Double-Covered Broadcast (DCB): A Simple Reliable Broadcast Algorithm in MANETs

Wei Lou and Jie Wu
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431
Email:{wlou, jie}@cse.fau.edu

***Abstract*— Mobile ad hoc networks (MANETs) suffer from high transmission error rate because of the nature of radio communications. The broadcast operation, as a fundamental service in MANETs, is prone to the broadcast storm problem if forward nodes are not carefully designated. The objective of reducing the broadcast redundancy while still providing high delivery ratio for each broadcast packet is a major challenge in a dynamic environment. In this paper, we propose a simple, reliable broadcast algorithm, called double-covered broadcast (DCB), that takes advantage of broadcast redundancy to improve the delivery ratio in the environment that has rather high transmission error rate. Among 1-hop neighbors of the sender, only selected forward nodes retransmit the broadcast message. Forward nodes are selected in such a way that (1) the sender's 2-hop neighbors are covered and (2) the sender's 1-hop neighbors are either a forward node, or a non-forward node but covered by at least two forwarding neighbors. The retransmissions of the forward nodes are received by the sender as confirmation of their receiving the packet. The non-forward 1-hop neighbors of the sender do not acknowledge the reception of the broadcast. If the sender does not detect all its forward nodes' retransmissions, it will resend the packet until the maximum times of retry is reached. Simulation results show that the algorithm provides good performance for a broadcast operation under high transmission error rate environment.**

***Index Terms*— Broadcast, forward node, MANETs, performance evaluation, reliability.**

I. Introduction

A mobile ad hoc network (MANET) enables wireless communications between participating mobile nodes without the assistance of any base station. Two nodes that are out of one another's transmission range need the support of intermediate nodes which relay messages to set up a communication between each other. The broadcast operation is the most fundamental role in MANETs because of the broadcasting nature of radio transmission: When a sender transmits a packet, all nodes within the sender's transmission range will be affected by this transmission. The advantage is that one packet can be received by all neighbors; the disadvantage is that it interferes with the sending and receiving of other transmissions, creating *exposed terminal problem*, that is, an outgoing transmission collides with an incoming transmission, and *hidden terminal problem*, that is, two incoming transmissions collide with each other.

Blind flooding, where each node forwards the packet once and only once, makes every node a forward node. If the forward nodes are not carefully designated, they will trigger many retransmissions at the same time which congest the network. This is referred to as the *broadcast storm problem* [1]. The fact that *only a subset of nodes forward the broadcast message and the remaining nodes are adjacent to forward nodes* can be used to reduce the broadcast congestion but still fulfill the broadcast operation. Basically, forward nodes form a *connected dominating set* (CDS). A *dominating set* (DS) is a subset of nodes such that every node in the graph is either in the set or is adjacent to a node in the set. If the subgraph induced from a DS of the network is connected, the DS is a CDS. Finding a *minimum connected dominating set* in a given graph is NP-complete; in a unit disk graph, it has also been proved to be NP-complete [2].

Along with the high transmission contention and congestion, MANETs also suffer from the high transmission error rate in radio environment. Therefore, it is a major challenge to provide a reliable broadcasting under such dynamic MANETs. We aim to reduce broadcast congestion by decreasing the number of the forward nodes yet still providing high delivery ratio for each broadcast packet in a high transmission error rate environment. As pointed out by other researchers, providing total reliability for broadcasting in MANETs is impractical and unnecessary when the physical communication channels are prone to errors. Usually, acknowledgments (ACKs) are used to ensure broadcast delivery. However, the requirement for all receivers to send ACKs in response to the reception of a packet may become another bottleneck of channel congestion and packet collision, which is called the *ACK implosion problem* [3] .

Our goal is to reduce the number of forward nodes without sacrificing the broadcast delivery ratio. Specifically, we propose a simple reliable broadcast algorithm, called *double-covered broadcast* (DCB), that takes advantage of broadcast redundancy to improve the delivery ratio in the environment that has rather high transmission error rate. Among 1-hop neighbors of the sender, only selected forward nodes will retransmit the broadcast message. Forward nodes are selected to meet the following two requirements: (1) the sender's 2-hop neighbor set is fully covered and (2) the sender's 1-hop neighbors are either forward nodes or non-forward nodes but covered by at least two forwarding neighbors, the sender itself and one of the selected forward nodes. The retransmissions of the forward nodes are received by the sender as the acknowledgement of their reception of the packet. The sender's non-forward 1-hop neighbors do not acknowledge the reception of the broadcast. If the sender fails to detect all its forward nodes retransmissions during a period of time, it assumes that a transmission failure has occurred for this broadcast, either because of the transmission error or because of the forward nodes' out-of-range movement. The sender re-sends the packet until it detects all forward nodes' retransmissions or the maximum times of retries is reached.

The proposed algorithm has the following merits:

(1) Only the forward nodes transmit the packet so that the broadcast collision and congestion are reduced.

(2) The retransmissions of forward nodes are also used as the ACKs to the sender so that no extra ACKs are needed. This scheme avoids the ACK implosion problem.

(3) The failure to overhear the forward node retransmission will trigger the sender to retransmit the broadcast so that the loss of a broadcast packet can be recovered in a local region.

(4) Each non-forward node is covered by at least two forwarding neighbors so that its chance to receive the broadcast packet successfully is doubled even in a high transmission error rate environment.

Simulation results show that the algorithm provides high delivery ratio, low forwarding ratio, low overhead and low end-to-end delay for a broadcast operation under high transmission error rate environment.

The remaining part of the paper is organized as follows: Preliminaries are briefly introduced in Section 2. Section 3 describes in detail the double-covered broadcast protocol. In Section 4, we simulate the broadcast protocol by using ns-2 test-bed and compare its performance with other reliable broadcast algorithms. Finally, some conclusions are drawn in Section 5.

II. Preliminaries

We describe a MANET as a unit disk graph $G(t)$ =(V, E), where the node set V represents a set of wireless mobile nodes and the edge set E represents a set of bi-directional links between the neighboring nodes. Two nodes are considered neighbors if and only if their geographic distance is less than the transmission range r. A *view* with respect to a particular broadcast process is a snapshot of network topology and broadcast state. The status of each node is determined by itself or by its neighbor based on a particular local view. All views used for status decision of nodes are made within a period, so $G(t)$ can be simply represented as G. For a specific node, the upstream node that has sent a broadcast packet to this node is viewed as a *forwarded node*. A *forward node* is a downstream node designated by this node that will forward the broadcast packet; a *non-forward node* is a downstream node that is designated not to forward the packet. Notice that the node status under the current view will be changed in the next view, that is, a forward node in current view will be a forwarded node in the next view.

For convenience, we use $N_k(v)$ to represent the k-hop neighbor set of v, where nodes in the set are no more than k hops further from v. $N_k(v)$ includes v itself. ($N_1(v)$, 1-hop neighbor set, can be simply represented as $N(v)$.) If S is a node set, $N(S)$ is the union of the neighbor sets of every node in S, that is, $N(S) = \cup_{\forall w \in S} N(w)$.

A. Neighbor-Designating-Based Broadcasting

In [4], Wu and Dai proposed a generic distributed broadcast scheme in which a CDS is constructed for a particular broadcast and dependent on the location of the source and the progress of the broadcast process. Each node v determines its status and the status of some of its neighbors under a current local view. Two categories of broadcasting approaches, called *self-pruning* and *neighbor designating* broadcasting approaches, are classified. We are interested in the class of neighbor designating broadcasting approach, where a node can determine its neighbor's forwarding/non-forwarding status. All of the following algorithms belong to this class and adopt the greedy strategy where a minimum number of designated forward nodes are selected so that other neighbors can take the non-forward status.

In [5], Qayyum et al proposed selected multipoint relays (MPRs) as forward nodes to propagate link state messages in their optimized link state routing (OLSR) protocol. The MPRs are selected from 1-hop neighbors to cover 2-hop neighbors. Forwarded nodes are not considered for a node to select its MPRs and, therefore, the entire set of 2-hop neighbors must be covered. A relaxed neighbor-designating requirement is applied in [6]: if an MPR first receives a broadcast packet from a neighbor that is not its designator, it does not forward this packet (Figure 1 (a)).

Specifically, if u intends to forward a packet, u selects its forward node set from $X = N(u)$ to cover 2-hop neighbors in $U = N_2(u)$ with a simple greedy algorithm used in the set coverage problem [7]. This *forward node set selection process* works as follows and $\{w_1, w_2, \ldots\}$ forms a forward node set that covers $N_2(v)$.

