BOSS: Beacon-less On Demand Strategy for Geographic Routing in Wireless Sensor Networks

Juan A. Sanchez, Rafael Marin-Perez and Pedro M. Ruiz
Dept. of Information and Communications Engineering
Fac. Informatica, Univ. of Murcia
Campus de Espinardo s/n
E-30100, Espinardo, Murcia
Email: {jlaguna, rafael81, pedrom}@dif.um.es

Abstract

Geographic Routing (GR) algorithms, require nodes to periodically transmit HELLO messages to allow neighbors know their positions (beaconing mechanism). To reduce the control overhead due to theses messages, beacon-less routing algorithms have recently been proposed.

However, existing beacon-less algorithms have not considered realistic physical layers. Therefore, those algorithms can not work properly in realistic scenarios.

In this paper we present a new beacon-less routing protocol called BOSS. Its design is based on the conclusions of our open-field experiments using Tmote-sky sensors. BOSS is adapted to error-prone networks and uses a new delay function to reduce collisions and duplicate messages produced during the phase of selecting the next forwarder. Through extensive simulations we compare BOSS with BLR and CBF and show that our scheme is able to achieve almost perfect packet delivery ratio (like BLR) while having a low bandwidth consumption (even lower than CBF).

1 Introduction

In the last few years, Wireless Sensor Networks (WSN) have become a hot topic not only for researchers but also for the industry. A WSN consists of a big set of lightweight and cheap devices with many integrated sensors and wireless communication interfaces.

Nodes use their wireless radio to communicate the information acquired with their sensors. When the destination is out of the radio range of source node, other nodes are used as relay stations.

Geographic Routing is one of the schemes which has gained most momentum in the recent years. In GR each node needs to know the position of its neighbors. To do that, they send periodic HELLO messages called beacons including the identifier of the sender and its position. These packets are not forwarded, therefore only one hop neighbors can receive them. The packets include the identifier of the sender and its position.

However, although GR in general is highly desirable, the beaconing mechanism has some issues, such as generating interferences with regular data transmission and consuming bandwidth and battery power. In particular in those sensors not taking part in any routing process, the energy and bandwidth consumption represents a total waste of resources. To overcome such issues several beacon-less routing protocols have been proposed for WSNs. The most representative examples are: IGF [2], GeRaF [23], CBF [8], and BLR [18]. Beacon-less routing protocols do not require the proactive transmission of control messages which saves network resources. In these algorithms, the next forwarder can be reactively selected using two different approaches: a reactive and distributed scheme where the neighbors collaboratively decide which one must be the next forwarder or in a centralized way at each step. That is, the node currently holding the packet determines the next forwarder among its neighbors.

Nevertheless most of the beacon-less protocols reported in the literature assume a perfect wireless channel and do not consider the problems of losses and interferences created by the transmission of messages during the next forwarder selection process. As it has been shown recently ([20, 19]), there are huge differences between a real link and the well known unit disk model widely used in the literature. Considering a realistic link layer has big implications on the performance and validity of the proposed algorithms.

In this paper, we propose BOSS, the Beacon-less On Demand Strategy for Geographic Routing in Wireless Sensor Networks. Its design takes into account the losses and collisions typical of radio communications. Concretely, we have

made a practical study to determine the impact of the packet size on the Packet Reception Ratio (PRR). As our experiments confirm, bigger packets are more likely to be lost. Thus, unlike previous schemes, BOSS starts sending out the data packet itself rather than a control message. By doing that, only the neighbors able to receive the data packet contend for becoming the next forwarder.

BOSS uses also a passive ACK scheme to reduce the control overhead. We also introduce the Discrete Dynamic Forwarding Delay (DDFD) a new timer-based contention scheme to achieve a reduction of the collisions of the answers during the selection phase.

The remainder of this paper is organized as follows. In section 2, an overview of existing geographic routing protocols is given. The routing algorithm BOSS is described in section 3. In section 3.1, some variations and optimizations are discussed. BOSS is evaluated in section 4 through simulations and compared to existing solutions in the literature. Finally, section 5 concludes this paper.

