Robcast: A Singlehop Reliable Broadcast Protocol for Wireless Sensor Networks

Murat Demirbas Srivats Balachandran
Department of Computer Science and Engineering
University at Buffalo, SUNY, Buffalo, NY, 14260
{ demirbas | sb83}@buffalo.edu

Abstract

Empirical studies show that wireless sensor networks (WSNs) are extremely prone to the hidden terminal problem. As much as 50% of packet losses have been blamed to the hidden terminal problem under bursty communication. Current protocols use RTS-CTS handshakes to avoid collisions and alleviate the hidden terminal problem for unicast communication. However, it is not possible to directly or efficiently generalize this approach to broadcast communication. Motivated by this problem, we propose Robcast, a round-based self-stabilizing protocol for reliably broadcasting data using receiver-side collision detection feedback. The singlehop level reliability of Robcast can form a building block on which future applications and protocols can be designed for WSNs.

1 Introduction

Recent advances in low-power wireless radios and MEMS technology have made it feasible to deploy large-scale wireless sensor networks (WSNs) for real-world application scenarios. Some of these examples include monitoring the nesting behavior of endangered birds in a remote island [32], monitoring the temperature and humidity in vineyards [6], sniper localization [29], and classification and tracking of trespassers [3, 4].

Such real-world experiments have provided us with a better understanding of the workings of WSNs. These studies, especially [40, 43], show the detrimental effects of hidden node problem on WSNs. In the hidden node problem, simultaneous transmissions from two transmitters that are outside each other's carrier sensing range collide at a receiver node, and lead to message losses. In particular, more than 50% message loss has been reported due to hidden node problem under bursty traffic loads [42]. Current solutions for avoiding the hidden node problem revolve around handshakes with RTS/CTS, providing a partial solution for unicast transmissions.

Although support for reliable unicast using RTS/CTS handshake has existed traditionally in 802.11 [1] and in WSN MAC layer protocols [41], there has not been any support for reliable broadcasting. Popular WSN MAC protocols [25, 41] provide best effort delivery of broadcast packets. However, reliable communication over a singlehop is an essential component of WSNs, and is identified as the narrow-waist of a unified WSN architecture [11]. Reliable packet delivery is important for tracking applications for classification of the type of the tresspasser based on the number of detections it triggers. Reliable broadcast is also essential in sensor/actuator networks where all nodes need to consistently reach a consensus. For example, sensor/actuator devices coordinating regulator valves in a factory floor may need to take consistent decisions to prevent a malfunction.

Simple extensions of popular unicast protocols using RTS/CTS or acknowledgments [31, 33–35] run into problems when applied to broadcast messages. Implosions of the CTS and ACK packets at the source decreases efficiency. Performing handshakes on a node-by-node basis similar to unicast can guarantee reliability. However the high communication overhead and difficulty in maintaining neighbor lists makes such protocols unattractive. Solutions that use separate control channels/frequencies like BTMA [36] are not suited to WSNs. Many TDMA based approaches [7, 20, 27, 38, 39] have been proposed for providing reliable transmissions, but at the expense of some wasted bandwidth and energy.

Contributions Our contribution is a singlehop reliable broadcast protocol, Robcast, for low power ad hoc WSNs.

In Robcast, receiver-side collision detection (RCD) techniques [10] are used for reducing data loss. All nodes are synchronized to maintain globally shared rounds with each round having three phases: RTS, NCTS and Data phases. If a node j has data to transmit, a request for transmission (RTS) is sent by j during the RTS phase. Neighboring nodes respond to an invalid RTS or collided RTS's by transmitting a not-clear-to-send message (NCTS) during the NCTS phase. The neighbors are able to sense collisions at the RTS phase using RCD techniques. The node j backs off from transmitting the data for this round if it either receives a NCTS message or detects a collision in the NCTS phase. (Again, the collisions of multiple NCTS messages are detectable at j due to RCD techniques.) This scheme avoids potential collision of data during the Data phase, and ensures reliable delivery of the payload within single-hop distance of the transmitting node.

Robcast is self-stabilizing [13, 14] in that starting from arbitrarily corrupted states of the protocol it can recover to satisfy its specification. Robcast also creates controlled

sleep periods (by turning the radio off when possible) in order to improve power efficiency of the system.

In order to compare the performance of Robcast with other Broadcast protocols, we performed simulations in Prowler [30], a MATLAB-based, event-driven simulator for WSNs. Prowler simulates the radio transmission/propagation/reception delays of Mica2 motes [22], including collisions in WSNs realistically.

With the increasing interest within the research community in WSNs and numerous protocols being designed, many papers have looked at the current issues in the field [2, 11]. One of the most important problems at hand being the lack of a system architecture for WSNs, recently [11] presented such an architecture, the SensorNet Protocol, and has identified single-hop broadcast communication as the "narrow-waist" for WSN's—analogous to IP [26] for Internet protocols. We believe that a reliable broadcast service, such as Robcast, can serve as a building block at the single-hop and provide a platform for the development of future protocols and applications.