Algorithm 1 Forward Node Set Selection Process (FNSSP)

```
1. Each node $w$ in $X$ calculates its effective node degree $deg_e(w) = |N(w) \cap U|$.
2. A node $w_1$ with the maximum $deg_e(w_1)$ is first selected, $w_1$ is removed from $X$ and $N(w_1)$ is removed from $U$.
3. If $U$ is not empty, each node re-computes its effective node degree and another node $w_2$ with the maximum $deg_e(w_2)$ is selected.
4. Repeat steps 2 and 3 until $U$ becomes empty.
```

Lim and Kim [8] provided a dominant pruning algorithm (DP). Unlike the MPR, the DP excludes the coverage of the forwarded node from the current node's 2-hop neighbor set. Suppose u is the last forwarded node and v is designated as the next forward node, v selects its forward node set from $X = N(v) - N(u)$ to cover 2-hop neighbor set $U = N_2(v) - N(u) - N(v)$ (Figure 1 (b)).

Lou and Wu [9] proposed partial dominant pruning algorithm (PDP) to extend the DP by further reducing the number of 2-hop neighbors to be covered by 1-hop neighbors. In

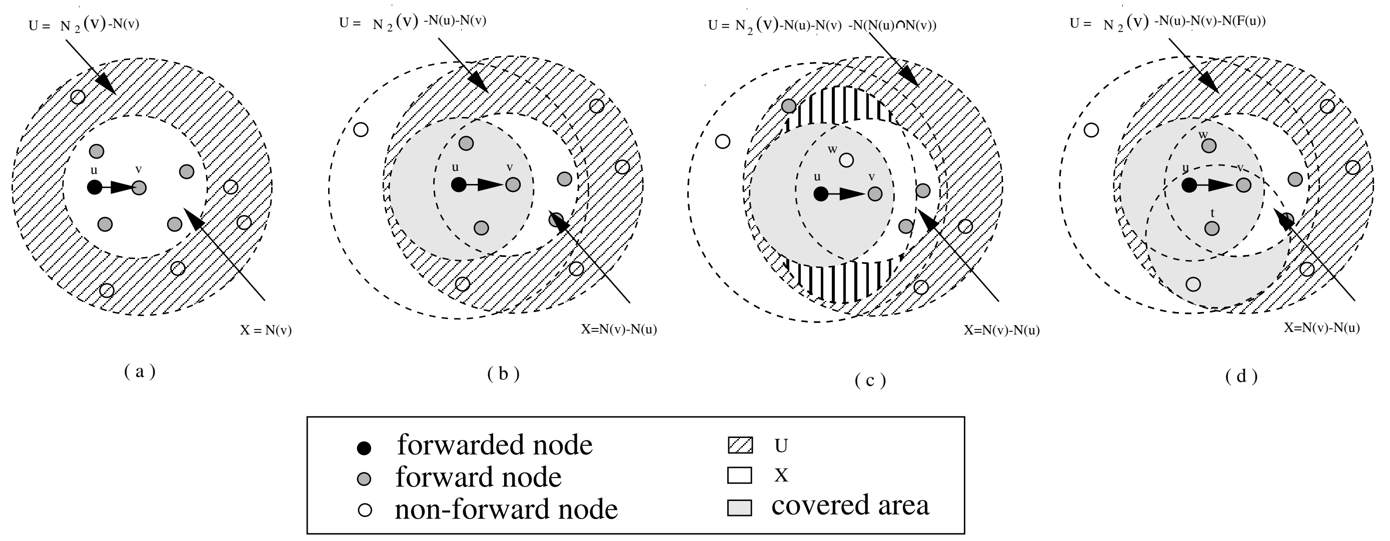

Fig. 1. Illustrations of four algorithms: (a) multiple relays (MPR), (b) dominant pruning (DP), (c) partial dominant pruning (PDP) and (d) CDS-based broadcasting (CDSB).

the PDP, the node v extracts the neighbors of the common neighbors of u and v (i.e., neighbors of nodes in $N(u) \cap N(v)$) from $N_2(v)$, that is, the uncovered 2-hop neighbor set $U = N_2(v) - N(u) - N(v) - N(N(u) \cap N(v))$ (Figure 1(c)).

Peng and Lu proposed a CDS-based broadcast algorithm (CDSB) in [10]. It considers not only the sender of the broadcast packet but also the forward nodes with lower node IDs that are selected by the sender to determine a selected forward node's forward node set. For a sender u, suppose u selects nodes t, v, w ($id(t) < id(v) < id(w)$) as its forward nodes. When nodes t, v, w receive the packet, t updates its uncovered 2-hop neighbor set $U(t) = N_2(t) - N(u) - N(t)$; v updates its uncovered 2-hop neighbor set $U(v) = N_2(v) - N(u) - N(t) - N(v)$ because $N(t)$ is covered by t. Likewise, w's uncovered 2-hop neighbor set is $U(w) = N_2(w) - N(u) - N(t) - N(v) - N(w)$ (Figure 1 (d)). Notice that v will not forward the packet if $U(v)$ is empty.

B. Reliable Broadcasting

In general, a reliable communication needs some feedback from receivers. Many approaches are provided for reliable communications in wired networks [11], [12], [13], [14]. The basic categories of reliable communication schemes are *sender initiated* and *receiver initiated* approaches [15]. In the sender initiated approach [11], [14], the receiver returns a positive ACK to the sender for each message it receives. The sender needs to maintain all records for each receiver to confirm the success of the delivery. Only missing packets are retransmitted by the sender, either to individual requested receivers, or to all receivers. The drawback of this scheme is that the sender may become the bottleneck of transmission when simultaneous ACKs return. Moreover, the amount of records that the sender must maintain may also grow large. In the receiver initiated approach [12], [13], the receiver is responsible for reliable delivery. Each receiver maintains receiving records and requests repairs via a negative acknowledgement (NACK) when errors occur. Several strategies can be applied for the receiver initiated approach, such as sender-oriented, flat-receiver-oriented and hierarchical-receiver-oriented approaches. The problem of the receiver initiated approach is the long end-to-end delay since the sender must wait for the next broadcast packet to determine if the previous one is successfully delivered or not. Therefore, it can be applied only when the sender has many packets to be sent.

There are several reliable broadcast schemes ([3], [16]) that aim to suppress MAC layer's collision and provide reliable MAC layer transmission. In the network layer, most reliable broadcast protocols come from the routing protocol proposed by Merlin and Segall [17]: The source starts a broadcast operation by sending a message to all its neighbors and waiting for the ACKs from its neighbors. When it receives all these ACKs, it sends the message asking the neighbors to propagate the message one more hop to their own neighbors. The neighbors of the source forward the message to their neighbors and send the ACKs back to the source when they receive all ACKs from all their own neighbors, and so forth. The scheme incurs too much communication overhead and needs stable linkages for MANETs.

A flooding-based reliable broadcast protocol proposed by J. J. Garcia and Zhang [18] allows the nodes that received the broadcast packet to forward the packet without further notice from the sender. Alagar and Venkatesan [19] also proposed a reliable broadcast protocol based on flooding. The protocol works as follows: The source broadcasts the message to its 1-hop neighbors. When a node receives the message, it sends an ACK back to the sender. If the message is a new one, the node retransmits the message; otherwise, it drops the message. If the sender does not receive an ACK from any of its neighbor for a predefined period, it re-sends the message. In case some links happen to be broken up, a handshake process is provided to make two neighbor nodes exchange all of the messages they have so far to keep all records identical. The obvious drawback of these flooding-based protocols is that the flooding may easily introduce the broadcast storm problem. The ACK implosion problem may worsen the broadcast storm problem.

Pagani and Rossi [20] proposed to set up a forwarding tree, which is rooted from the clusterhead of source to each clusterhead, based on a virtual cluster architecture for a reliable broadcast in MANETs. The broadcast packet is forwarded downward the tree from the root source to the leaf nodes and the ACKs are collected by each clusterhead and upward the tree from the leaves to the root. The source retransmits the packet if error occurs. The algorithm changes to flooding when the rate of topology change of the network becomes high. Apparently, maintaining the underlying cluster and the forwarding tree is almost impractical in dynamic MANETs.

All the above reliable broadcast algorithms require each receiver to send ACKs in response to the reception of a packet. These ACKs may become another bottleneck of channel congestion and lead to severe transmission contention and collision which is referred to as ACK implosion problem.

III. A Double-Covered Broadcast Algorithm

A. Basic Idea

A broadcast operation requires the packet be disseminated to all nodes in the network. But the interference of the transmission of neighbors and the movement of the nodes may cause the failure of some nodes to receive the broadcast packet. The broadcast redundancy can provide more chance for a node to successfully receive the packet. If the sender can retransmit the missed packet, the broadcast delivery ratio can also be improved.

