Enhanced Partial Dominant Pruning (EPDP) Based Broadcasting in Ad hoc Wireless Networks

Ashikur Rahman*, Md. Endadul Hoque, Farzana Rahman, Sabuj Kumar Kundu
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology
Dhaka-1000, Bangladesh
Email: ashikur@cs.ualberta.ca
Phone: 880-1558861114, Fax: 8802-9665612
Pawel Gburzynski
Department of Computing Science
University of Alberta, Edmonton, Canada

Abstract*— In many applications of ad-hoc wireless networks, one often has to broadcast the same message to all nodes. The major goal of any broadcasting algorithm is to minimize the number of retransmissions, i.e., to accomplish the goal with the minimum amount of traffic in the network. In addition to reducing the bandwidth expense needed to convey the message to all the nodes, this objective will try to minimize the total amount of energy spent by the nodes on this communal task. This is of paramount importance in sensor networks, which are often built of disposable nodes, whose life-time is directly determined by the efficiency of their power management scheme. In this paper, we present a heuristic broadcast algorithm dubbed EPDP, for Enhanced Partial Dominant Pruning, and demonstrate its superiority, in terms of the total number of retransmissions, over other solutions addressing the same issue.

General Terms: **Algorithm, broadcast, performance, protocols.**

keywords: **Ad-hoc Networks, Sensor Networks, Broadcasting, Power Management, Medium Access Control.**

I. INTRODUCTION

One of the fundamental modes of communication in a wireless ad-hoc network is broadcasting, where it is often used for route or service discovery, as well as various orchestrated communal actions, e.g., clock synchronization or implementing global duty cycles. Many unicast routing protocols such as DSR [1], AODV [2], ZRP [3], and LAR [4] use variants of broadcasting to establish and maintain routes.

In wireless sensor networks, which operate within a stringent resource budget (most notably power), an accurate implementation of global consensus is often critical from the viewpoint of energy efficiency, and thus survivability, of the entire network. While a globally synchronized action, initiated by a *master* node, may often help the network save power (e.g., by putting all nodes to sleep for the anticipated period of no activity), the cost of carrying out that action (i.e., its power budget) must be minimized in order to make it worthwhile. This cost is directly determined by the total number of packets that must be transmitted in order to convey the broadcast message to all the nodes. Consequently, minimizing that number should be a primary objective of any broadcast scheme.

Another relevant optimization criterion is the *broadcast latency* understood as the amount of time elapsing from the moment the broadcast packet departs form the originating node until the last node in the network receives its copy. As demonstrated in [5], these two objectives may be in conflict: there is a tradeoff between reducing the coverage overlap of the different copies of the broadcast packet and the amount of time needed to arrive at good trimming decisions. While certain applications of wireless ad-hoc networking may assign a high weight to the latency issue [5], it seems that in low-cost sensor networks (which tend to be slow and not bandwidth-constrained), the first objective is considerably more important.

The most straightforward (and still the most popular) approach to broadcasting is based on limited *flooding*, whereby each node rebroadcasts the received packet exactly once. This is usually quite costly and, especially in dense neighborhoods, may result in considerable redundancy, contention, and congestion referred to as the *broadcast storm* problem [6].

Several selective broadcasting techniques have been proposed to overcome the redundancy of flooding [7]–[10]. All these solutions utilize neighborhood information to reduce the number of redundant packets. We broadly classify all those attempts into two categories. By a *reactive* scheme we understand one in which a copy of

A preliminary version of this paper has appeared in the Proceedings of the International Symposium on Performance of Computer and Telecommunication Systems (SPECTS 2008), Edinburg, UK, June 2008.

*Correspondence to: Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka-1000, Bangladesh

doi:10.4304/jnw.4.9.895-904

the broadcast packet is always intended for all the nodes that can receive it. Then, it is up to the receiving node to decide whether the packet should be re-broadcast further. In contrast, with a *proactive* solution, the transmitting node indicates which of its neighbors are supposed to re-broadcast the packet. Then, having received such a packet, a node essentially knows it role. If it has been chosen as one of the forwarders, it is required to produce its own list of neighbors for which the re-broadcast copy of the packet will be intended.

With the reactive approach, a receiving node decides to re-broadcast the packet only if it concludes that its retransmission is going to cover new nodes, i.e., ones that have not been already covered by the received packet. For example, with *self pruning* [8], a re-broadcasting node includes the list of its neighbors in the packet header. A receiving node consults that list and retransmits the packet only if its own set of neighbors includes nodes that are not mentioned in the packet's list.

In a proactive scheme, a transmitting node selecting the forwarders from among its neighbors may use such criteria as node degree, power level, coverage area, etc. Examples of such solutions include *dominant pruning* [8], *partial dominant pruning*, and *total dominant pruning* [9]. Dominant pruning (DP) exploits 2-hop neighbor information. Each node maintains a subset of its one-hop neighbors (called *forward list*) whose retransmissions will cover all nodes located two hops away from the node. The forward list is passed in the header of every packet retransmitted by the node. Total dominant pruning (TDP) and Partial dominant pruning (PDP) attempt to reduce the redundancy of DP by creatively eliminating some nodes from the forward list.

In this paper, we propose an algorithm called Enhanced Partial Dominant Pruning (EPDP), which improves upon PDP by taking advantage of information that a node about to retransmit a broadcast packet can overhear from its neighbors. In a nutshell, the idea is to delay the retransmission for a moderate amount of time, as to give some of your neighbors a chance to go first. With the proper selection of the delays, a noticeable reduction in redundancy can be achieved. This way, at the cost of some increase in the latency of the broadcast operation, we can further reduce the total number of retransmissions needed for its completion. This kind of trade-off may be appropriate for a network whose power budget is the primary concern.

II. Related Work

The redundancy of straightforward flooding was studied in [6], where the broadcast storm problem was identified. As a way out, the authors suggested a probabilistic approach driven by several types of heuristics, including counter-based, distance-based, location-based, and cluster-based schemes. All those solutions mainly differ on two issues: how a node can assess the redundancy of a retransmission, and how the nodes can collectively utilize such assessments. The main problem of all those algorithms is that they only yield a probabilistic coverage of any broadcast operation.