Outline of the paper. After the related work section, we introduce preliminaries in Section 3. We present our Robcast protocol and a formal proof of correctness in Sections 4.1 and 4.2 respectively. Self-stabilization and extensions to the protocol are discussed in Sections 4.3 and 4.4. Section 5 presents simulation results that compare performance of Robcast with other broadcast protocols. Finally, concluding remarks are in Section 6. For reasons of space, we relegate the proofs to the full paper [5].

2 Related Work

Tang and Gerla [33] proposed a simple extension to IEEE802.11 to support broadcast. The extension incorporates RTS/CTS control frames similar to unicast schemes. When a $source$ broadcasts RTS, the neighbors not in YIELD state reply with a CTS. The source node expects a CTS from any of its neighbors before transmitting DATA. Nodes not involved in the broadcast set their state to YIELD if they receive a CTS. Similarly, "Robust Broadcast" protocol [37] choses a neighboring node as a virtual collision detector and performs a RTS-CTS handshake to take control of the channel before a broadcast. However, none of these protocols guarantee that collisions are absent at other nodes. The hidden node problem still remains and may affect some of the neighboring nodes.

BSMA [34] extends [33] by incorporating negative acknowledgments (NAK). If neighbors do not receive DATA after transmitting a CTS, they transmit a NAK. The $source$ backs off upon receiving a NAK and retransmits DATA. However, these protocols [33, 34] require direct sequence capture ability which enables the radio to lock to a sufficiently strong signal in the presence of interfering signals.

BMW [35] provides reliable broadcast using RTS-CTS-DATA-ACK handshakes with each neighbor individually in round robin while reverting to IEEE802.11 under high channel contention. BMMM [31] improves BMW by broadcasting the DATA packet only once. The $source$ reserves the channel using a RTS-CTS handshake with each of its neighbors, broadcasts the DATA and requests acknowledgments from each neighbor individually. BMW and BMMM thus deliver the message eventually but with individual transactions with its neighbors. The disadvantages of these schemes are the high latency and control overheads.

BTMA [36] prevents collisions by maintaining a separate radio channel for control information. A busy tone is transmitted by the receiver if currently receiving data. A $source$ transmits data only if the control channel is idle, thus avoiding collisions and hidden terminals. However, resource limitations of WSNs make the implementation of a separate radio/frequency difficult to achieve. Recently, Busy Elimination Multiple Access (BEMA) protocol [12] adopted BTMA for WSNs by using time synchronized rounds across all nodes (similar to the rounds in Robcast) to allocate a control channel in the time domain instead of in the frequency domain. The disadvantages of BEMA compared to Robcast are its need for finer-grain RCD for performing busy elimination, its vulnerability to certain obstacle arrangements, and its high energy consumption due to the large number of nodes transmitting busy packets in the control phase. We provide detailed comparison of the performance of BEMA and Robcast in Section 5.

TDMA protocols provide collision free medium access. However such a system requires efficient time synchronization for the entire network. Changes in the network topology requires a modification in the schedule or slot allocation. Finally, static allocation of slots can leave many slots unused reducing the throughput of the network. Many protocols have been proposed to improve pure TDMA [7,8,17,20,23,27,38,39]. Robcast grants on-demand access to the channel rather than assigning fixed slots as in TDMA based approaches. Hence, Robcast avoids the overheads of allocating and maintaining a time slot for each node in the network.

3 Preliminaries

Notation & terminology. A *program* consists of a set of variables and actions at each node. We use the notation $v_i.x$ to denote a program variable x residing at the node v_i. The set of all variables and their values at a given time defines the state of a program. Each action has the form:

$$< \textit{guard} > \longrightarrow < \textit{statement} >$$

A $guard$ is a boolean expression over variables. An action whose guard evaluates to TRUE in a particular state is said to be *enabled* for that state and is executed. An assignment statement updates one or more variables.

When a node uses its radio to transmit a message, the message is broadcast over the channel. Hence, we use the terms transmit and broadcast interchangeably and denote a message broadcast statement as $bcast(msg)$. Similarly, successful reception of a message at a node is denoted by the statement $receive(msg)$. The receipt of a collision is represented by the statement $receive(\pm)$.

Network model. We consider a network that is represented by a graph $G(V, E)$, where V and E are the set of all nodes and links in the network respectively. The network consists of N stationary nodes identified by v_1, v_2 ..., v_N. The set of all singlehop neighbors of v_i is represented as $Nbr(v_i)$. Any node v_k is said to be a singlehop neighbor of a node v_i iff v_k can establish bi-directional links directly with v_i. Note that $v_k \notin Nbr(v_k)$.

Each node possesses an omni directional radio that is half duplex. Its transmit power is maintained as constant at all times. The radio is capable of performing carrier sensing to detect channel activity. These characteristics are satisfied by the commonly used Chipcon radios [22].