The proposed double-covered broadcast algorithm works as follows: When a sender broadcasts a packet, it selects a subset of 1-hop neighbors as its forward nodes to forward the broadcast based on a greedy approach. The selected forward nodes satisfy two requirements: (1) They cover all the nodes within 2 hops of the sender. (2) The sender's 1-hop neighbors are either forward nodes or non-forward nodes but covered by at least two neighbors, once by the sender itself and once by one of the selected forward nodes. After receiving the broadcast packet, each forward node records the packet, computes its forward nodes and re-broadcasts the packet as a new sender. The retransmissions of the forward nodes are received by the sender as the acknowledgement of receiving the packet. The non-forward 1-hop neighbors of the sender do not acknowledge receipt of the broadcast. The sender waits for a predefined duration to overhear the rebroadcasting from its forward nodes. If the sender fails to detect all its forward nodes retransmitting during this duration, it assumes that a transmission failure has occurred for this broadcast because of the transmission error or because the missed forward nodes are out of its transmission range. The sender then re-sends the packet until all forward nodes are retransmitted or the maximum number of retries is reached.

The proposed algorithm utilizes the method that the sender overhears the retransmission of the forward nodes to avoid the ACK implosion problem. Also, the algorithm guarantees that each node is covered by at least two transmissions so that

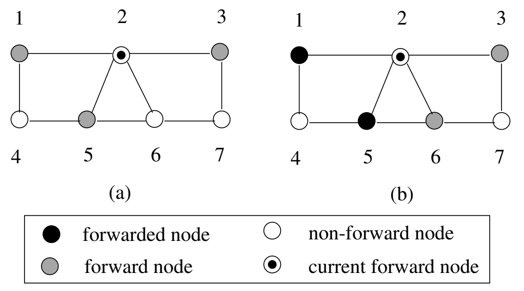

Fig. 2. A sample network where the node 2 uses the FNSSP to select its forward nodes in (a) case 1 and (b) case 2.

it can avoid a single error due to the transmission collision. Moreover, the algorithm does not suffer the disadvantage of the receiver-initiated approach that needs a much longer delay to detect a missed packet.

B. Forward Node Set Selection Process

We suppose neighboring nodes exchange their 1-hop neighbor set information with each other and, therefore, each node v has its 2-hop neighbor set information $N_2(v)$. The forward node set selection process executes at each forward node to determine its own forward node set. Therefore, each node v in $N(u)$ can be one of two cases: 1) v is a forward node that will forward the broadcast packet; 2) v is a non-forward node which is adjacent to at least two nodes that will forward the broadcast packet: one is u and the other is the forward node that also covers v. Therefore, v has at least two chances to correctly receive the broadcast packet. Unlike in the DP algorithm [8] where only nodes in $N_2(u) - N(u)$ need to be covered by forward node set $F(u)$, the selection process here guarantees each node in $N(u)$ is covered by the forward node set $F(u)$.

We consider the two cases where a node v determines its forward node set $F(v)$: (1) v is the source of the broadcast: v uses FNSSP algorithm to find $F(v)$ in $X = N(v)$ to cover $U = N_2(v)$. (2) v is a selected forward node to relay the broadcast packet: Suppose v has already received the packet from a node set $V(v)$ and each node w in $V(v)$ has its own forward node set $F(w)$. v uses FNSSP algorithm to find $F(v)$ in $X = N(v) - V(v) - \cup_{\forall w \in V(v)} F(w)$ to cover $U = N_2(v) - N(V(v)) - \cup_{\forall w \in V(v)} N(F(w))$.

The correctness of the algorithm is easy to prove. We need to prove that (1) a forward node v's 2-hop neighbor set $N_2(v)$ is completely covered and (2) v's 1-hop neighbor set $N(v)$ is covered twice. Case 1 is a special case of case 2. And the correctness of case 2 is obvious since the algorithm guarantees that $N_2(v)$ is covered by the union of $V(v)$, $\cup_{\forall w \in V(v)} N(F(x))$ and $F(v)$ where all nodes in $V(v)$, $\cup_{\forall x \in V(v)} N(F(x))$ and $F(v)$ are designated to forward the packet. More over, $N(v)$ is also fully covered by v itself. Therefore, $N(v)$ is covered twice.

In the sample network shown in Figure 2 (a), $N(2) = \{1, 2, 3, 5, 6\}$ and $N_2(2) = \{1, 2, 3, 4, 5, 6, 7\}$. When using the FNSSP, sender node 2 selects nodes 1, 3 and 5 as its forward nodes. Node 1 is selected because there is no node in $N(1)$

to cover it. In Figure 2 (b), suppose the source of a broadcast is node 4, and node 2 has received the broadcast packet from nodes 1 and 5. 1 and 5's forward node sets are $F(1) = \{2\}$ and $F(5) = \{2, 6\}$. Therefore, node 2's uncovered 2-hop neighbor set is $N_2(2) - N(1) - N(5) - N(6) = \{3\}$. Using the FNSSP, node 2 selects node 3 as its forward node.

C. The Double-Covered Broadcast Algorithm

The double-covered broadcast algorithm (DCB) is described as a set of event-driven rules. The following symbols are used:

- $F(v)$: the forward node set of node v.
- $U(v)$: the uncovered 2-hop neighbor set of node v.
- C_v: the counter for the times a packet has been sent by node v.
- T_v: a timer at node v for re-sending the packet.
- $P(s, v, F(v))$: a unique broadcast packet from source s, attaching $F(v)$, and forwarded by node v.
- T_{wait}: the bound on the timer for a sender to overhear the retransmission of all its forward nodes.
- RT_{max}: the maximum times of retries.

We assume a broadcast process starts from source s, s uses the FNSSP algorithm to select its forward node set $F(s)$, and then piggybacks $F(s)$ with the packet and broadcasts the packet among its 1-hop neighbor set $N(s)$.

For a node v that receives a new broadcast packet from an upstream sender u, v initializes its uncovered 2-hop neighbors $U(v) = N_2(v)$. If v is a forward node (i.e., v is in $F(u)$), v updates $U(v)$ by excluding $N(u)$ and $N(F(u) - \{v\})$, that is, $U(v) = U(v) - N(u) - N(F(u) - \{v\})$. The reason that $U(v)$ can exclude $N(u)$ and $N(F(u) - \{v\})$ is explained in the above section. If v receives the packet for the first time, it computes its forward nodes $F(v)$ to cover its updated uncovered neighbor set $U(v)$ and broadcasts the packet among $N(v)$; otherwise, v locally broadcasts the packet. The reasons that v locally broadcasts the packet are (1) to satisfy the requirement of the double coverage of non-forward neighbors and (2) to acknowledge to the sender the reception of the broadcast packet.

The sender u broadcasts the packet and waits for a duration T_{wait} to overhear the retransmission of its forward nodes. If u overhears a retransmission packet from its forward node v, v will be removed from $F(u)$. If u does not overhear from all of its forward nodes during this duration, it assumes the transmission failure has occurred for this broadcast packet. u then computes new $F(v)$ to cover the rest of the uncovered $U(v)$ and re-sends the packet until the RT_{max} limit is reached and stops further broadcast attempts.

D. Reliability issues

When a sender transmits a packet to all its neighbors, a neighbor may fail to receive this packet because of a transmission collision with other neighbors, the high transmission error rate of the radio channel, or the out-of-range movement of the node.

We treat the non-forward node and forward node differently: When a non-forward node v missed the packet (Figure 3 (a)), based on the FNSSP, v has been at least covered by two forwarding nodes u and f; even when v missed the packet from u, it still has a second chance to receive the packet from f. Note that a non-forward node that missed the packet does not cause other transmission error propagations in the network. When a forward node f missed the packet, it may cause the transmission error to propagate since forward nodes are the key nodes in the network that need to relay the broadcast packet. There are two main causes for the packet loss:

Transmission collision and high transmission error rate: In figure 3 (b), if f missed the transmission from u because of the transmission collision or transmission error of the radio channel, the nodes in the shaded area may also miss the packet. The simple *Re-send* algorithm is adaptive to this case: u waits a period of time T_u when it sends a broadcast packet. If u fails to detect f's retransmission signal during T_u, u re-sends the packet until the maximum retry is reached.