2   Related work

Geographic Routing algorithms have become very popular in the field of WSNs because of their localized operation, reduced computation and storage requirements and, above all, because of their scalability with the number of nodes [11, 14].

The idea of geographic algorithms is to forward the packet in the direction of the destination. If the density of nodes is high enough, at each step the packet can be moved to a node closer to the destination than the previous one. This is called greedy routing, but it is possible to reach a node where no neighbor closer to the destination exist. In those cases, a recovery strategy must be used to surround the void area reached. This is called perimeter routing and it is based in the application of the right-hand rule [3] over a planarized vision of the underlying graph. The combination of a greedy approach and a recovery strategy is widely used in the literature. The most well-known algorithm doing that is Greedy Face Greedy Routing [4]

They usually employ a beaconing mechanism to know neighbors positions. Some of the beacons may be useless if data packets are not routed through some of the nodes. Moreover, recent studies have shown the limitations of these algorithms have when mobility is taken into account [13, 17].

To avoid these periodic transmissions, beacon-less protocols have been proposed in the literature. The general idea of beacon-less is to reactively discover the information about neighbors's positions necessary to select the next forwarder when routing data packets.

The Implicit Geographic Forwarding [2] (IGF) is a state-free routing protocol. IGF is a modification of the 802.11 DCF MAC protocol adding to the RTS/CTS handshaking a DATA/ACK interchange. Authors define the time each neighbor receiving the RTS from the forwarder node must wait before answering with a CTS. That time depends on the increase in distance towards the destination.

A similar scheme is also used in the Geographic Random Forwarding [23] (GeRAF) protocol. The main difference is that here two different frequencies are used by the collision avoidance MAC scheme defined. More recently a newer paper from the same authors[22] presented an extension of GeRAF. This variant called GeRaF-1R does collision resolution with a single radio frequency. Candidate relays schedule the forwarding of received messages by means of a probabilistic approach. Doing so they are able to reschedule the operation avoiding the possible collisions.

Lately, Heissenbüttel et al. proposed a new algorithm called Beacon-Less Routing [12] (BLR) which follows a different approach. In BLR, the selection of the next forwarder is totally distributed. That is, the current forwarder just broadcasts the data packet and the closest neighbor to the destination will resend the packet because its timer will expire the first. The rest of neighbors hearing the transmission cancel their timers. To avoid duplications due to neighbors not hearing transmissions, the authors determine that only the nodes located in a specific area, denominated Forwarding Area, can participate in the process. This area is relative to the forwarding node and can be of any shape provided that all nodes within the area can overhear each other. Additionally, a recovery strategy is defined. When the forwarder node does not hear any retransmission during a predefined time interval it broadcast again the packet and all the nodes answer with their positions. The forwarding node then, is able to locally build the planar graph and using the recovery scheme of GFG [4] determine the next forwarding node in perimeter mode.

The Contention-Based Forwarding algorithm [8] (CBF) mixes the centralized selection of the next forwarder based in a RTS/CTS/DATA process with a delimited Forwarding Area whose goal is reducing the duplicates in the same way as in BLR.

The Blind Geographic Routing [16] (BGR) protocol is very similar to BLR. The novelties introduced are a different recovery process and a strategy to avoid the problem of simultaneous forwarding. The results achieved are very similar to the ones of BLR.

Finally, Chawla et al. [6] proposed a new delay function to reduce the number of neighbors replying in perimeter mode. This function assigns shorter delays to neighbors closer to the forwarding node. Therefore, farther neighbors can cancel their responses after hearing the perimeter response of other neighbors, because they determine they are not part of planar graph of the forwarding node.
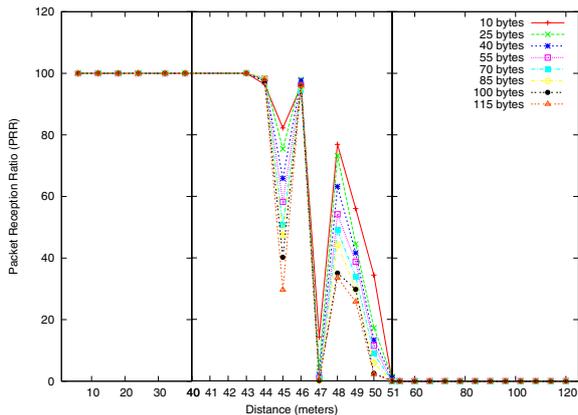
Figure 1. Percentage of messages successfully transmited at varying the distance for different packet sizes using the CC2420 radio interface.