Lim and Kim [10] proved that building a minimum flooding tree is equivalent to finding a *minimum connected dominating set* (MCDS), which is an NP-complete problem. In [8], they also suggested a few efficient forwarding schemes for broadcasting as well as multicasting, notably *self pruning* and *dominant pruning*. With self pruning, each node only has to be aware of its one-hop neighbors, which is accomplished via periodic HELLO messages. For dominant pruning, based on 2-hop neighbor information. the HELLO messages are sent with the TTL (time to live) of 2, which means that each of them is re-broadcast once.

Peng et al. [7] presented a modification of the self-pruning algorithm named SBA (for *scalable broadcast algorithm*). The scheme imposes randomized delays before retransmissions. Similar to [8], a node does not rebroadcast its copy of the packet, if the copies received during the waiting time appear to have covered all its neighbors. The performance of this scheme turs out to be very sensitive to the length of the waiting period.

In [5], two of us introduced a generic broadcast algorithm based on delaying the retransmission in order to collect more information about the neighborhood. The proper selection of *defer time* plays a significant role in the performance of the proposed schemes. Except for the most naive probabilistic criterion, it is natural to expect that the longer the defer time, the more information is likely to reach the node before it is forced to make the decision. This demonstrates the trade-off between the latency and redundancy of the broadcast operation.

Total dominant pruning (TDP) and partial dominant pruning (PDP) proposed in [9] appear to make the most efficient use of neighborhood information. With TDP, 2-hop neighborhood information from the immediate

sender is included in the header of every broadcast packet. A node receiving such a packet builds its forward list based on that information. The main drawback of TDP is that it requires high bandwidth (and long packets), at least in those scenarios where the neighborhoods tend to be large. This problem is avoided in PDP.

III. The Algorithms

An ad-hoc wireless network is represented as a graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. Each edge (u, v) expresses the fact that u and v are neighbors, which relation is assumed to be symmetric. The requisite 2-hop neighborhood information is obtained by periodically broadcasting HELLO packets with the TTL of 2 hops.

A. Dominant Pruning

Let $N(u)$ be the set of all one-hop neighbors of node u. By $N^2(u)$ we shall denote the set of all one-hop and two-hop neighbors of u, i.e.,

$$N^2(u) = \{v | v \in N(u) \vee \exists z[v \in N(z) \wedge z \in N(u)]\}$$

The dominant pruning algorithm [8] is a deterministic broadcast scheme with complete coverage. The latter means that a broadcast packet reaches all nodes in the network under the assumption that a single-hop broadcast is always reliable and reaches all neighbors of the sender. Viewed as flooding containment scheme, dominant pruning limits the population of forwarding nodes to the so-called *connected dominating set*, which is any set of nodes $\mathcal{S}$ satisfying this property:

$$\forall u \, [u \in \mathcal{S} \vee \exists v \in \mathcal{S} \, [u \in N(v)]]$$

By definition, every node in the network is either in $\mathcal{S}$ or is directly reachable by a node in $\mathcal{S}$; thus, when every node in $\mathcal{S}$ rebroadcasts a packet, it will reach 100% of nodes. Therefore, one solution to the optimal broadcast problem is to find a connected dominating set of the minimum size. Unfortunately, this problem is NP-hard [8].

Each node u determines its forward list as a subset of its one-hop neighbors whose transmissions will cover all two-hop neighbors of u.

Then, when transmitting a broadcast packet, u explicitly indicates that it should be rebroadcast only by the nodes on the forward list. As it turns out, finding a minimum-size forward list is still NP-hard [8]; thus, Lim and Kim have suggested the following greedy approach:

Suppose that v has just received a packet from node u. The packet's header includes the forward list F_u inserted there by u. The case $v \notin F_u$ is simple: the node will not rebroadcast the packet; otherwise, v has to create its own forward list F_v to be inserted into the header of the rebroadcast copy.

The node starts by constructing U_v, which is the set of uncovered two-hop neighbors of v. This set includes all two-hop neighbors of v that have not been covered by the received packet, i.e., $U_v = N^2(v) - N(v) - N(u)$. Note that every node knows the population of its two-hop neighbors; thus, $N(u)$ is known to v. Then, v sets $F_v = \emptyset$ and $B(u,v) = N(v) - N(u)$. As illustrated in Figure 1, the set $B(u,v)$ represents those neighbors of v that are possible candidates for inclusion in F_v. Then, in each iteration, v selects a neighbor $w \in B(u,v)$, such that w is not in F_v and the list of neighbors of w covers the maximum number of nodes in U_v, i.e., $|N(w) \cap U_v|$ is maximized. Next v includes w in F_v and sets $U_v = U_v - N(w)$. The iterations stop when either U_v becomes empty or no more progress can be accomplished (i.e., F_v can grow no further).

B. Partial Dominant Pruning

Attempts to improve upon DP focus on reducing the size of the forward list created by each of the retransmitting nodes. PDP, as proposed in [9], eliminates from the initial content of U_v (as defined by DP), the set of nodes reachable from the intersection of the neighborhoods of u and v.

Let $P(u,v) = N(N(u) \cap N(v))$. The starting set of the uncovered 2-hop neighbors is defined as $U_v = N^2(v) - N(u) - N(v) - P$. Then, the algorithm proceeds exactly as DP.

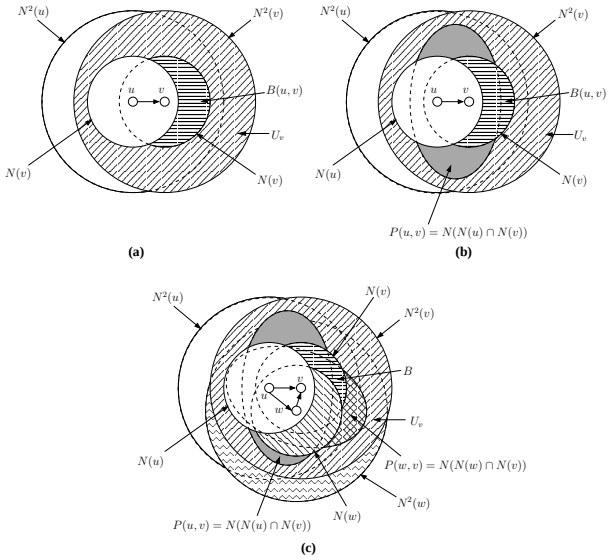


Figure 1. Neighbor sets in three algorithms: (a) Dominant Pruning, (b) Partial Dominant Pruning, (c) Enhanced Partial Dominant Pruning.