Out-of-range movement of the node: A selected forward node may move out of the range of the sender node, and this

Algorithm 2 The Double-Covered Broadcast algorithm (DCB)

```
For new packet starts from source $s$
    $s$ uses FNSSP to find $F(s)$ in $N(s)$ to cover $N_2(s)$
    $C_s = 0$
    $T_s = T_{wait}$
    $s$ broadcasts $P(s, s, F(s))$ among $N(s)$
When $v$ receives $P(s, u, F(u))$ from $u$
    if $P$ is a new one then
        $U(v) := N_2(v)$
        $X(v) := N(v)$
    else
        $U(v) := U(v) - N(u) - N(F(u) - \{v\})$
        $X(v) := X(v) - F(u) - \{u\}$
    end if
    if ($v \in F(u)$) then
        if ($P$ has never been received) then
            $v$ uses FNSSP to find $F(v)$ in $X(v)$ to cover $U(v)$
            $C_v := 0$
            $T_v := T_{wait}$
            $v$ broadcasts $P(s, v, F(v))$ among $N(v)$
        else if ($P$ has never been sent) then
            $v$ locally broadcasts $P(s, v, \phi)$ among $N(v)$
        end if
    end if
    if $u \in F(v)$ then
        $F(v) := F(v) - \{u\}$
    end if
When timer $T_v$ is expired
    if $(F(v) \neq \phi) \land (C_v < RT_{max})$ then
        $C_v := C_v + 1$
        $T_v := T_{wait}$
        $v$ uses the Re-send(or Re-select/Re-calculate) algorithm
        to determine $F(v)$
        $v$ broadcasts $P(s, v, F(v))$ among $N(v)$
    end if
```

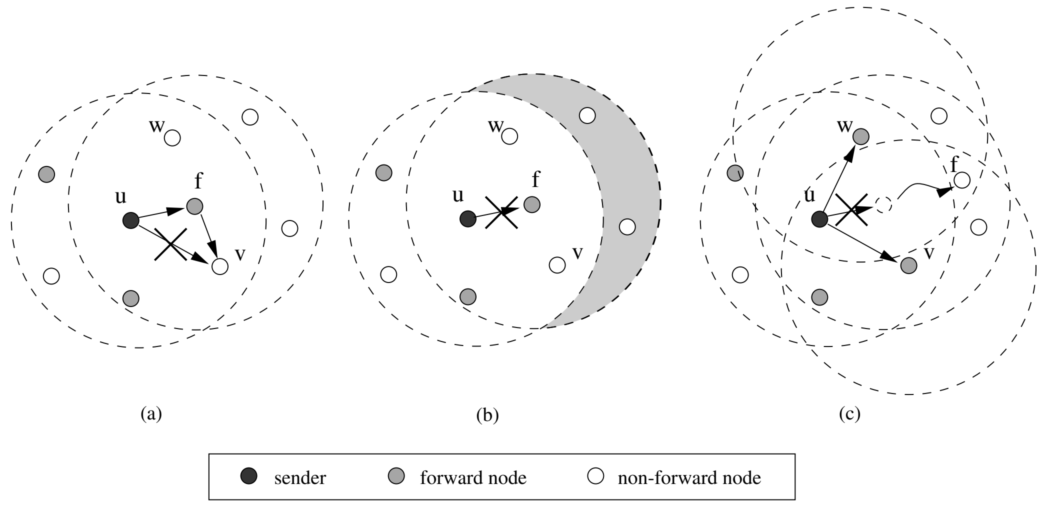

Fig. 3. An illustration of transmission errors: (1) a transmission error occurs at a non-forward node n. (2) transmission error occurs at a forward node f that causes nodes in the shaded area to miss the packet. (3) alternative forward nodes m and n are selected to cover the area that is supposed to be covered by the missed forward node f.

results in a transmission failure. In figure 3 (c), f moved out of the transmission range of u and missed the packet. The *Re-select* algorithm is used for this case: When u fails to detect f's retransmission signal during T_u, u supposes f has moved out of its range and re-selects alternative forward nodes to cover the area which is supposed to be covered by f.

More specifically, suppose u selects its forward node set $F(u)$=$\{f_1, f_2, ...f_m\}$ and sends the broadcast packet. u waits for T_u and does not detect the retransmission from the forward nodes $f'_1, f'_2, ..., f'_k$. The uncovered $U(u)$ is $N_2(u) - N(F(u) - \bigcup_{i=k}^{i=1}\{f'_i\})$. The selection criteria are as follows: (1) Add $f_n \in N(u)$ in $F(u)$ such that f_n is the only node that covers some nodes in $U(u)$. (2) Add $f_n \in N(u)$ in $F(u)$ such that f_n covers the largest number of nodes in $U(u)$. If there is a tie, the node that sent HELLO message most recently has the highest priority. (3) Set the nodes $f'_1, f'_2, ..., f'_k$ to the least priority to be selected even though they may cover more nodes in $U(u)$ than other nodes. In Figure 3 (c), when u does not overhear f's retransmission, s may select v and w to substitute f for rebroadcasting.

In the above two cases, u does not know if f is out of its range or not. If u can refresh its neighbor set on time, u can recalculate its forward node set on demand when it needs to re-send the duplicated packet based on the FNSSP algorithm. This method, called *Re-calculate* algorithm, is suitable for the case when some new nodes move into the transmission range of u and u re-sends its stored packets locally. The downside of this algorithm is its long delay since each node has to wait for enough time to gather all neighbor's HELLO messages for refreshing neighbor set information.

E. Networks with Asymmetric Links

In the above discussion, we assume all nodes have the same transmission range r. Therefore, the generated network is always symmetric. In real ad hoc networks, asymmetric links may occur for several reasons including different transmission

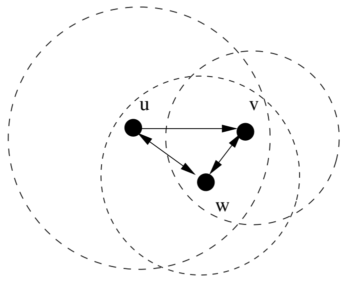

Fig. 4. A sample network with asymmetric links.

ranges of nodes, local congestion, use of directional antennas and external interference. For networks with asymmetric links, a conservative approach is that two nodes consider each other neighbors only when they are both within the transmission ranges of each other. The asymmetric links are ignored in order to apply the ACK mechanism used only in the presence of symmetric links.

Asymmetric links can also be used to send ACKs. That is, the receiver uses a directed path with multiple nodes to send the ACK back to the sender to confirm the reception of the packet. Figure 4 shows such a case: Due to the different transmission ranges of nodes u, v and w, there is an asymmetric link (u, v) (from node u to v), and symmetric links between v and w and between w and u. v realizes an asymmetric link (u, v) if v receives the HELLO message from u with u's 1-hop neighbor set $N(u) = \{w\}$ and finds itself not in $N(u)$. v starts a local broadcast REQ with TTL=2 to find u. Intermediate node w attaches its ID and forwards the REQ. When u receives the REQ, it recognizes the asymmetric link (u, v), builds the feedback path (v, w, u) and informs v of the feedback path. Then v can use this path to send the ACK to u via w. Thus, a node can build a feedback path with one intermediate node. In this way, the node can build paths to its 1-hop neighbors with at most one intermediate node. Notice

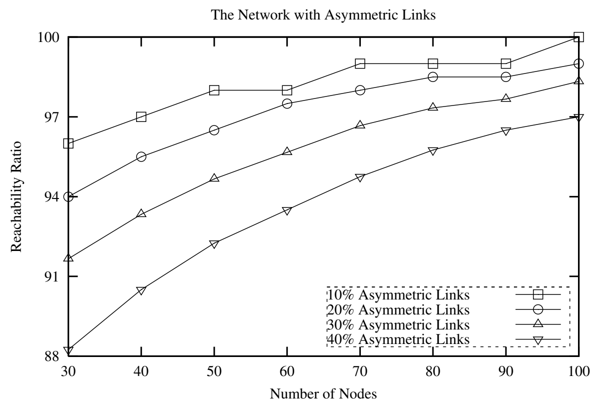

Fig. 5. The reachability ratio of the network with asymmetric links.

that the links in the path (v, w, u) can also be asymmetric links.

Figure 5 shows the reachability ratio of the network with asymmetric links. Reachability ratio is defined as the ratio of number of asymmetric links that have feedback path with at most one intermediate node to the total number of asymmetric links in the network. In Figure 5, the links of the network are composed of different percentages of asymmetric links from 10% to 40%. We can see that the reachability ratio increases when the network becomes dense. Also, even when the asymmetric links are as high as 40% of total links, only 3% of total asymmetric links have no feedback paths with one intermediate node in the case that the size of network is 100. The connectivity of the network, therefore, can be greatly improved.

For the DCB algorithm, the ACK message is omitted because the sender can overhear the broadcast message when the selected forward nodes retransmit it. Thus, the ACK message can be saved. In an asymmetric network, if the selected forward node, v, has an asymmetric link from its upstream sender u, v's retransmission can not be heard by u and v needs to send an explicit ACK through the feedback path (v, w, u) to confirm its reception. In this case, extra ACK overhead from w to u is introduced. That is, one explicit ACK from the intermediate node to the sender is needed. Note that since v is a forward node, there is no additional overhead for messages from v to w.