3 Beacon-less Routing with BOSS

BOSS uses a three way handshake mechanism to select and forward messages in a similar way to the RTS/CTS scheme used in IEEE 802.11. Firstly the forwarding node broadcasts the data at maximum power to reach all its possible neighbors and waits for the first response of a neighbor. Then it confirms the selection with a final control packet. The key aspect of BOSS is the way of discovering the neighborhood. To do that, instead of using a control packet we use the data packet. Data messages are bigger than control messages, so that, they are more likely to be lost. Thus, the idea behind BOSS is to use the data packet being routed to discover the neighbors. In that way, neighbors not able to receive a packet of that size due to wireless errors will not take part in the selection process.

To confirm the importance of the packet size in the percentage of packets received and, therefore, support the goodness of our idea, we have made some experiments whose results are shown in the next section.

3.1 Analysis of the CC2420 Radio

The BOSS protocol is based on the assumption that the size of the packets has a direct relationship with the probability of error. Concretely, bigger packets have less probability of being received than smaller ones. If that is the case, discovering the neighborhood using one small control packet may cause to select a neighbor which may not be able to receive the bigger data packet. Our goal is to validate such assumption. Therefore, we run a set of real experiments in order to obtain the relation between Packet

Size and the Packet Reception Ratio.

We use are the new Tmote-sky [1] sensors based on the CC2420 radio chip from Chipcon [7]. The main elements integrated on the Tmote-sky, are the following: a MSP430 microcontroller (with 10kB RAM memory and 48kB Flash memory), a CC2420 radio chip (based in standard IEEE 802.15.4), an omnidirectional antenna (that provides a radio range of up to 50 meters indoor and 125 meters outdoor) and optionally, sensors of humidity, temperature, and light to monitor the environment.

The measurements have been obtained in an outdoor area of 100x100 meters. We use two sensors (source and receiver) placed at 0,5m above the ground and connected via USB to laptops. Each test consist in a 50 sequences of 100 packets for each packet size sent at maximum power (0 db) by the source node. The receiver reports to its connected laptop the following measured parameters: Radio Signal Strength Indicator (RSSI) which is a 8-bit value given by CC2420 chip that indicates the received signal strength in db. The Link Quality Indicator (LQI) which can be viewed as the chip error rate and it is calculated over 8 bits following the start frame delimiter (SFD). The LQI values are usually between 110 and 50, and correspond to maximum and minimum quality frames respectively. And finally, the Packet Size (PS) which is the sum of the payload size and header size of the received packet. The Packet Reception Ratio (PRR) is calculated in the laptop and it is defined as the ratio between the number of packets received and the total number of packets sent.

We have performed tests using 8 different payload sizes (10, 25, 40, 55, 70, 85, 100 and 115 bytes) and varying the distance between source and receiver (from 5m to 120m). As the results in the range between 44 and 51 meters have greater variability, the resolution in that range is greater.

Fig. 1 summarizes the results of our experiments. It shows the value of the PRR at varying the distance. Each curve represents the data obtained with each packet size tested. The packet size is the sum of the payload and the headers of the MAC and link layer.

The first conclusion that we can extract is that, as we anticipated, the greater the packet size, the lower the PRR. The second conclusion is that there is no direct relation between the distance and the PRR. As it can be seen, the results at some distances such as 48m and 45m, are better than at some other shortest distances, respectively 47 and 45. This coincides with some other empirical results [10][21][5] that show the irregularities of wireless communications in WSNs.