C. Enhanced Partial Dominant Pruning

The key idea behind our improvement is to take advantage of the fact that a node about to retransmit a broadcast packet may be able to overhear some traffic related to the current forwarding step. Consider a node v waiting to re-transmit a packet received from its neighbor u. Quite likely, the same packet has been received by other nodes, some of them neighbors of v, which find themselves in the same situation. The underlying MAC (medium access control) scheme will attempt to schedule all those potentially interfering transmission in some manner that attempts to resolve the contention and avoid collision. Thus, while waiting for its turn to transmit, v may overhear a retransmission of one of its neighbors. The question is this: can the information extracted from the overheard packet help v correct its decision? The gist of our idea is to introduce intentional delays into the retransmissions aimed at increasing the likelihood of overhearing useful information.

Thus, v will decide to postpone its retransmission for the prescribed value of *defer time*, which is a tunable parameter of the algorithm. Suppose that, while waiting for its turn to transmit, v has received copies of the same broadcast packet from nodes w_1, w_2, ..., w_n. Then, it can conclude that all nodes in $N(w_1) \cup N(w_2) \cup \ldots \cup N(w_n)$ have already received their copies. Therefore, it can redefine the uncovered set as

$$U_v = N^2(v) - N(u) - P - \sum_{i=1}^{n} N(N(w_i) \cap N(v))$$

where, $P = N(N(u) \cap N(v))$.

Formally, the algorithm operates as follows:
1. Node v receives a broadcast packet m from node u. If m has not been seen before, and v is in the forward list, v uses $N^2(v)$, $N(u)$, and $N(v)$ to construct:

$$P = N(N(u) \cap N(v))$$
$$U_v = N^2(v) - N(u) - N(v) - P$$
$$B = N(v) - N(u).$$

2. Node v sets a timer at *current time + defer time* (the selection of *defer time* is explained below). When the timer goes off, the node finds itself at step 4.
3. The node is waiting for the timer to go off. Whenever v receives a copy of m from some node w, it updates U and B:

$$P = N(N(v) \cap N(w))$$
$$U_v = U_v - N(w) - P$$
$$B = B - N(w)$$

and then discards m.
4. The timer has expired. If $B = \emptyset$ (note that this implies $U_v = \emptyset$, as $U_v \subseteq N(B)$, see [9]), v cancels its retransmission of m. Otherwise, v follows the same procedure for creating F_v as DP and sends the packet.

The complete algorithm is described in Algorithm 1 and Algorithm 2.

Algorithm 1 CREATEFORWARDLIST(U_v, B_v)

/*
Input: Set U_v and set B_v of node v
Output: The forward list F_v of node v
*/

1: $F_v \leftarrow \emptyset$
2: **repeat**
3: Select $w \in B_v$ such that $w \notin F_v$ and $|N(w) \cap U_v|$ is maximized
4: $F_v \leftarrow F_v \cup \{w\}$
5: $U_v \leftarrow U_v - N(w)$
6: **until** $(U_v = \emptyset) \vee (F_v$ *grows no further)*
7: **return** F_v

The proper selection of *defer time* is critical for the success of the above scheme. Note that the same value used by all nodes is equivalent to zero, as then the actual ordering of retransmissions will be determined (probably at random) by the MAC layer. While it is possible that some savings will be obtained that way, one can think of a more intelligent selection for defer time.

For example, it seems natural to let a node with more neighbors transmit earlier, as the large number of covered nodes will more likely render other scheduled retransmissions redundant. We propose a deterministic selection of *defer time* in proportion to the node's coverage, as implied by its perception in DP.

Suppose node u broadcasts a packet at time t.

Let the forward list of the packet consist of three nodes, i.e., $F_u = [v, w, y]$ where $v, w, y \in N(u)$. Each of the three nodes v, w, y receives its first copy of the packet at approximately the same time $t + \delta$, where δ represents the processing latency and propagation time, which may slightly differ for different nodes. According to the procedure of constructing F_u, the positions of the nodes on the list reflect the decreasing order of their coverage, i.e., v was selected as the node covering the largest subset of U_u. Thus we postulate that the amount of waiting time be a straightforward linear function of the node's position on the forward list extracted from the received copy of the packet. Specifically, we introduce a unit of waiting time d and make *defer time* equal to $i \times d$, where i is the node's index on the forward list F_u. Thus, the first node from the list will wait for interval d, the second for $2d$, and so on. The setting of d should be the minimum allowing for a clear separation of the transmissions scheduled at different intervals, after allowing for the maximum difference in processing latencies and MAC-layer delays.

While self pruning is a reactive algorithm and both DP and PDP are proactive ones, our scheme exemplifies a combination of both paradigms. Its proactive component consists in the fact that every transmitting node creates a forward list to narrow down the population of its neighbors that should retransmit the packet. However, despite its presence in the forward list, a neighbor may conclude that its transmission would be redundant and opt out from its assigned duties.

Algorithm 2 ENHANCED PARTIAL DOMINANT PRUNING

/*
Precondition: Node v receives an unseen packet m from node u and $v \in F_u$
Input: Set $N(u)$, $N(v)$ and $N^2(v)$
Output: The forward list F_v of node v
*/

1: **S1**: $P = N(N(u) \cap N(v))$
2: $U_v = N^2(v) - N(u) - N(v) - P$
3: $B_v = N(v) - N(u)$
4: *wait time* $\leftarrow$ *current time* $+$ *defer time*
5: In S2, if the same message m is heard again, interrupt the waiting and perform S4.
6: **S2**: Wait for *wait time*.
7: **S3**: $F_v \leftarrow CreateForwardList(U_v, B_v)$
8: Rebroadcast message m appending F_v.
9: The procedure exits.
10: **S4**: $P = N(N(v) \cap N(w))$
11: $U_v = U_v - N(w) - P$
12: $B_v = B_v - N(w)$
13: Discard the copy of message m.
14: **if** $B_v = \emptyset$ **then**
15: Cancel the transmission of message m.
16: The host is prohibited from rebraodacsting.
17: The procedure exits.
18: **else**
19: Resume waiting in S2.
20: **end if**

IV. EXAMPLE

We shall now demonstrate the operation of DP, PDP and EPDP in the sample network shown in Fig. 2, whose neighborhoods are listed in Table I. Even though 2-hop neighborhoods could be derived from the one-hop ones, both types are included in the table to facilitate our explanation. We consider a broadcast operation initiated by node 11, i.e., this node sends the first copy of the broadcast packet, which is expected to reach all the remaining nodes.