F. Probabilistic analysis

We study the probability increase for the following case: a node that should forward a broadcast packet but did not because of the transmission error. We assume an error model as follows: A transmission error may occur at both ends of a wireless channel, that is, an error occurring at the sender may cause all its neighbors to lose the packet and an error occurring at the receiver may only affect the current receiver but does not affect other receivers. We assume that the errors follow the uniform distribution with probability p_{err} at both ends of a wireless channel so that nodes can only probabilistically send messages to their neighbors.

For a single transmission from u to v (Figure 3 (a)), the probability of a successful transmission is

$$P_s = (1 - p_{err})(1 - p_{err}) \quad (1)$$

and the probability of a failed transmission is

$$P_f = 1 - P_s = 2p_{err} - p_{err}^2 \quad (2)$$

With the retransmission mechanism, a sender can resend the packet several times if it does not overhear its forward node's retransmission signal, and the probability for a node to successfully receive the message increases. For a forward node f, the probability of a failed reception is

$$P_f'(f) = P_f^R \quad (3)$$

where R is the maximum times of retry and the probability of a successful reception is

$$P_s'(f) = 1 - P_f'(f) = 1 - P_f^R \quad (4)$$

For a non-forward node v, its probability of success is at least

$$P_s'(v) = P_s'(f) + P_f'(f)P_s'(f)P_s \quad (5)$$

We now calculate the probability that a forward node may correctly forward a broadcast packet. In the 1-hop neighbor set of f, suppose there are m forwarded nodes (black nodes) that select f as a forward node and n forwarded nodes (gray nodes) that select f as a non-forward node. The probability that f correctly forwards a packet is equaled to the probability that the first transmission attempt is from a black node and the first successful transmission is also from a black node.

Without the retransmission mechanism, the probability of a transmission attempt from a black node is

$$\frac{m}{m+n} \quad (6)$$

and the probability of a successful transmission from a black node is

$$\begin{aligned} P(m,n) &= \frac{m}{m+n}(P_s + P_f P(m-1,n)) \\ &\quad + \frac{n}{m+n} P_f P(m,n-1) \end{aligned} \quad (7)$$

where $P(0,n)$ = 0, $P(m,0)$ = $1 - P_f^m$, $\forall m, n > 0$. Therefore, the probability that f correctly forwards a packet is

$$P_{nr}(m,n) = \frac{m}{m+n}(P_s + P_f P(m-1,n)) \quad (8)$$

If each forward node has the retransmission mechanism and suppose each forward node can retry up to R times, which is equal to the case that there are Rm black nodes and Rn gray nodes in f's neighborhood, then the probability that f may correctly receive a packet from a black node first is

$$P_r(m,n) = P_{nr}(Rm, Rn) \quad (9)$$

For example, suppose m=4, n=2, R=3, p_{err}=0.3, by using (1),(4),(5),(8) and (9), we get

$$P_s = 0.49, P_s'(f) = 0.8673, P_s'(v) = 0.9237,$$

$$P_{nr}(3,5) = 0.5236, P_r(3,5) = 0.5467,$$

$$\Delta = P_r(3,5) - P_{nr}(3,5) = 0.0230.$$

TABLE I

SIMULATION PARAMETERS

Parameter	Value
Simulator	ns-2 (version 2.26)
Network Area	$900 \times 900\ m^2$
Transmission Range	$250\ m$
MAC Layer	IEEE 802.11
Data Packet Size	64 bytes
Bandwidth	2 M b/s
Simulation Time	$100\ s$
Number of Trials	20
Confidence Interval	95%

TABLE II

SIMULATED ALGORITHMS

Algorithm	Description		
	Transmit	Acknowledge	Retransmit
DCB-SD	forward nodes	forward nodes	Re-send
DCB-ST	forward nodes	forward nodes	Re-select
DCB-RE	forward nodes	forward nodes	Re-calculate
DP[8]	forward nodes	none	none
BF	all nodes	none	none
RBAV[19]	all nodes	all nodes	flooding

IV. SIMULATIONS

A. Simulation descriptions

In order to analyze the performance of the proposed algorithm, we ran the simulation under the ns-2 test bed with CMU wireless extension. The simulator parameters are listed in Table I: The network area is confined within $900 \times 900\ m^2$. Each node in the network has a constant transmission range of $250\ m$. We use *two-ray ground reflection model* as the radio propagation model. The MAC layer scheme follows the IEEE 802.11 MAC specification. We use the broadcast mode with no RTS/CTS/ACK mechanisms for all message transmissions, including HELLO, DATA and ACK messages. Since transmission errors may occur when nodes send messages in real wireless channels, we use the uniform distribution with probability p_{err} at both ends of a wireless channel as the error model. The movement pattern of each node follows the *random way-point model*: Each node moves to a randomly selected destination with a constant speed between 0 and the maximum speed V_{max}. When it reaches the destination, it stays there for a random period T_s and starts moving to a new destination. The pause time T_s is always 0 in our simulation. The network traffic load also affects the performance of the protocol; we change the value of constant-packet-rate CPR (packet per second) while each packet has a constant length of 64 bytes. A node may fail to receive a message because of a transmission error, a transmission collision or the node's out-of-range movement. After sending a message, a node will wait for a period of time T_{wait} and resend the message until it reaches the maximum value RT_{max}. Each simulation was run for 100 seconds and run 20 times to achieve the 95% confidence interval for the results.

1) Chosen algorithms: We compare the performances of the algorithms listed in Table II through simulations to see the benefits and losses of the double-covered broadcast algorithm.

2) Simulation metrics: We measure the following metrics:

(a) *Broadcast delivery ratio*: Broadcast delivery ratio is the ratio of the nodes that received packets to the number of the nodes in the network for one broadcast operation.

(b) *Broadcast forwarding ratio*: Broadcast forwarding ratio is the fraction of the total number of the nodes in the network that at least retransmit broadcast packets once for one broadcast operation.

(c) *Broadcast overhead*: Broadcast overhead measures the extra data of the control packets, including HELLO and ACK messages, sent by each node for successfully accomplishing one broadcast operation. It is measured by bytes per broadcast byte.

(d) *Broadcast end-to-end delay*: Broadcast end-to-end delay measures the period from the time the source broadcasted the packet to the time the last node receives the packet or no more nodes re-send the packet for one broadcast operation.

3) Affected parameters: We consider the following parameters that affect the performance of the broadcast:

(a) *Network size (n)*: The number of nodes in a network determines the density of the network. A dense network will easily cause the collision and contention.

(b) *Transmission error rate (p_{err})*: The physical radio channel is affected by many environment parameters. Therefore, the Signal-to-Noise Ratio (SNR) at the receiver may be below the threshold even though the receiver is in the transmission range of the sender. This affect can be estimated as a transmission error rate p_{err}, which specifies the transmission error model that messages may have been lost at both ends of a channel.

(c) *Mobility of the node (V_{max})*: The mobility of the node affects the performance of the broadcast operation. The faster the node moves, the higher possibility of the node to lose the broadcast packet.

(d) *Network data traffic load (CPR)*: A heavy data traffic load will cause the network congestion that sharply deteriorates the performance of the broadcast operation.

(e) *Interval of HELLO message (T_{HELLO})*: Since the nodes get neighbor information through HELLO messages, the hello interval determines the accuracy of one node's neighbor set. A large value of the interval will cause the information of the neighbor set to be out-of-date which misleads the forward node's broadcast decision. But increasing the frequency of the interval also increases the cost and causes network congestion because sending HELLO messages compares to a flooding operation.

(f) *The times of retry (RT_{max})*: It is intuitive that to increase the times of retry can improve the broadcast delivery ratio but also increase the end-to-end delay. Also, if RT_{max} is set to 0, the algorithm can only get benefit from double coverage but not from message resend mechanism. By default, we set RT_{max} to 1.