We assume all traffic consists of broadcast messages, and discuss in Section 4.4, an extension to ensure the reliable delivery of unicast messages.

Fault model. We consider a system where faults can occur arbitrarily. Nodes may fail, stop and crash - corrupting the state of a node. Arrival of new nodes or changes to the topology are considered as transient faults. Moreover, the channel might corrupt messages due to collision, fading or interference. With such a fault model, we say that a program is *self-stabilizing* iff after faults stop occurring, the program eventually recovers from an arbitrary state to a state where its specification is satisfied [13, 14].

3.1 Collision detection

Since each message is addressed to all nodes in the neighborhood, we treat any loss of data, due to bit errors or collision of transmitted messages, as a collision. Though protocols presented in [8, 23, 24] utilize the information of detected collisions in simulations, they do not mention techniques for detecting collisions.

Carrier sensing, physical and virtual, has been widely employed in wireless devices to avoid collisions in a best-effort manner. Traditionally in MAC layers like IEEE 802.11, IEEE 802.15.4, and most WSN MAC protocols, carrier sensing has been used primarily at the transmitters: If a node wants to transmit $DATA$, it first senses the medium for any activity in the channel, and begins transmission only if there is no activity. This technique however is not very effective, as the collision occurs at the receiver and the physical carrier sensing at the transmitter does not realize the possible collisions at the receiver's radio. Hence we are motivated to design Robcast to sense the medium at the receiver and inform the transmitter of possible collisions using separate mechanisms.

Using physical carrier sensing, nodes can differentiate between genuine activity, such as a message or collision, and noise based on the variance in the channel energy. When there is genuine activity on the channel, there is a fairly constant channel energy which stays above the noise floor. Random noise exhibits significant variance in channel energy which can be identified by occasional pits below the noise floor. Further, we can identify collisions if a node in the idle state (when it is not transmitting or receiving a message) detects, using its carrier sensing mechanism, a genuine activity on the medium. In our preliminary experiments with the Mica2 mote platform [22], our sensing mechanism searches for the pits in the idle state and detects genuine activity in the radio if a pit is not found for a long period. We find that our receiver side carrier-sensing based collision detection (RCD) performs well and detects more than 90% of the collisions accurately.

3.2 Synchronization

Various protocols [15, 16, 21] have been developed for synchronization in WSNs. Some protocols like SMAC [41] utilize a SYNC packet as part of their protocol to synchronize the individual nodes' sleep schedules. Real world experiments [29] have proved that it is possible to obtain a fine level of time synchronization using such protocols.

However, round synchronization does not require all the nodes to have a global view of time. It has been noted in [9] that round synchronization can be achieved with gradient synchronization instead of a global synchronization. Robcast, could use such a round level synchronization. Given any two nodes v_i and v_j, both nodes are assumed to have the same global view of rounds i.e., rounds start (and end) at the same time on all nodes as illustrated in Figure 1.

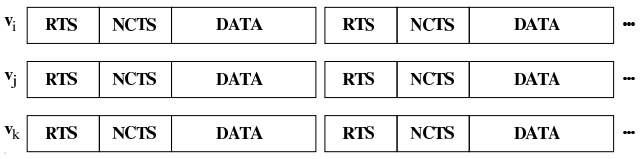

Figure 1. Synchronized rounds in Robcast

4 Protocol

In this section, we present Robcast and provide a formal proof of correctness, showing that Robcast eliminates the hidden node problem. We also prove that Robcast is self-stabilizing in the face of arbitrary state corruptions, and discuss extensions to Robcast for achieving energy-efficiency and handling unicast traffic.

4.1 Protocol description

Each node v_i maintains a single variable, $state$. $v_i.state$ has a domain of {*idle, candidate, transmit, veto*}. As a shorthand, we use $v_i.x$ to denote $v_i.state = x$.

$v_i.candidate$ means v_i wants to transmit a message, and, $v_i.transmit$ means v_i has exclusive access to the channel and it will be transmitting the rest of its packets in the DATA phase of consecutive rounds. $v_i.veto$ implies that there exists multiple *candidates* in $Nbr(v_i)$, and v_i will veto the candidates from becoming transmitters. If none of the above holds for v_i, $v_i.idle$ is true by default. Initially for all v_i, $v_i.state = idle$.

The variable "phase" is an external variable (provided by a round synchronization service), notifying v_i of which phase of the round, RTS, NCTS or DATA, v_i is in. All the nodes have a consistent view of the phase.

As seen in Figure 2, Robcast consists of six actions.

Action 1 is enabled in the RTS phase when a node has data to send and needs access to the channel. This happens either when an $idle$ node has new data to be transmitted, or when a node is in the $transmit$ state during a multi-part message's transmission. In case the node is transmitting the first message, i.e. $state := idle$, the node transits to $candidate$ state.