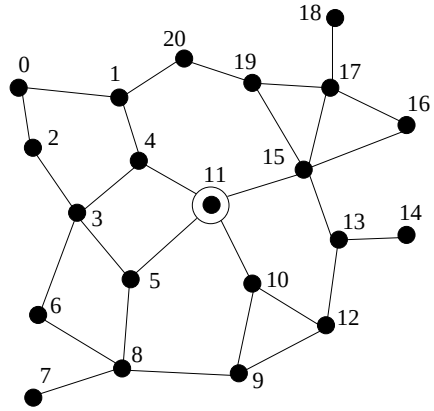


Figure 2. A sample network of 21 nodes with source node 11.

TABLE I.
TWO-HOP NEIGHBORS

v	$N(v)$	$N^2(v)$
0	0,1,2	0,1,2,3,4,20
1	0,1,4,20	0,1,2,3,4,11,19,20
2	0,2,3	0,1,2,3,4,5,6
3	2,3,4,5,6	0,1,2,3,4,5,6,8,11
4	1,3,4,11	0,1,2,3,4,5,6,10,11,15,20
5	3,5,8,11	2,3,4,5,6,7,8,9,10,11,15
6	3,6,8	2,3,4,5,6,7,8,9
7	7,8	5,6,7,8,9
8	5,6,7,8,9	3,5,6,7,8,9,10,11,12
9	8,9,10,12	5,6,7,8,9,10,11,12,13
10	9,10,11,12	4,5,8,9,10,11,12,13,15
11	4,5,10,11,15	1,3,4,5,8,9,10,11,12,13,15,16,17,19
12	9,10,12,13	8,9,10,11,12,13,14,15
13	12,13,14,15	9,10,11,12,13,14,15,16,17,19
14	13,14	12,13,14,15
15	11,13,15,16,17,19	4,5,10,11,12,13,14,15,16,17, 18,19,20
16	15,16,17	11,13,15,16,17,18,19
17	15,16,17,18,19	11,13,15,16,17,18,19,20
18	17,18	15,16,17,18,19
19	15,17,19,20	1,11,13,15,16,17,18,1,20
20	1,19,20	0,1,4,15,17,19,20

Let us begin with DP. When node 11 broadcasts the packet, it constructs:

$$U_{11} = N^2(11) - N(11) = \{1, 3, 8, 9, 12, 13, 16, 17, 19\}$$

and

$$B(\phi, 11) = N(11) = \{4, 5, 10, 15\},$$

Note that the packet *departs* from node 11, as opposed to having been received from one of its neighbors; thus, the construction of B is somewhat degenerate. Consequently, the forward set of the packet becomes:

$$F_{11} = [15, 4, 10, 5]$$

with the four nodes having been added in the listed order. When node 15 re-broadcasts the packet, it determines:

$$U_{15} = N^2(15) - N(11) - N(15) = \{12, 14, 18, 20\}$$

and forms the forward list as

$$F_{15} = [13, 17, 19].$$

Similarly, for node 4, we have:

$$U_4 = N^2(4) - N(11) - N(4) = \{0, 2, 6, 20\}$$

and

$$F_4 = [1, 3].$$

The operation continues in this fashion until all the nodes have received a copy of the packet. The requisite sets constructed by all nodes in its course are listed in Table II. The reader can easily verify that it takes the total of 16 transmissions (including the original broadcast at node 11) to convey the packet to all nodes.

PDP acts in a similar manner, except that some of the sets end up a bit smaller. Again, node 11 starts the broadcast operation with its initial transmission. Clearly, its forward list is going to be exactly the same as in the previous case. Also, when node 15 gets to retransmitting its copy of the packet, it constructs:

$$U_{15} = N^2(15) - N(11) - N(15) - P(11, 15) = \{12, 14, 18, 20\}$$

and its forward list becomes:

$$F_{15} = [13, 17, 19],$$

i.e., exactly as before. Similarly, for node 4 we get

$$U_4 = N^2(4) - N(11) - N(4) - P(11, 4) = \{0, 2, 6, 20\}$$

and the forward list becomes

$$F_4 = [1, 3].$$

Then nodes 10, 5, and 13 rebroadcast their copies. Note that when node 17 is about to rebroadcast the packet, it gets:

$$U_{17} = N^2(17) - N(15) - N(17) - P(15, 17) = \phi,$$

and its forward list F_{17} becomes empty. This means that the packet should only be received (by all neighbors that haven't yet seen a copy), but never re-broadcast.

The tally of the number of transmissions for PDP (Table III) yields exactly the same performance as for DP. Even though the sets U_{17}, U_{19}, U_9 and U_{12}, are smaller than in the previous case, the reductions do not affect the forward lists.