(g) *Waiting time (T_{wait})*: The period of the waiting time for overhearing forward nodes' retransmissions also affects the behavior of a node's broadcasting retransmission. If the value of T_{wait} is compared to the broadcast delay, the sender will

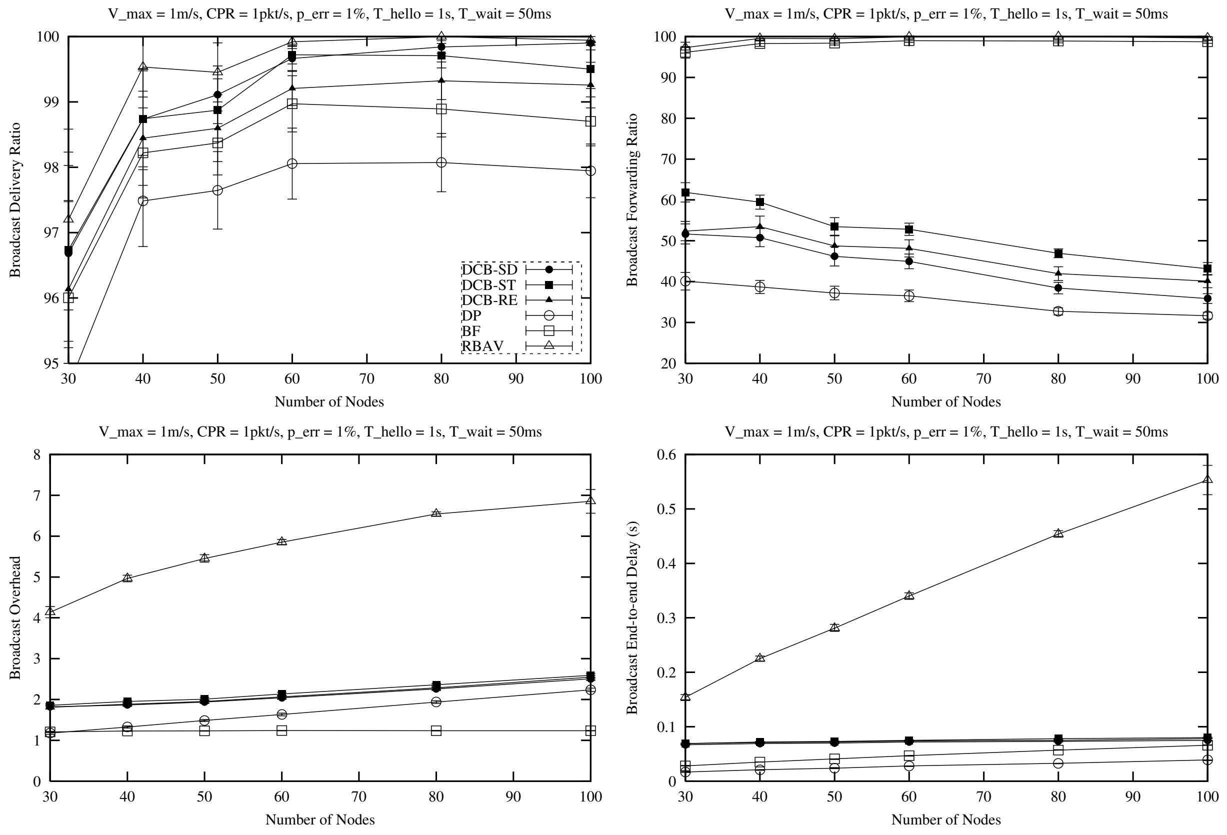

Fig. 6. Sensitivity to size of the network: (a) delivery ratio, (b) forwarding ratio, (c) overhead and (d) end-to-end delay.

resend a second copy of the packet once the first one is missed. In this case, the receiver will have the best chance to receive a broadcast packet from a shortest path from the source. If, on the other hand, the value of T_{wait} is much larger than the broadcast delay, the node that missed the packet is more likely to receive the packet from another neighbor's relaying when a transmission error occurs.

B. Results and Analysis

1) Sensitivity to network size: Figure 6 shows the case where the network has low mobility (V_{max} = 1 m/s), low transmission error rate (p_{err} = 1%), very low data traffic load (CPR = 1 pkt/s), typical hello interval (T_{HELLO} = 1 s) and waiting time (T_{wait} = 50 ms). We identify the affect of network size n to each metric. The network under this environment can be considered as a static error free network. Most of the packet losses come from the transmission collisions.

Figure 6(a) shows the broadcast delivery ratio. We can see that under such environment, all algorithms have high delivery ratios ($> 95\%$). The delivery ratios of all DCB algorithms (DCB-SD, DCB-ST, DCB-RE) are higher than DP and BF which benefit from the retransmission mechanism. Among all three DCBs, the DCB-SD outperforms the other two. The RBAV has the best delivery ratio. Notice that even under such a static and very low traffic load environment, BF cannot guarantee 100% coverage. When the size of the network is small (n=30), the network may sometimes disconnect which leads to the delivery ratio lower than that in a large size network. Figure 6(b) shows the broadcast forwarding ratio. Both BF and RBAV have almost every node forwarding while all DCBs and DP have less than 50% of total nodes forwarding a broadcast packet. The DP has the least forward nodes but the gap between DP and DCBs are slight as n increases. Figure 6(c) shows the broadcast overhead. Since the traffic load is very low, the control overhead such as neighbor set information and broadcast retransmissions cost more than what they save by reducing forwarding ratio. Therefore, BF has the least value followed by DP and DCBs. RBAV shows the highest overhead of all the algorithms since each node that receives a packet needs to send back an ACK message. Figure 6(d) shows the broadcast end-to-end delay. The DP, BF and DCB have similar short end-to-end delay while RBAV has much longer end-to-end delay.

From this simulation, we can see that the DCBs outperform BF and DP by greatly improving the delivery ratio with very little sacrifice of the forwarding ratio, end-to-end delay and overhead. Although the RBAV has the highest delivery ratio, its other metrics are much worse than other algorithms'. Also, we notice that under very low data traffic load, we cannot use neighbor designating algorithms(such as DCB and DP) that reduce the forwarding ratio to save cost.

2) Sensitivity to transmission error rate: Figure 7 shows the performance of the algorithms under different transmission error rate. In this case, n = 100, V_{max} = 1 m/s, CPR = 1

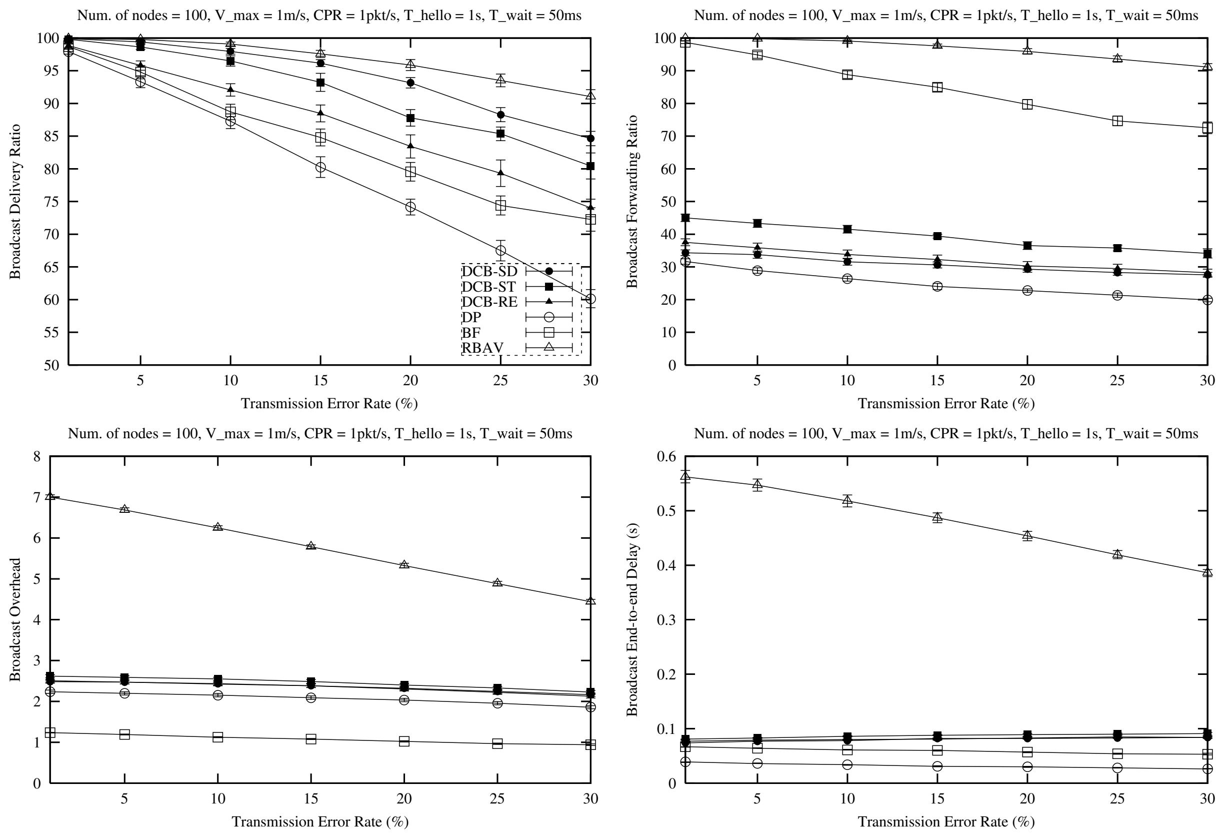

Fig. 7. Sensitivity to transmission error rate of the network: (a) delivery ratio, (b) forwarding ratio, (c) overhead and (d) end-to-end delay.

pkt/s, T_{HELLO} = 1 s and T_{wait} = 50 ms. We change the transmission error rate p_{err} from 0% to 30% to see its affect to each metric.