Moreover, as it can be seen, sensors placed farther than 51m are not able to communicate directly. Although the maximum theoretical range is 125m, placing the sensors near the floor causes too much reflections. In some other tests done with sensors placed at 2m above the floor, the
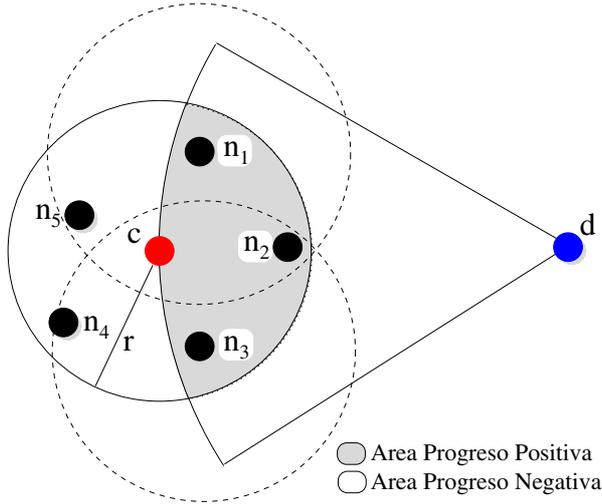
Figure 2. Node c currently holding the message toward d and its neighbors. Nodes n_1, n_2 and n_3 are in the Positive Progress Area. Nodes n_4 and n_5 are in the Negative Progress Area. Nodes n_1 and n_3 can not hear each other replies.

range grows up to 150m.

3.2 Forwarding in BOSS

BOSS uses four different types of messages: *DATA*, *RESPONSE*, *SELECTION* and ACK. In addition, given the forwarding node (i.e. the node currently holding the message) we define two relative areas around it. The Positive Progress Area (PPA) and the Negative Progress Area (NPA). PPA comprises each node whose position is closest to the destination than the forwarding node while the NPA comprises the rest of neighbors of the forwarding node. That is, those not providing progress towards the destination. Additionally, the *DATA*, *RESPONSE* and *SELECTION* messages include a bit in their headers to indicate the routing mode currently being used (**G**reedy or **P**erimeter). This bit is called the Routing Mode bit (RM). The RM bit is set to **G** mode by default. We now describe the detailed operation of the protocol, firstly in the greedy and then in perimeter mode.

In greedy mode, the forwarding node sends a broadcast with a *DATA* message and waits for responses for a predefined maximum time of T_{Max} seconds. The *DATA* message contains the original message, the position of the forwarding node and the final destination's position. Each neighbor receiving the message stores it and determines the relative area in which it is located (PPA or NPA). Finally,

instead of answering immediately, the node starts a timer whose value depends on its position. When the timer finishes the neighbor broadcast a *RESPONSE* message. The *RESPONSE* message contains the neighbor position.

Each neighbor in the PPA receiving a *RESPONSE* message from another neighbor in the same area, cancels its timer and deletes the stored message. The *RESPONSE* messages from NPA neighbors do not cancel any timer. Notice that it is possible that some neighbors do not receive this message because of their positions. Fig. 2 shows an example of this situation, in which, neighbors n_1 and n_2 cannot hear the *RESPONSE* messages from each other. The forwarding node stops its timer if a *RESPONSE* message from a PPA neighbor is received. It then broadcasts a *SELECTION* message. This message contains the identifier of the neighbor selected by the forwarding node to become the next forwarding node. That neighbor is the one whose *RESPONSE* message arrived in the first place to the forwarding node. More than one *RESPONSE* message from PPA neighbors might arrive to the forwarding node but only the first one is used. Each neighbor receiving the *SELECTION* message will immediately cancel its timer and delete the stored message except for the one selected by the forwarding node. The selected node knows it is the new forwarding node by examining the header of the *SELECTION* message. Finally, the new forwarding node starts again the protocol by broadcasting a new *DATA* message.

Some nodes may not have any neighbors providing advance toward the destination. In that case, a so-called void area, is found and the routing process can not continue in greedy mode. In Geographic Routing protocols there exist different strategies to surround these void areas but, usually, it is necessary to know the position of the neighbors in order to locally build a planar graph to determine the next perimeter forwarder. The most common algorithms to do that are the Relative Neighbor Graph [15] (RNG) and Gabriel Graph [9] (GG). As we have already commented, in BOSS, the *RESPONSE* messages from neighbors in the NPA do not cancel any timer. Thus, when the forwarding node does not have any neighbor providing advance towards the destination, all the *RESPONSE* messages from the other neighbors (those in the NPA area) are received and stored. When the timer expires, the forwarding node can thus build the planar graph using all the information gathered and then selects the next forwarding node using the desired recovery mechanism, in our case we use the one proposed in GFG [4].