Now, let us trace the behavior of EPDP on the same network. Again, node 11 starts with the forward list:

$$F_{11} = [15, 4, 10, 5].$$

Similarly, having received the packet from node 11, node 15 constructs:

$$\begin{aligned} U_{15} = & \quad N^2(15) - N(11) - N(15) - P(11,15) \\ = & \quad \{12, 14, 18, 20\} \end{aligned}$$

and

$$F_{15} = [13, 17, 19],$$

while node 4 arrives at

$$U_4 = N^2(4) - N(11) - N(4) - P(11,4) = \{0, 2, 6, 20\}$$

and

$$F_4 = [1, 3].$$

Here we arrive at the first difference between EPDP and PDP/DP. According to EPDP, node 10 delays is retransmission for $3d$ time units. What happens in the meantime, is the retransmission of nodes 15 and then 13 (note that each of them incurs the delay of $1d$ with respect to the time when node 10 received the packet. Before node 13 transmits its copy, it obtains:

$$U_{13} = N^2(13) - N(15) - N(13) - P(15,13) = \{9, 10\}$$

and

$$F_{13} = [12].$$

When node 12 receives the packet sent by node 13, it determines:

$$\begin{aligned} U_{12} = & \quad N^2(12) - N(12) - N(13) - P(12,13) \\ = & \quad \{8,9,10,11,12,13,14,15\} - \{9,10,12,13\} \\ & -\{12,13,14,15\} - \phi \\ = & \quad \{8, 11\} \end{aligned}$$

and

$$B(13,12) = \{9, 10\}.$$

When node 10 finally retransmits its copy of the packet, node 12 will receive it (note that the two nodes are neighbors) and accordingly update its sets:

$$\begin{aligned} U_{12} &= U_{12} - N(10) - N(N(12) \cap N(10)) \\ &= \{8, 11\} - \{9, 10, 11, 12\} - \{8\} = \phi \end{aligned}$$

TABLE II.
THE DP ALGORITHM

u	v	U_v	$B(u,v)$	F_v
ϕ	11	1,3,8,9,12,13,16,17,19	4,5,10,15	15,4,10,5
11	15	12,14,18,20	13,16,17,19	13,17,19
11	4	0,2,6,20	1,3	1,3
11	10	8,13	9,12	9,12
11	5	2,6,7,9	3,8	8,3
15	13	9,10	12,14	12
15	17	20	18	[]
15	19	1,18	20	20
4	1	2,19	0,20	0,20
4	3	0,8	2,5,6	2,5
10	9	5,6,7,13	8	8
10	12	8, 14, 15	13	13
5	8	10,12	6,7,9	9
19	20	0,4	1	1
1	0	3	2	2
3	2	1	0	0

$$B = B - N(10) = \{9, 10\} - \{9, 10, 12\} = \phi.$$

Note that node 15, which (according to PDP/DP) could have re-broadcast by now, has been delayed for $4d$ time units (it was the 4'th node in F_{11}). Nodes 17, 1, and 12 will all go ahead of node 15 (by $1d$). Node 17 computes:

$$U_{17} = N^2(17) - N(15) - N(17) - P(15,17) = \phi$$

and

$$F_{17} = \phi.$$

Despite tha fact that $F_{17} = \phi$, the node still has to re-broadcast as $B_{17} \neq \phi$. Node 12 calculates:

$$U_{12} = \phi, B = \phi$$

and

$$F_{12} = \phi.$$

Thus, node 12 cancels its retransmission. From Table IV, we can see that nodes 20 and 2 encounter the same situation and also cancel their retransmissions. This way, the total number of transmissions for EPDP is 13, which constitutes a visible improvement over DP and PDP.

TABLE III.
THE PDP ALGORITHM

u	v	U_v	$B(u,v)$	F_v
ϕ	11	1,3,8,9,12,13,16,17,19	4,5,10,15	15,4,10,5
11	15	12,14,18,20	13,16,17,19	13,17,19
11	4	0,2,6,20	1,3	1,3
11	10	8,13	9,12	9,12
11	5	2,6,7,9	3,8	8,3
15	13	9,10	12,14	12
15	17	ϕ	18	[]
15	19	1	20	20
4	1	2,19	0,20	0,20
4	3	0,8	2,5,6	2,5
10	9	5,6,7	8	8
10	12	14,15	13	13
5	8	10,12	6,7,9	9
19	20	0,4	1	1
1	0	3	2	2
3	2	1	0	0

TABLE IV.
(THE EPDP ALGORITHM)

u	v	U_v	$B(u,v)$	F_v	Rebroadcast
ϕ	11	1,3,8,9,12,13,16,17,19	4,5,10,15	15,4,10,5	Broadcast
11	15	12, 14, 18, 20	13, 16, 17,19	13, 17,19	Yes
11	4	0,2,6,20	1,3	1,3	Yes
15	13	9,10	12,14	12	Yes
11	10	8,13	9,12	9,12	Yes
15	17	ϕ	18	[]	Yes
4	1	2,19	0,20	0,20	Yes
13	12	ϕ	ϕ	[]	No
11	5	2,6,7,9	3,8	8,3	Yes
4	19	1	20	20	Yes
15	3	0	2,6	2	Yes
10	9	5,6,7	8	8	Yes
1	0	3	2	2	Yes
1	20	ϕ	ϕ	[]	No
5	8	ϕ	6,7	[]	Yes
3	2	ϕ	ϕ	[]	No

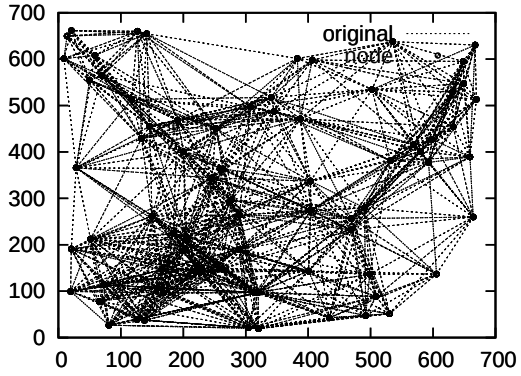


Figure 3. A sample Scenario under uniform node distribution of 75 nodes

V. Simulation Results

While the example from the previous section shows that there exist situations when EPDP brings about an improvement over DP and PDP, one would like to see how much of that improvement will materialize statistically in "typical" networks. To this end, we have carried out simulation experiments in random networks collecting the following performance measures:

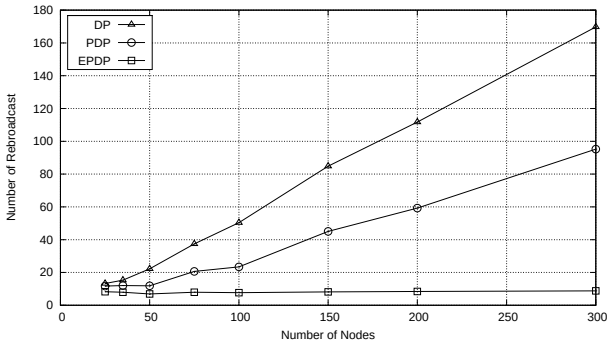
a) The normalized average number of transmissions, i.e., the number of transmissions required to propagate a broadcast packet within the network averaged over the number of cases and divided by the total number of nodes.

b) The average latency, i.e., the amount of time elapsing from the moment a broadcast packet is generated at the source, until its copy is received by the last node, averaged over the number of cases.