In Figure 7(a), we see that the delivery ratio is affected by p_{err}. When p_{err} increases, the delivery ratio drops for all algorithms. But the DCBs are better than both BF and DP when p_{err} is high. Among the DCBs, the DCB-SD is better than DCB-ST and DCB-RE(over 10% when n=100). The forwarding ratio of DCBs and DP are much less than BF and RBAV. However, overhead of the DCBs is similar to DP, but larger than the BF (Figure 7(b,c)). The end-to-end delay of DCB is a little larger than DP and BF (Figure 7(d)). As we can see, RBAV has the largest value for forwarding ratio, overhead and end-to-end delay.

From this simulation, we conclude that DCBs outperform DP and BF when p_{err} is high. This is due to the retransmission mechanism of DCB. Compared with RBAV, DCB uses much less broadcast overhead to provide comparable delivery ratio while RBAV needs the high forwarding ratio and overhead and long end-to-end delay to reach high delivery ratio.

3) Sensitivity to mobility of the node: Figure 8 shows the affect of the node's mobility on the performance of broadcast operation. In this case, n = 100, CPR = 1 pkt/s, p_{err} = 1%, T_{HELLO} = 1 s and T_{wait} = 50 ms. We show the affect of the node's mobility to each metric.

Figure 8(a) shows the broadcast delivery ratio of each algorithm. The delivery ratio of BF and RBAV is almost 100% while that of DCBs and DP drop as the node's mobility increases. DCBs are a little better than DP. DCB-ST is better than DCB-SD and DCB-RE, but the difference is slight. Figure 8(b) shows the broadcast forwarding ratio. DCBs and BF have almost the same forwarding ratio and their value decreases as the node's movement increases. The value of forwarding ratio for the BF and RBAV is always 100%. Figure 8(c) and (d) show the broadcast overhead and end-to-end delay. The mobility affects these metrics only slightly.

4) Sensitivity to network data traffic load: In this simulation, we change the network data traffic load CPR from 1 to 80 to see its affect on the performance of a broadcast operation. In this case, $n = 100$, $V_{max} = 1m/s$, $p_{err} = 1\%$, $T_{HELLO} = 1s$ and $T_{wait} = 50ms$. Simulation results show the results that the traffic load affects all the metrics remarkably. The delivery ratio of RBAV drops under 90% when the network is only $4pkt/s$. When CBR is more than 4 pkt/s, RBAV drops sharply since the ack implosion problem occurs. For the other three algorithms, their delivery ratios drop below 90% when the CBR is more than 20 pkt/s. Among all the DCBs, Re-send algorithm works best. For the other three metrics of forwarding ratio,transmission overhead and end-to-end delay, both DCB and DP outperform the BF when CBR is over 5 pkt/s.

5) Sensitivity to hello interval: In order to investigate the affect of hello interval on the performance of the DCB algorithm, we set hello interval T_{HELLO} at 0.05, 0.1, 0.5, 1, 2, 10 s. Here, we use the DCB-SD; other DCB algorithms have similar results. In this case, n = 100, $p_{err} = 1\%$, CPR = 10 pkt/s and T_{wait} = 50 ms. V_{max} ranges from 1 to 160

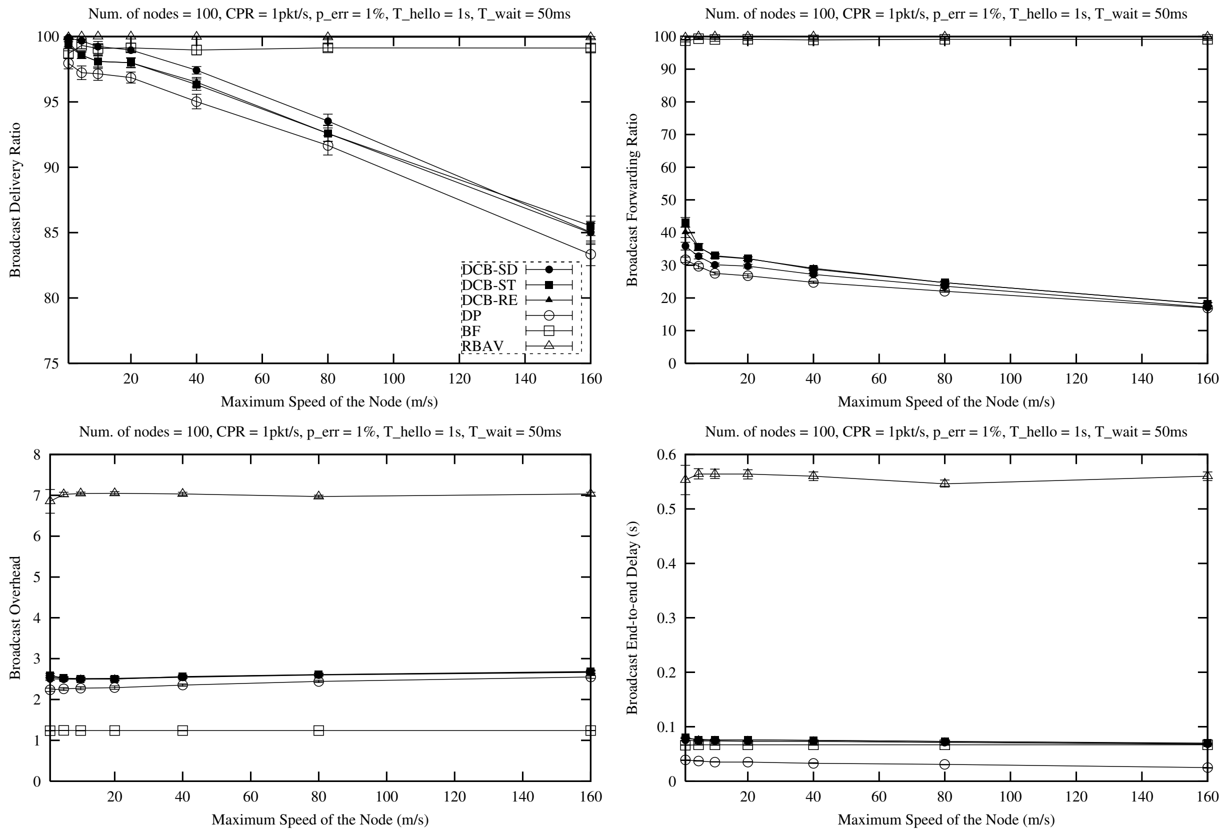

Fig. 8. Sensitivity to mobility of the node: (a) delivery ratio, (b) forwarding ratio, (c) overhead and (d) end-to-end delay.

m/s. Simulation results show that the delivery ratio is highest when T_{HELLO} is 0.5 s and second highest when T_{HELLO} is 1 s (Both are over 90% when V_{max} is 160 m/s). If the hello interval is longer than 1 s or shorter than 0.5 s, the delivery ratio is rather lower. This suggests that the interval of the hello message can not be too short or too long. Simulation results show that updating the hello messages too frequently generates large overhead while updating too infrequently causes the neighbor information to be inaccurate. From these figures, a proper value for the hello interval should be chosen from 0.5 to 1 s.

6) Sensitivity to times of retry: We test the performance of the DCB under different values of RT_{max}. In this case, n = 100, V_{max} = 1 m/s, CPR = 10 pkt/s, T_{HELLO} = 1 s and T_{wait} = 50 ms. RT_{max} is set from 0 to 3. Figure 9 shows the affect of the times of retry on the performance of DCB-SD algorithm. Figure 9 (a) shows that the delivery ratio can be remarkably improved when retransmission mechanism is applied. On the contrary, increasing times of retry only slightly improves the delivery ratio but results in increasing forwarding ratio, broadcast overhead and end-to-end delay (Figure 9 (b-d)). Therefore, the best value for the times of retry is 1.

7) Sensitivity to waiting time: We set T_{wait} at 5, 50, 500, 5000 ms to investigate the affect of waiting time on the performance of DCB-SD. In this case, n = 100, p_{err} = 1%, CPR = 10 pkt/s, T_{HELLO} = 1 s. The value of the waiting time only affects the delivery ratio and end-to-end delay. The DCB algorithm achieved the highest delivery ratio and lowest end-to-end delay when T_{wait} is 50 ms.