Action 2 is enabled in the RTS phase when a node is $idle$ and is not transmitting data in this round. While listening to the channel for activity any collision detected indicates collision of RTS messages. In this case, the node detects the existence of multiple transmitters in its single-hop neighborhood, and goes to $veto$ state to block a $candidate$ from going to $transmit$ state. If an RTS message is successfully received, the number of packets of the data message to be received as part of this transmission is stored in $data\_to\_receive$.

Action 3 is enabled in the $NCTS$ phase when a node is in the $veto$ state. Upon execution, the node broadcasts a veto message in the format of a $NCTS$ message[1].

Action 4 is enabled in the NCTS phase for a node v_i in the $candidate$ state. If v_i receives an $NCTS$ message or detects a collision, v_j backs off from the transmission and transits to $idle$ state.

Action 5 is enabled in the DATA phase when a node v_i's state is $candidate$ or $transmit$. v_i's state is set to $transmit$ and the $DATA$ message is broadcast. If the $data\_to\_send$ field of the message indicates that all packets part of this message have been transmitted, v_i transits to $idle$ state.

Action 6 is enabled in the DATA phase when a node v_i's state is $idle$. If $data\_to\_receive > 0$ from Action 2, the node expects packets to arrive and receives the part of the message that has been transmitted. However, in case a $candidate$ receives an $NCTS$ during the $NCTS$ phase and backs off via Action 4, a timeout occurs in Action 6 because of the absence of a transmitter. In this case the data is not received by the node and its state is set to $idle$.

[1]Switching from transmission to listening is on the order of microseconds, hence the feasibility of this scheme [25].

(1) phase=$RTS \wedge (v_i.data\_to\_send > 0) \wedge (v_i.data\_to\_receive == 0)$
$\longrightarrow$ bcast(RTS)
if ($v_i.idle$)
then $v_i.state := candidate$

[]

(2) phase=$RTS \wedge v_i.idle$
$\wedge (v_i.data\_to\_send == 0 \vee v_i.back\_off == TRUE)$
$\longrightarrow$ if (receive($\pm$))
$v_i.state := veto$
else if (receive(msg))
$v_i.data\_to\_receive := msg.data\_to\_receive$

[]

(3) phase=$NCTS \wedge v_i.veto$
$\longrightarrow$ bcast($NCTS$)
$v_i.state := idle$

[]

(4) phase=$NCTS \wedge v_i.candidate \wedge receive(\pm \; or \; msg)$
$\longrightarrow$ $v_i.state := idle$

[]

(5) phase=$DATA \wedge (v_i.candidate \vee v_i.transmit)$
$\longrightarrow$ $v_i.state := transmit$
bcast(msg)
$v_i.data\_to\_send := v_i.data\_to\_send - 1$
if $v_i.data\_to\_send == 0$
then $v_i.status := idle$

[]

(6) phase=$DATA \wedge (v_i.idle \wedge v_i.data\_to\_receive > 0)$
$\longrightarrow$ receive(msg)
if (receive_timeout)
$v_i.data\_to\_receive := 0$
else
$v_i.data\_to\_receive := v_i.data\_to\_receive - 1$

Figure 2. Program actions for j.

4.2 Correctness proof

Lemma 4.1 states that if at the beginning of a round's RTS phase, for any node v_i there exists no node in its neighborhood v_j, such that $v_j.transmit$, then there can exist only one node access to the channel during the $DATA$ phase of that round.

Lemma 4.1. ***(Leader election)*** *If* $(\forall v_i : v_i \in G : (\forall v_j : v_j \in Nbr(v_i) : \neg v_j.transmit))$ [2] *in the beginning of a round, then* $(\forall v_j, v_k : v_j, v_k \in Nbr(v_i) : (v_j.transmit \wedge v_k.transmit) \implies v_j = v_k)$ *in the DATA phase of that*

round.

If the transmitting node has multiple packets to transmit as part of its message, it will retain access over the channel until all the packets are transmitted successfully. In order for this to happen reliably, no other $candidate$ must be allowed to go to $transmit$ state. Lemma 4.2 states that if at the beginning of a round, for any node v_i there exists a node $v_j \in Nbr(v_i)$, such that $v_j.transmit$, then there will exist only one node, v_j, with access to the channel during the $DATA$ phase of that round.

Lemma 4.2. *(**Leader preservation**) If* $(v_i : v_i \in G : (\exists v_j : v_j \in Nbr(v_i) : v_j.transmit))$ *in the beginning of a round, then* $(\forall v_k : v_k \in Nbr(v_i) : v_k.transmit \implies v_k = v_j)$ *in the DATA phase of the round.*

We now prove that in Robcast there exists no hidden node, that is, for any given node, there can exist at most one transmitter in its singlehop neighborhood. Theorem 4.3 states that if $I1$ holds, there can be at most one node within singlehop of a node v_i that has access to the channel in the $DATA$ phase of any round.