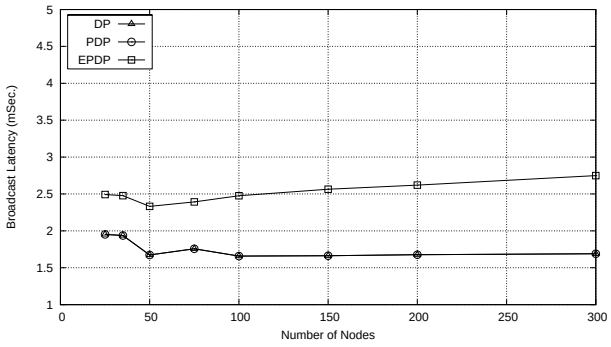
We have also monitored the *reachability*, i.e., the ratio of the number of nodes receiving a copy of the broadcast packet divided by total number of nodes in the network. Of course, unless the network is disconnected, the reachability should be 1.0, so the primary role of this measure was to assert the correctness of our scheme.

We deployed random networks with 25-300 nodes distributed over a flat square area (typically 670m$\times$670m). The transmission range, 250m, was the same for all nodes. An ideal MAC layer was assumed, with no contention and no losses. The transmission rate was 2 Mbps (2,000,000 bps) and the size of each broadcast packet was the same and equal to 64 bytes (64$\times$8 bits), excluding the dynamic length of headers needed to accommodate the neighbor lists. The unit of defer time d was set to 0.5 msec, which covers the transmission time as well as the (negligible in this context) propagation delay.

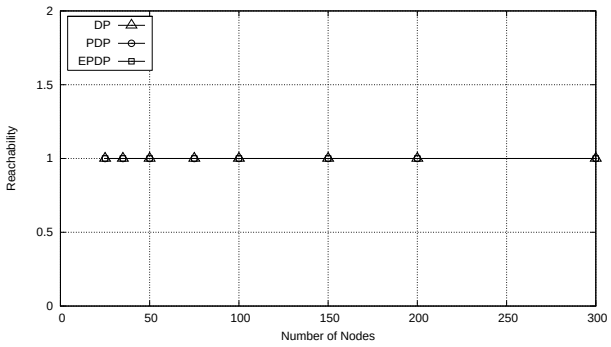
Each node generated a broadcast packet in turn, but only one broadcast packet (possibly in multiple copies) was present in the network at any given time. The performance measures were collected for each broadcast packet and then, at the end of run, the averages of all those measures were taken. The actual points used to draw the curves were the averages of those averages collected over five independent experiments.



(a)



(b)



(c)

Figure 4. Performance of EPDP, PDP, and DP versus the number of nodes (Uniform distribution): (a) number of re-broadcasts, (b) latency, (c) reachability.

Two types of node distributions were considered: uniform and Zipf. For topologies under uniform distribution, we use random node placement, which yields a fairly uniform topology. A typical example of such topology is shown in Figure 3. It shows a typical uniform deployment of 75 nodes over a rectangular area of 670m$\times$670m, each node having a maximum communication range of 250m.

Fig. 4, (a), (b), and (c) show the performance results for uniform node distribution. Fig. 4(a) shows the average number of re-broadcasts for different number of nodes

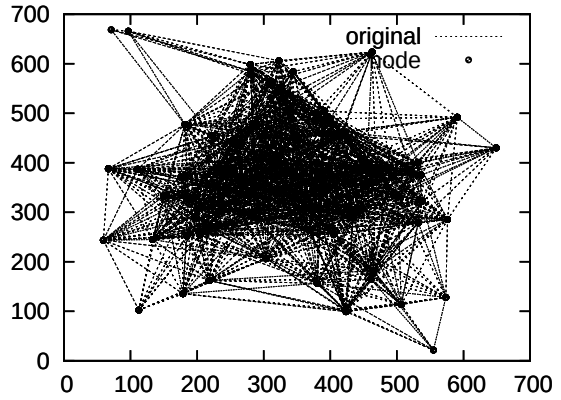


Figure 5. A sample Scenario under uniform node distribution of 75 nodes

(from 25 to 300). We can conclude that EPDP performs much better than DP and PDP, especially (and not surprisingly) for networks consisting of a non-trivial number of nodes.

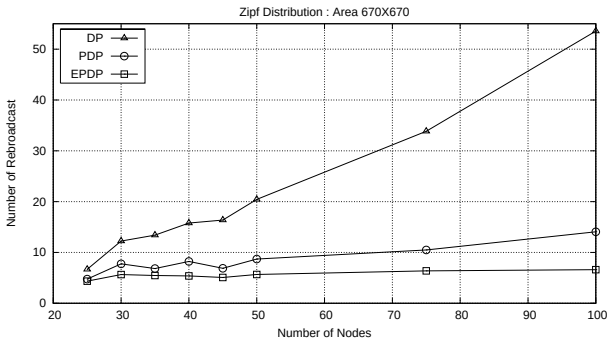
Fig. $4(b)$ shows the broadcast latency, which exhibits an increase in comparison to DP and PDP. However, its magnitude (which directly depends on the unit of defer time d) appears to be within an acceptable range.

Finally, Fig. $4(c)$ depicts the reachability, which demonstrates the correctness of EPDP: its reachability (1.0) is the same as for the other two schemes.