V. Conclusions

In this paper, we propose a simple reliable broadcast algorithm that provides high delivery ratio while suppressing broadcast redundancy. This is achieved by requiring only some selected forward nodes among the sender's 1-hop neighbor set to forward the packet. The double covered forward node set selection process provides some redundancy to increase the delivery ratio for non-forward nodes so that retransmissions can be remarkably suppressed when the transmission error is considered. The simulation results show that the double covered broadcast algorithm has high delivery ratio, low forwarding ratio, low overhead and low end-to-end delay for a broadcast operation under high transmission error ratio environment. From the simulation, we observe that the DCB is sensitive to the node's mobility. When the node's mobility increases, the delivery ratio of the DCB drops significantly. The reason for this is that high mobility of nodes makes node neighbor sets outdated quickly. This incorrect neighbor set information may lead to more nodes missing the broadcast packet. An effective method to handle this issue is to allow each node to use two transmission ranges, a small one for sending HELLO messages to find neighbors, and a large one for sending broadcast data messages [21].

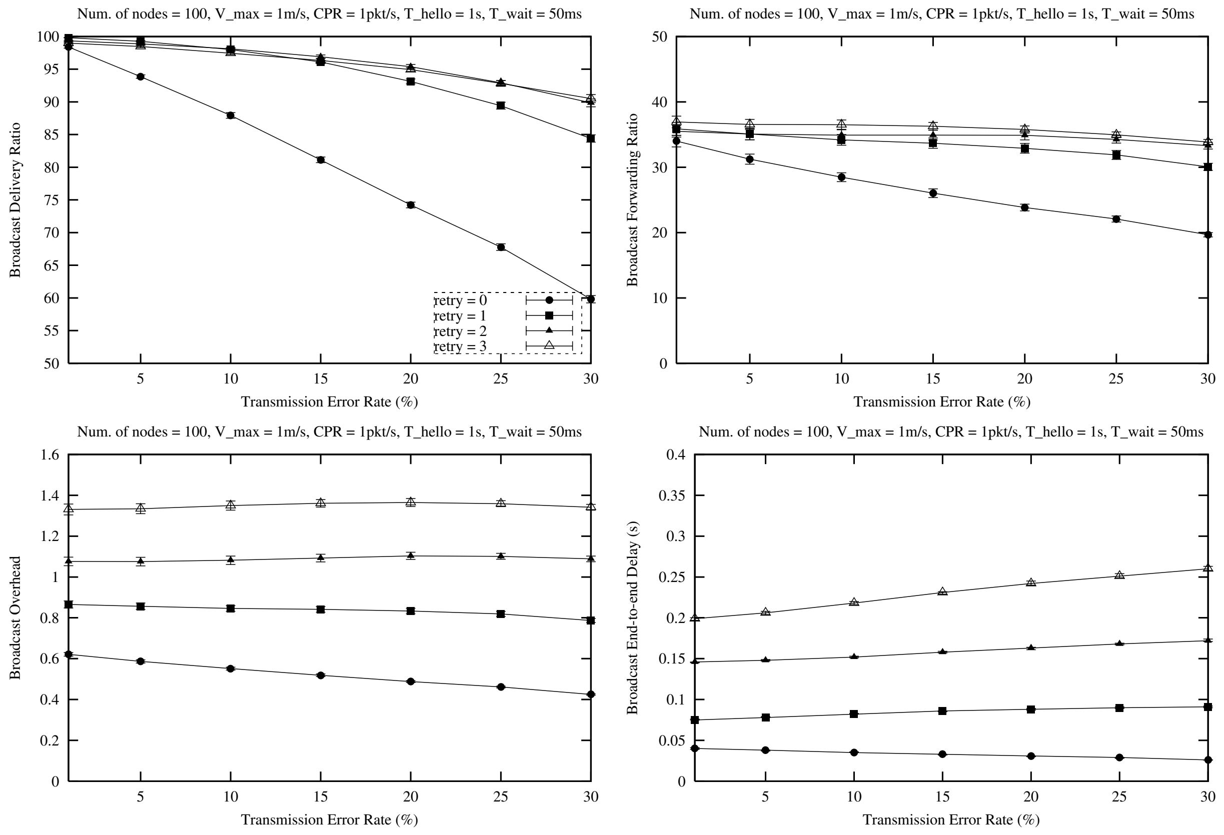

Fig. 9. Sensitivity to times of retry: (a) delivery ratio, (b) forwarding ratio, (c) overhead and (d) end-to-end delay.

VI. Acknowledgement

This work was supported in part by NSF grants ANI 0073736, EIA 0130806 and CCR 0329741.

References

[1] S. Ni, Y. Tseng, Y. Chen, and J. Sheu, "The broadcast storm problem in a mobile ad hoc network," *Proc. of ACM/IEEE MOBICOM'99*, pp. 151–162, Aug. 1999.

[2] M. V. Marathe, H. Breu, H. B. H. III, S. S. Ravi, and D. J. Rosenkrantz, "Simple heuristics for unit disk graphs," *Networks*, vol. 25, pp. 59–68, 1995.

[3] M. Impett, M. S. Corson, and V. Park, "A receiver-oriented approach to reliable broadcast ad hoc networks," *Proc. of Wireless Communications and Networking Conference (WCNC'2000)*, vol. 1, pp. 117–122, Sep. 2000.

[4] J. Wu and F. Dai, "A generic distributed broadcast scheme in ad hoc wireless networks," *Proc. of ICDCS'2003*, pp. 460–468, May 2003.

[5] A. Qayyum, L. Viennot, and A. Laouiti, "Multipoint relaying for flooding broadcast message in mobile wireless networks," *Proc. of 35th Hawaii Int'l Conf. on System Sciences (HICSS-35)*, pp. 3898–3907, Jan. 2002.

[6] P. Jacquet, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, and L. Viennot, "Optimized link state routing protocol," Nov. 2002, draft-ietf-manet-olsr-07.txt.

[7] L. Lovasz, "On the ratio of optimal integral and fractional covers," *Discrete Mathematics*, vol. 13, pp. 383–390, 1975.

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

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

[10] W. Peng and X. Lu, "Efficient broadcast in mobile ad hoc networks using connected dominating sets," *Journal of Software*, vol. 12, no. 4, pp. 529–536, 2001.

[11] P. Bhagwat, P. Misra, and S. Tripathi, "Effect of topology on performance of reliable multicast communication," *Proc. of IEEE INFOCOM'94*, pp. 602–609, Jun. 1996.

[12] A. Erramilli and R. P. Singh, "A reliable and efficient multicast protocols for broadband broadcast networks," *Proc. of ACM SIGCOMM'88*, pp. 343–352, Aug. 1988.

[13] S. Floyd, V. Jacobson, S. McCanne, C. G. Liu, and L. Zhang, "A reliable multicast framework for light-weight sessions and application-level framing," *Proc. of ACM SIGCOMM'95*, pp. 342–356, Aug. 1995.

[14] J. C. Lin and S. Paul, "RMTP: A reliable multicast transport protocol," *Proc. of IEEE INFOCOM'96*, pp. 1414–1424, Apr. 1996.

[15] D. Towsley, J. Kurose, and S. Pingali, "A comparison of sender-initiated and receiver-initiated reliable multicast protocols," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 3, pp. 398–406, Apr. 1997.

[16] S.-T. Shue, Y. Tsai, and J. Chen, "A highly reliable broadcast scheme for IEEE 802.11 multi-hop ad hoc networks," *Proc. of 2002 IEEE Int'l Conf. on Communications (ICC'2002)*, vol. 1, pp. 610–615, Apr.-May 2002.

[17] P. M. Merlin and A. Segall, "A failsafe distributed routing protocol," *IEEE Trans. on Communications*, vol. 27, no. 9, pp. 1280–1288, 1979.

[18] J. J. Garcia-Luna-Aceves and Y. Zhang, "Reliable broadcasting in dynamic network," *Proc. of 1996 IEEE Int'l Conf. on Communications (ICC'96)*, vol. 3, pp. 1630–1634, Jun. 1996.

[19] S. Alagar, S. Venkatesan, and J. Cleveland, "Reliable broadcast in mobile wireless networks," *Proc. of Military Communications Conference (MILCOM'95)*, pp. 236–240, Nov. 1995.

[20] E. Pagani and G. P. Rossi, "Providing reliable and fault tolerant broadcast delivery in mobile ad hoc networks," *Mobile Networks and Applications*, vol. 4, pp. 175–192, 1999.

[21] J. Wu and F. Dai, "Mobility management and its applications in efficient broadcasting in mobile ad hoc networks," *accepted to appear in Proc. of INFOCOM 2004*, Mar. 2004.