Theorem 4.3. *(**No hidden node**) Let* $I1$ *denote phase* = $DATA \wedge (\forall v_i : v_i \in G : (\forall v_j, v_k : v_j, v_k, \in Nbr(v_i) : v_j.transmit \wedge v_k.transmit \implies v_j = v_k))$. $I1$ *is an invariant of the Robcast protocol.*

For the protocol to reliably deliver data, along with the absence of collisions the intended recipients must be ready to receive data. Lemma 4.4 states that, it is always the case that if a node v_i is in $transmit$ state, then all nodes in $Nbr(v_i)$ will be $idle$ and hence receive data during the $DATA$ phase.

Lemma 4.4. *Let* $I2$ *denote phase* = $DATA \wedge (\forall v_i : v_i \in G : (\exists v_j : v_j \in Nbr(v_i) : v_j.transmit) \implies v_i.idle)$. $I2$ *is an invariant of the Robcast protocol.*

4.3 Self-stabilization

As we prove in Lemma 4.3 and Lemma 4.4, in the absence of faults, starting from initial states, $I1$ and $I2$ hold for Robcast and, hence, from Theorem 4.3 we conclude that Robcast eliminates the hidden node problem. However, due to faults, such as transient memory corruption, message loss, or changes in network topology, $I1$ and $I2$ can be violated. Here, we show that Robcast protocol is self-stabilizing, that is, starting from any arbitrary state, after the faults stop occurring (or no faults occur for a period sufficient enough for stabilization) Robcast starts satisfying its specification.

We prove Theorem 4.5 by showing that starting from arbitrary states Robcast converges to states where $I1$ and $I2$ are satisfied. More specifically, $I1$ and $I2$ are re-established within at most $max\_message\_length$ rounds: the maximum number of packets that a message can span. Once the invariant $I1$ and $I2$ are satisfied, Theorem 4.3 states that the hidden-node problem is eliminated in the $DATA$ phase of the subsequent rounds.

Theorem 4.5. *Robcast is self-stabilizing.*

Proof sketch. Our proof is by demonstrating a variant function g that always decreases outside the invariant states. We choose g as the "number of transmitters within singlehop of a node". □

4.4 Extensions

Here, we discuss some extensions that could be applied to the Robcast protocol.

When a node is listening to the channel, it spends as much energy as transmitting [25]. Therefore, it is important to reduce any idle listening in our MAC protocol. To this end, we can extend Robcast so that when a node, v_i, detects that it is not receiving any message transmission in the beginning of a DATA phase, it can turn off its radio, set a timer, and sleep for the rest of the $DATA$ phase. Later, upon expiration of its timer, v_i can wake up at the beginning of the RTS phase. Similarly, when a contending node v_i is deferred from access to the channel via Action 4, v_i can turn off its radio and sleep until the beginning of the next round's RTS phase. This allows the node to sleep during the $DATA$ phase - up to 90% of the round duration. This kind of energy savings is highly attractive for battery operated wireless sensor nodes. In future work, using PowerTOSSIM [28], we will quantify the energy-savings we achieve by eliminating idle-listening as mentioned.

Secondly, it is possible to extend Robcast to also support unicast messages with a simple extension as in [23]. To achieve this, we introduce a third type of packet - CTS message that can be transmitted during the $NCTS$ phase. In case of a successful receipt of a unicast message, a node replies with a CTS message if it is the intended receiver. Any possible interference caused by the transmitter in its neighborhood by this unicast transmission will be avoided by an $NCTS$ response from its neighbors. Thus, the receipt of a CTS at the transmitter gives the go ahead and the unicast message will be transmitted. If the transmitter does not receive an CTS message or receives a collision, it backs off due to the possibility of corrupting parallel transmissions. This scheme will allow the Robcast protocol to reliably deliver both unicast and broadcast messages.

[2] A formula $(op\ j : R.j : X.j)$ denotes the value obtained by performing the (commutative and associative) op on the $X.j$ values for all j that satisfy $R.j$. As special cases, where op is conjunction, we write $(\forall j : R.j : X.j)$, and where op is disjunction, we write $(\exists j : R.j : X.j)$. Thus, $(\forall j : R.j : X.j)$ may be read as "if $R.j$ is true then so is $X.j$", and $(\exists j : R.j : X.j)$ may be read as "there exists an j such that both $R.j$ and $X.j$ are true". Where $R.j$ is true, we omit $R.j$.

5 Simulations

Our simulations were carried out in Prowler [30] with the network laid out as a 5-by-5 grid of nodes, a total of 25 nodes. The traffic load was varied by varying the number of nodes requesting to transmit data.

Simulations were conducted using realistic Mica radio instead of the idealized model. In the idealized model, transmissions are maintained free from external influences like noise and multi-path fading. A message in this environment can be lost only when it collides with another message. However, in the realistic radio model, transmissions are subject to Rician fading and multipath interference effects as well as collisions along with a 5% error probability for each message reception.

We measure the time during which the network is active with "*settling time*" which is defined as the duration between the last time a packet is received and the first time a packet is sent in the network. With this definition, we compare the following metrics for various protocols.

Goodput : Calculated as *number of data bits received/settling time*, goodput presents the effective usage of the bandwidth for data bits.