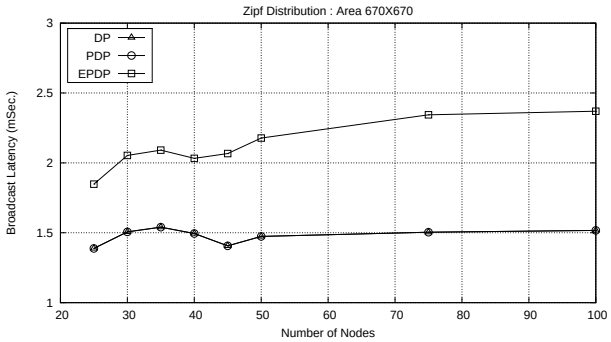
The results shown in Figures $6(a)$, $6(b)$, and $6(c)$ correspond to the Zipf distribution of nodes. In that case, 80% of nodes occupy 20% of the total area. Figure 5 shows a typical topology involving 75 nodes under this biased zipf distribution of nodes. The maximum communication range of each node was $250m$. Deployment area was 670m$\times$670m.

While EPDP exhibits consistent superiority over DP and PDP, it comes at the price of increased latency. This can be explained by the less balanced neighborhoods resulting in occasionally longer forward lists, which in turn increase the range of indexes for defer intervals. Also, occurrences of sparse areas in the network translate into relatively more retransmissions (compared to the uniform case), which tend to feed into the overall latency. The incidental departures from perfect reachability (Fig˙ $6(c)$) for small networks result from their disconnected nature. Note that the likelihood of a network being disconnected is considerably higher under Zipf distribution than in the previous (uniform) case. Notably, all three protocols exhibit identical reachability under such conditions.

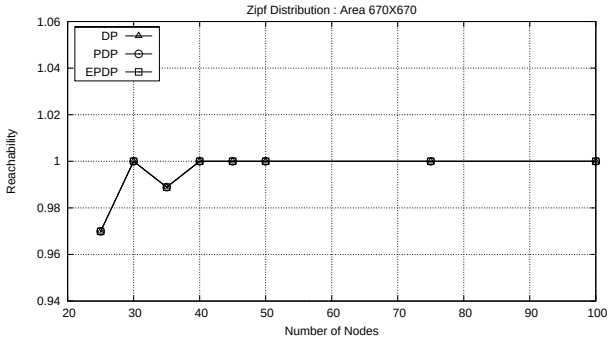
The last two figures illustrate the impact of node density on the behavior of EPDP. In those experiments, we varied the network area (the x-axis corresponds to the side of the square), while the number of nodes remained fixed (between 25 and 100). Fig. $7(a)$ shows that the number of re-broadcasts tends to increase with the decreasing density of nodes. This is expected, as increased density implies larger neighborhoods and, consequently, better coverage of a single transmission. The latency tends to increase



(a)



(b)



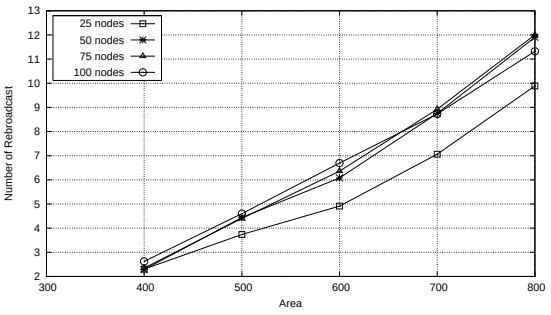
(c)

Figure 6. Performance of EPDP, PDP, and DP versus the number of nodes (Zipf distribution): (a) number of re-broadcasts, (b) latency, (c) reachability.

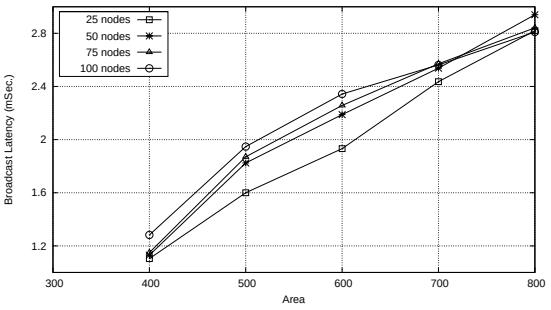
along with the number of re-broadcasts, which is less obvious. Note that larger neighborhoods tend to result in longer forward lists and thus

larger defer delays. However, as it turns out, the positive impact of the reduced total number of re-broadcasts is more pronounced than the negative impact of larger neighborhoods.

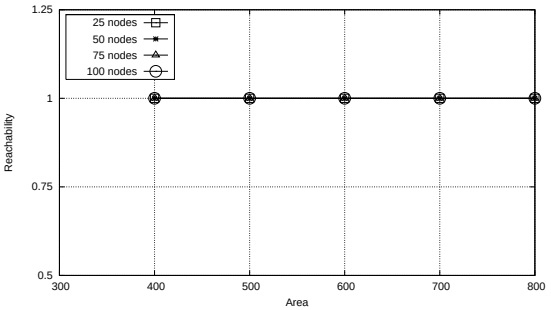
Next we compare the energy consumption of different broadcast algorithms. The energy consumption is calculated as follows. Suppose length of a data packet is L



(a)



(b)



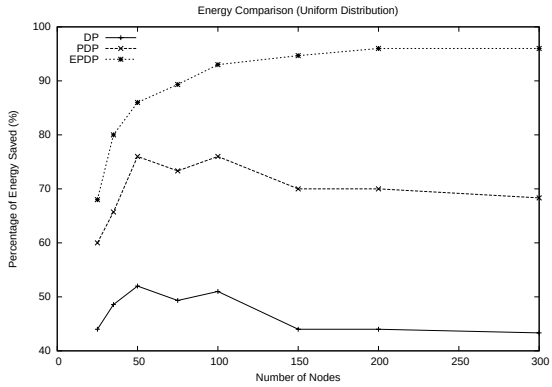
(c)

Figure 7. Performance of EPDP for different network density: (a) number of re-broadcasts, (b) latency, (c) reachability

