uncovered node. Now, consider the situation after frame  $i$ ,  $i \geq 1$ . At this stage, because of the layer by layer fashion of broadcast, the nodes in the conflicting sets constitute the last layer of nodes which have so far received the message. The neighboring uncovered nodes of each conflicting set would be a member of the next layer of nodes.

## 4 Comparison with Protocols in [BBC99]

Authors in [BBC99] have proposed two protocols for deterministic distributed broadcast in multi-hop mobile ad hoc networks. Our purpose in this section is to demonstrate the relative superiority of our proposed protocol over these two existing protocols. Note that all three protocols operate under exactly the same system model; specifically each of these protocols executes the broadcast in a layer by layer fashion using the same

		values of $\tau$ for						values of $\tau$ for					
$n$	$r$	$\lceil \log_r n \rceil$	$\Delta=3$	$\Delta=7$	$\Delta=15$	$\Delta=31$	$n$	$r$	$\lceil \log_r n \rceil$	$\Delta=3$	$\Delta=7$	$\Delta=15$	$\Delta=31$
256	2	8	<b>16</b>	224	2688	26880	16384	2	14	28	728	17472	384384
256	3	6	18	270	3240	29160	16384	3	9	<b>27</b>	<b>648</b>	13608	244944
256	4	4	<b>16</b>	<b>192</b>	<b>1536</b>	<b>6144</b>	16384	4	7	28	672	<b>13440</b>	<b>215040</b>
256	5	4	20	300	3000	15000	16384	5	7	35	1050	26250	525000
256	6	4	24	432	5184	31104	16384	6	6	36	1080	25920	466560
256	7	3	21	294	2058	--	16384	7	5	35	980	20580	288120
256	8	3	24	384	3072	--	16384	8	5	40	1280	30720	491520
512	2	9	<b>18</b>	288	4032	48384	65536	2	16	32	960	26880	698880
512	3	6	<b>18</b>	<b>270</b>	3240	29160	65536	3	11	33	990	26730	641520
512	4	5	20	320	3840	30720	65536	4	8	<b>32</b>	<b>896</b>	<b>21504</b>	<b>430080</b>
512	5	4	20	300	<b>3000</b>	<b>15000</b>	65536	5	7	35	1050	26250	525000
512	6	4	24	432	5184	31104	65536	6	7	42	1512	45260	10888640
512	7	4	28	588	8232	57624	65536	7	6	42	1470	41160	864360
512	8	3	24	384	3072	--	65536	8	6	48	1920	61440	1474560
1024	2	10	<b>20</b>	360	5760	80640	4194304	2	22	44	1848	73920	2808960
1024	3	7	21	378	5670	68040	4194304	3	14	<b>42</b>	<b>1638</b>	<b>58968</b>	<b>1945944</b>
1024	4	5	<b>20</b>	<b>320</b>	<b>3840</b>	<b>30720</b>	4194304	4	11	44	1760	63360	2027520
1024	5	5	25	500	7500	75000	4194304	5	10	50	2250	90000	3150000
1024	6	4	24	432	5184	31104	4194304	6	9	54	2592	108864	3919104
1024	7	4	28	588	8232	57624	4194304	7	8	56	2744	115248	4033680
1024	8	4	32	768	12288	98304	4194304	8	8	64	3584	172032	6881280

**Table 1.** Number of Rounds  $\tau$  in a Frame for Different  $n, r, \Delta$

frame length at each layer; thus the frame length  $\tau$  is used as the performance metric for the protocols. The first protocol of [BBC99], called *Simple* does not use  $\Delta$ , the degree of the network; thus the frame length does not depend on  $\Delta$ ; the frame length is always  $n$ . The protocol *Division* of [BBC99] does use  $\Delta$  as does our proposed protocol; the frame length required by *Division* is same as that needed by our protocol when we choose  $r = 2$  (Remark 4). Our proposed protocol has the added advantage of tuning the value of  $r$  to suit a particular value of given  $\Delta$  in order to reduce the frame length. Table 1 shows the required number of rounds ( $\tau$ ) per frame by the proposed scheme for different values of  $n, r$  and  $\Delta$  (the values of  $\tau$  shown for  $r = 2$  are also equal to the number of rounds per frame needed by the *Division* protocol of [BBC99]). The minimum value of  $\tau$  for a given value of  $n$  and  $\Delta$  (over possible choices of  $r$ ) is shown by bold entries in the table. Out of these minimum values, those which are smaller than

the corresponding values of  $n$ , are also boxed by rectangles. We make the following observations:

- As noted in [BBC99], the *Division* protocol remains better than the *Simple* protocol as long as  $h < \log_2 n / (\log_2 \log_2 n + 1)$  (since for larger  $h$  values the frame length,  $\tau$ , exceeds  $n$ ); note that this bound on  $h$  is approximate and not exact. Our protocol remains better than the *Simple* protocol as long as  $h < \log_r n / (\log_r \log_r n + 1)$  (the bound is approximate, not exact) by suitably choosing  $r$ . For higher values of  $n$ , the range of  $h$  and hence that of  $\Delta$  over which the protocol remains better than *Simple* is larger in our protocol. For example, for  $n = 16384$ , and  $\Delta = 15$  (corresponding to  $h = 4$ ), frame length  $\tau$  for our protocol is 13440 ( $< n$ ) (by choosing  $r = 4$ ) compared to 17472 for the *Division* protocol ( $r = 2$  in our protocol).
- For lower values of  $\Delta$  (corresponding to when  $h < \log_r n / (\log_r \log_r n + 1)$  after proper choice of  $r$ ), frame length  $\tau$  in our protocol is always less than or equal to that in the *Division* protocol and in most cases substantially less. For example, for  $n = 256$  and  $\Delta = 7$ , the minimum  $\tau$  is 192 (for  $r = 4$ ) which is smaller than the corresponding value of 224 for the *Division* protocol. This improvement is more prominent for larger values of  $n$  and  $\Delta$ . For example, for  $n = 4194304$  and  $\Delta = 31$ , the minimum value of  $\tau$  in our method is 1945944 ( $r = 3$ ), while the number of rounds needed in the *Division* is 2808960.

## 5 Acknowledgement

Srimani's work was supported by an NSF Award # ANI-0219485.

## References

- [Bas98] S. Basagni. *On the Broadcast and Clustering Problems in Peer-to-Peer Networks*. PhD thesis, University degli Studi di Milano, Milano, Italy, May 1998.
- [BBC99] S. Basagni, D. Bruschi, and I. Chlamtac. A mobility-transparent deterministic broadcast mechanism for ad hoc networks. *IEEE Transactions on Networking*, 7(6):799–807, December 1999.
- [BP97] D. Bruschi and M. D. Pinto. Lower bounds for the broadcast problem in mobile radio networks. *Distributed Computing*, 10:129–135, 1997.
- [BYGI92] R. Bar-Yehuda, O. Goldreich, and A. Itai. On the time complexity of broadcast in multi-hop radio networks. *Journal of Computer and Systems Science*, 45:104–125, August 1992.
- [CK87] I. Chlamtac and S. Kutten. Tree based broadcasting in multihop radio networks. *IEEE Transactions on Computers*, C-36(10), October 1987.
- [LBA93] C. Lee, J. E. Burns, and M. H. Ammar. Improved randomized broadcast protocols in multi-hop radio networks. In *Proceedings of International Conference on network Protocols*, pages 234–241, San Francisco, CA, 1993.
- [LM87] W. F. Lo and H. T. Mouftah. Collision detection and multitone tree search for multiple access protocols on radio channels. *IEEE Journal on Selected Areas Communication*, SAC-5:1035–1040, July 1987.
- [PR97] E. Pagani and G. P. Rossi. Reliable broadcast in mobile multihop packet networks. In *Proceedings of MOBICOM 97*, pages 34–42, Budapest, Hungary, 1997.