Total Loss of Packets : In our attempt to understand the performance of the protocol we measure the number of data bits lost in the transmission as the number of unique data packets received for each data packet transmitted.

Table 1 presents the message format of the protocols compared. The data payload in all the protocols is 960 bits long. In BEMA the CONTROL phase is for 100 bit-time. The RTS/CTS/NCTS and all other control messages in Robcast, BSMA and BMMM are 48 bits long, as implemented in SMAC [41].

	Control packets	Data packets
CSMA	0 bits	960 bits
BSMA	48 bits	960 bits
BMMM	48 bits	960 bits
BEMA	100 bits	960 bits
Robcast	48 bits	960 bits

Table 1. Message formats of the protocols.

The reported data for our experiments are values averaged over 10 independent runs for each configuration.

Goodput. CSMA does not attempt to provide any reliability, and transmits any data as soon as it can without eliminating the hidden node problem. Hence, it offers the highest goodput while suffering in reliability as seen in Figure 4. The goodput of BSMA linearly decreases with respect to the number of transmitters, due to the corresponding linear increase in the number of collisions. BMMM' guarantees reliable delivery of data to all neighbors (it ensures virtually no collisions), however, due to the individual handshakes, a large synchronization overhead and latency is incurred

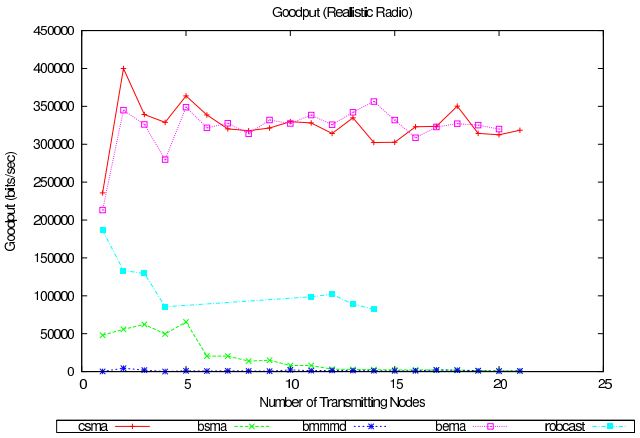

Figure 3. Goodput in realistic radio model.

which results in the low goodput. BEMA has a high goodput among the protocols. BEMA attempts to elect leaders during each of its rounds to reliably deliver data while decreasing collisions. It also scales well with a increase in the number of transmitters. Robcast however, pays the price for back-offs in a round based system and has a high settling time —hence the lower goodput compared to BEMA. Though BEMA and Robcast are round based, BEMA elects a leader always, hence providing better goodput.

Total loss of packets. Figure 4 shows the number of packets completely lost after transmission for the protocols. Since CSMA employs no special control messages to prevent collisions, the number of collisions is highest for CSMA due to hidden node problem and the reliability decreases with respect to the number of transmitters. BSMA's reliability hovers around 0.80 when the number of transmitters in the network increases beyond 40% of the network size, i.e. 10 transmitters. BMMM' suffers because, a transmitter that transmits data to its neighbor after a RTS-CTS handshake requests an acknowledgment. However, if the request for acknowledgment (RAK) or acknowledgment (ACK) is corrupted due to the realistic radio, the bits transmitted during the entire handshake, data transmission is considered a lost effort. BEMA shows very few data collisions and offers high reliability, which is by and large constant with respect to the number of transmitters. The data collisions that arise could be due to selection of the same contention length or *unidirectionality* in some links or non-deterministic interference among nodes. Robcast, however, offers the best reliability in comparison. This is because, Robcast transmits a data packet only if there is no contention in the channel for that round.

Radio's power consumption. Figure 5 shows the power consumption of the radio. Robcast reduces the number of collisions and hence reduces the number of repeated transmissions leading to fewer messages transmitted and received as compared to the other protocols. Also, the power consumption of BEMA is the highest due to the large num-

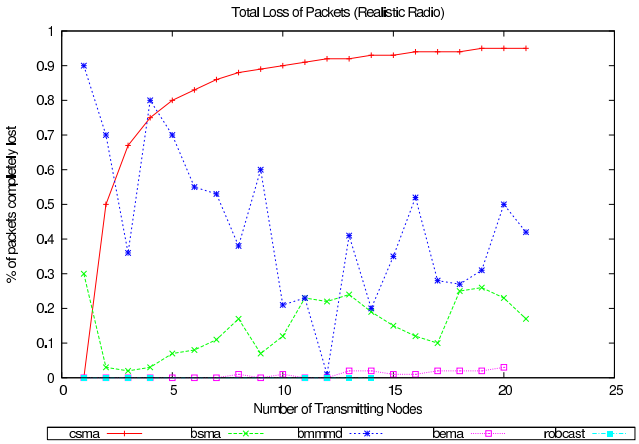

Figure 4. Total loss of packets in realistic radio model.

ber of BUSY packets required during the CONTROL phase.