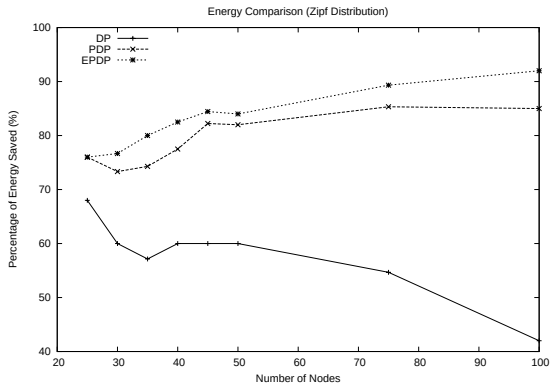
bits and the data transmission rate of a node is C bps. Therefore the transmission time of a packet is $\frac{L}{C}$ seconds. If the transmission power required to transmit a packet is P watts then the total energy consumed to transmit the packet is $P \times \frac{L}{C}$ Joule. To compare energy consumptions of DP, PDP and EPDP, we took energy consumption of blind flooding as a reference level. In blind flooding, each node rebroadcasts a packet exactly once to complete a broadcast. Suppose the total energy consumed in blind flooding is E_f, and the total energy consumed by a broadcast algorithm (either DP, PDP or EPDP) is E_a, then percentage of energy saved by the algorithm can be defined using the following equation:

$$\text{Percentage of energy saved} = \frac{E_f - E_a}{E_f} \times 100(\%)$$

Figure 8(a) and (b) shows percentage of energy saved by using DP, PDP and EPDP algorithms under uniform and zipf distribution respectively. The value of different parameter‘s for calculating energy spent is listed in Table V. The energy savings of EPDP is higher than PDP and much higher than DP under both uniform and zipf distributions. Also EPDP has more energy savings compared to DP and PDP in dense network than sparse network. The reason is, the number of rebroadcast (shown in Figure 8 (a) and (b)) in EPDP is much lower than DP and PDP in dense networks. The similar argument is applicable for zipf distribution.



(a)



(b)

Figure 8. Energy Consumption of DP, PDP and EPDP for different number of nodes: (a) under uniform distribution, (b) under zipf distribution

VI. CONCLUSIONS

We have proposed a broadcast algorithm for ad-hoc wireless networks, which provides full coverage (as long as the network remains connected), while greatelly reducing the number of redundant retransmissions. Our

TABLE V.
ENERGY CALCULATION PARAMETERS AT A GLANCE

Topic	Description
Packet length (L)	64 bits
Transmission rate (C)	2 Mbps
Transmission power (P)	28 mille-Watts
Node Distribution	Uniform and Zipf

simulation studies have demonstrated the superiority of EPDP over previous algorithms for practically all kinds of networks, including those with unbalanced distribution of nodes.

Although, EPDP exhibits a slight increase in broadcast latency over DP and PDP, this increase is insignificant and typically completely offset by the energy savings resulting from the fewer number of necessary re-broadcasts.

In future work, we plan to analyze the possible applicability and impact of our deferral scheme to other broadcast algorithm, such as TDP [9] and AMCDS [11], which have shown better performance than DP and PDP [9].

REFERENCES

[1] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*, 1996, vol. 353.

[2] C. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector (AODV) routing," in *Proceedings Second IEEE Workshop Mobile Computing Systems and Application (WMCSA)*, 1999, pp. 90–100.

[3] Z. Haas and M. Pearlman, "The performance of query control schemes for the zone routing protocol," in *SIGCOMM'98*, 1998.

[4] Y. Ko and N. H. Vaidya, "Location-aided routing (LAR) in mobile ad hoc networks," in *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, 1998, pp. 66–75.

[5] A. Rahman and P. Gburzynski, "MAC-assisted broadcast speedup in ad-hoc wireless networks," in *Proceedings International Wireless Communications and Mobile Computing (IWCMC)*, 2006, pp. 923–928.

[6] S. Ni, Y. Tseng, Y. Chen, and J. Sheu, "The broadcast storm problem in a mobile ad hoc network," in *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, 1999, pp. 151–162.

[7] W. Peng and X. Lu, "On the reduction of broadcast redundancy in mobile ad hoc networks," in *Proceedings First Annual Workshop Mobile and Ad Hoc Networking and Computing (MOBIHOC)*, 2000, pp. 129–130.

[8] H. Lim and C. Kim, "Multicast tree construction and fbooding in wireless ad hoc networks," in *Proceedings of the 3rd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM)*, 2000, pp. 61–68.

[9] W. Lou and J. Wu, "On reducing broadcast redundancy in ad-hoc wireless networks," *IEEE Transaction on Mobile Computing*, vol. 1, no. 2, pp. 111–123, 2002.

[10] H. Lim and C. Kim, "Flooding in wireless ad-hoc networks," *Computer Communications J.*, vol. 24, no. 3-4, pp. 353–363, 2001.

[11] B. Das, R. Sivakumar, and V. Bharghavan, "Routing in ad-hoc networks using a virtual backbone," in *Proceedings International Conference on Computer Communications and Networks (ICCCN)*, 1997, pp. 1–20.

Ashikur Rahman received his BSc and MSc degrees in Computer Science and Engineering from the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh in 1998 and 2001, respectively. He received his PhD in 2006 from the Department of Computing Science, University of Alberta, Canada. After finishing his PhD, he worked as a postdoctoral researcher at Simon Fraser University, Canada. His research interests include ad-hoc and sensor networks, peer-to-peer computing, swarm intelligence, back-end compiler optimization and neural networks. He is currently working as an Assistant Professor in the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET).

Endadul Hoque received his BSc degree in Computer Science and Engineering from the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh in 2008. Now he is a graduate student at the Department of Mathematics, statistics and Computer Science, Marquette University, USA. His research interest includes ad-hoc and sensor networks, distributed computing, Ubiquitous computing etc.

Farzana Rahman has received her BSc Engineering degree in Computer Science and Engineering from the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh in January, 2008. She is currently a graduate student at the Department of Mathematics, statistics and Computer Science, Marquette University, USA. Her research interest includes ad-hoc and sensor networks, Ubiquitous computing etc.

Sabuj Kumar Kundu received his BSc degree in Computer Science and Engineering from the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh in 2008. His research interest includes ad-hoc and sensor networks, distributed computing etc.

Pawel Gburzynski received his MSc and PhD in Computer Science from the University of Warsaw, Poland in 1976 and 1982, respectively. Before coming to Canada in 1984, he had been a research associate, systems programmer, and consultant in the Department of Mathematics, Informatics, and Mechanics at the University of Warsaw. Since 1985, he has been with the Department of Computing Science, University of Alberta, where he is a Professor. Dr. Gburzynskis research interests are in communication networks, operating systems, simulation, and performance evaluation.