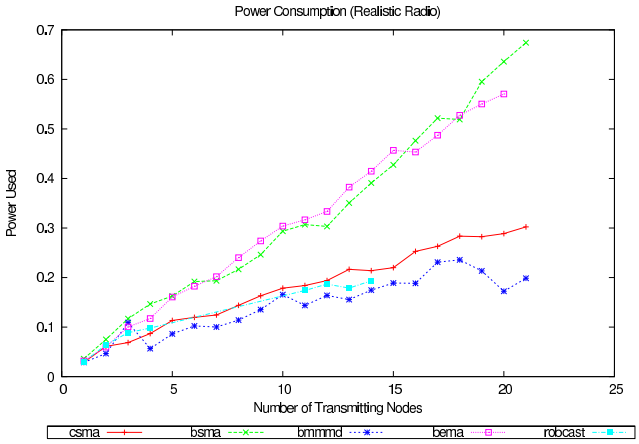

Figure 5. Radio's power consumption in realistic radio model.

5.1 Discussion

As the graphs demonstrate Robcast and BEMA perform better than the other protocols when both goodput and reliability need to be achieved. When compared to BEMA, Robcast performs better as it eliminates the hidden node problem completely: Robcast is not susceptible to the obstruction problem as depicted in Figure 6, a variant of the hidden node problem, as it is based on a receiver side collision detection mechanism. Also Robcast is more energy-efficient than BEMA as shown in Figure 5.

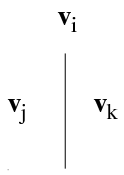

Figure 6. Obstacle arrangement where Robcast still solves the hidden node problem.

Unlike protocols like [24], Robcast does not transmit data if the virtual carrier sensing fails. So the transmission semantics is "**All** or **none**" like LBP [19] and BMMM [31]. This results in the loss of only the control packets and not data packets, hence resulting in a better goodput. However Robcast incurs the overhead of reserving the channel over the entire neighborhood of the transmitter. This may result in poor efficiency if a lot of unicast messages are transmitted - a price for reliability.

6 Concluding Remarks

We presented a self-stabilizing MAC protocol, Robcast, that solves the singlehop reliable broadcast problem. Robcast provides on-demand access to the channel using a RTS-NCTS handshake enabled by our RCD techniques. By avoiding collisions during the DATA phase and eliminating the hidden-terminal problem, Robcast provides a useful building block for applications with reliability requirements. Simulations show that Robcast's overhead is small and has the least total packet loss among BEMA [12], BSMA [34], BMMM [31], and CSMA/CA [1].

In future work, we will implement Robcast in TinyOS [18] on top of BMAC [25]. We will further investigate the performance improvements Robcast provides for handling bursty traffic patterns in WSNs, We will also work on a light-weight ad hoc synchronization scheme for implementing rounds in Robcast.

References

[1] Wireless lan medium access control(mac) and physical layer (phy) specification. IEEE Std 802.11, 1999.

[2] M. Ali, U. Saif, A. Dunkels, T. Voigt, K. Romer, K. Langendoen, J. Polastre, and Z. A. Uzmi. Medium access control issues in sensor networks. *SIGCOMM Comput. Commun. Rev.*, 36(2):33–36, 2006.

[3] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y.-R. Choi, T. Herman, S. S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita. A line in the sand: A wireless sensor network for target detection, classification, and tracking. *Computer Networks (Elsevier)*, 46(5):605–634, 2004.

[4] A. Arora and et. al. Exscal: Elements of an extreme scale wireless sensor network. *11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, 2005.

[5] S. Balachandran. Robcast: A reliable mac layer protocol for broadcast in wireless sensor networks. Master's thesis, University at Buffalo, SUNY, 2006. Technical Report no: 2006-29.

[6] J. Burrell, T. Brooke, and R. Beckwith. Vineyard computing: Sensor networks in agricultural production. *IEEE Pervasive computing*, 3:38–45, 2003.

[7] C. Busch, M. Magdon-Ismail, F. Sivrikaya, and B. Yener. Contention-free mac protocols for wireless sensor networks. In *DISC*, pages 245–259, 2004.

[8] I. Chlamtac, A. Myers, V. Syrotiuk, and G. Zaruba. An adaptive medium access control (mac) protocol for reliable broadcast in wireless networks. In *IEEE International Conference on Communications*, 2000.

[9] G. Chockler, M. Demirbas, S. Gilbert, and C. Newport. A middleware framework for robust applications in wireless ad hoc networks. In *43rd Allerton Conf. on Communication, Control, and Computing*, 2005.

[10] G. Chockler, M. Demirbas, S. Gilbert, C. Newport, and T. Nolte. Consensus and collision detectors in wireless ad hoc networks. In *PODC*, pages 197–206, 2005.

[11] D. Culler, P. Dutta, C. T. Ee, R. Fonseca, J. Hui, P. Levis, J. Polastre, S. Shenker, I. Stoica, G. Tolle, and J. Zhao. Towards a sensor network architecture: Lowering the waistline. *The Tenth Workshop on Hot Topics in Operating Systems (HotOS X)*, 2005.

[12] M. Demirbas and M. Hussain. A mac layer protocol for priority-based reliable broadcast in wireless ad hoc networks. *BroadNets*, 2006.

[13] E. W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Communications of the ACM*, 17(11), 1974.

[14] S. Dolev. *Self-Stabilization*. MIT Press, 2000.

[15] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. *SIGOPS Oper. Syst. Rev.*, 36(SI):147–163, 2002.

[16] S. Ganeriwal, R. Kumar, and M. B. Srivastava. Timing-sync protocol for sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 138–149, New York, NY, USA, 2003. ACM Press.

[17] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan. Application specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Networking*, 2002.

[18] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for network sensors. *ASPLOS*, pages 93–104, 2000.

[19] D. Kimt, J. Park, and Y. Choir. An efficient reliable multicasting protocol for micro-cellular wireless networks.

[20] S. S. Kulkarni and M. Arumugam. Ss-tdma: A self-stabilizing mac for sensor networks. In *IEEE Press. To appear*, 2005.

[21] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi. The flooding time synchronization protocol. *SenSys*, 2004.

[22] Crossbow technology, Mica2 platform. `www.xbow.com/Products/Wireless_Sensor_Networks.htm`.

[23] A. D. Myers, G. V. Zoruba, and V. R. Syrotiuk. An adaptive generalized transmission protocol for ad hoc networks. *Mob. Netw. Appl.*, 7(6):493–502, 2002.

[24] S. Park and R. R. Palasdeokar. Reliable one-hop broadcasting (rob) in mobile ad hoc networks. In *PE-WASUN '05: Proceedings of the 2nd ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, pages 234–237, New York, NY, USA, 2005. ACM Press.

[25] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 95–107, 2004.

[26] J. Postel. Internet protocol DARPA internet program protocol specification. RFC 791, Internet Engineering Task Force (IETF), Sept. 1981.

[27] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves. Energy-efficient collision-free medium access control for wireless sensor networks. In *SenSys*, pages 181–192, 2003.

[28] V. Shnayder, M. Hempstead, B. Chen, G. W. Allen, and M. Welsh. Simulating the power consumption of large-scale sensor network applications. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 188–200, New York, NY, USA, 2004. ACM Press.

[29] G. Simon, M. Maroti, A. Ledeczi, B. Kusy, A. Nadas, G. Pap, J. Sallai, and K. Frampton. Sensor network-based countersniper system. *Sensys*, pages 1–12, 2004.

[30] G. Simon, P. Volgyesi, M. Maroti, and A. Ledeczi. Simulation-based optimization of communication protocols for large-scale wireless sensor networks. *IEEE Aerospace Conference*, pages 255–267, March 2003.

[31] M.-T. Sun, L. Huang, A. Arora, and T.-H. Lai. Reliable MAC layer multicast in IEEE 802.11 wireless networks. *Proc. ICPP*, pages 527–537, 2002.

[32] R. Szewczyk, A. Mainwaring, J. Polastre, and D. Culler. An analysis of a large scale habitat monitoring application. *In Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[33] K. Tang and M. Gerla. Mac layer broadcast support in 802.11 wireless networks. *Proc. IEEE MILCOM*, pages 544–548, 2000.

[34] K. Tang and M. Gerla. Random access mac for efficient broadcast support in ad hoc networks. *Proc. IEEE WCNC*, pages 454–459, 2000.

[35] K. Tang and M. Gerla. Mac reliable broadcast in ad hoc networks. *Proc. IEEE MILCOM*, pages 1008–1013, 2001.

[36] F. A. Tobagi and L. Kleinrock. Packet switching in radio channels: part II the hidden terminal problem in carrier sense multiple-access and the busy-tone solution. *IEEE trans. on commun.*, COM-23:1417–1433, 1975.

[37] J. Tourrilhes. Robust broadcast : Improving the reliability of broadcast transmissions on CSMA/CA. Technical Report HPL-98-38, Hewlett Packard Laboratories, Feb. 24 1998.

[38] T. van Dam and K. Langendoen. An adaptive energy-efficient mac protocol for wireless sensor networks. In *SenSys*, pages 171–180, 2003.

[39] L. F. W. van Hoesel and P. J. M. Havinga. A TDMA-based MAC protocol for WSNs. In *SenSys*, pages 303–304, 2004.

[40] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 14–27. ACM Press, 2003.

[41] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient mac protocol for wireless sensor networks. In *INFOCOMM*, pages 1567–1576, 2002.

[42] H. Zhang, A. Arora, Y. Choi, and M. G. Gouda. Reliable bursty convergecast in wireless sensor networks. In *MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pages 266–276, New York, NY, USA, 2005. ACM Press.

[43] J. Zhao and R. Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *Sensys*, pages 1–13, 2003.