In the case of perimeter routing the *SELECTION* message must include some extra information. Concretely the identifier of the forwarding node, its position, identifier of the next hop selected and the current perimeter information defined by GFG which consists of: the position of the node where perimeter routing started (L_p), the first edge (E_0) traversed on current face, and the (L_f) point, that is the cross

point between the $\overline{L_pD}$ line and the current face, being D the position of the destination node. Additionally, the RM bit must be set to **P** indicating perimeter routing.

When messages are being routed in perimeter routing, the behavior of neighbors is slightly different. To begin with, the forwarding node must include in the *DATA* message the L_p point. A neighbor receiving the *DATA* message must check if it is placed closer to the destination than the L_p point. If that is the case, it resumes to greedy mode. Thus, the RM bit of the *RESPONSE* message is set to **G**. Only in those cases, the *RESPONSE* message will be sent but, after the timer expires as in greedy routing. The forwarding node may stop its timer if it receives a *RESPONSE* message including a RM≡**G**. In that case, it changes to greedy mode and continues in that mode by sending the appropriate *SELECTION* message. If there is no neighbor closer to the destination than L_p then the forwarding node will wait up to T_{Max} seconds. Then, it selects the next forwarder to continue in perimeter mode and selects it by broadcasting the corresponding *SELECTION* message including also a RM≡**P**.

3.3 Design Decisions

The previous section describes the behavior of the BOSS algorithm but there are some important design decisions to consider during the implementation phase. As we have already commented, the neighbors receiving a *DATA* message store it and then wait for a timer to answer. The question is how long must that data packet be stored. Obviously, if the node receives a *RESPONSE* message, the data packet can be deleted. The same occurs after receiving a *SELECTION* message directed to another neighbor. But, as we are dealing with error-prone networks, both messages might be lost. In that case, the node will send its own *RESPONSE* message when its timer expires. The forwarding node receiving that late response must ignore it, but the neighbor is waiting for a *SELECTION* message to select it as next forwarder. This message will never arrive. Therefore, the data packet must be deleted after a maximum time.

Moreover, as messages can be lost, we need a confirmation of their reception. At least, the reception of the *SELECTION* message must be confirmed. To do that, we use two different techniques: a passive acknowledgement (PACK) and an active one (ACK). The use of the ACK introduces a new message in the process incrementing the protocol overhead in a 33%. Thus, we use the *DATA* message of the next forwarder as PACK to confirm the reception of the previous *SELECTION* message. The ACK is also needed when the message arrives to its destination because there is no more *DATA* forwarding. So, when a forwarding node does not receive a PACK or an ACK, it resends the *SELECTION* message up to a maximum number of 3 times. That means
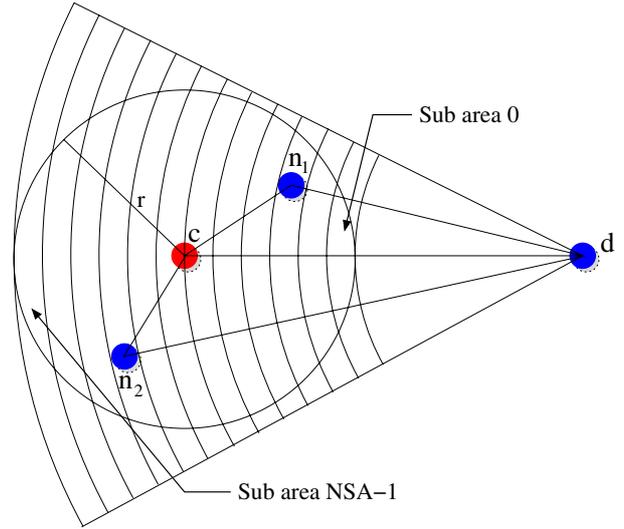

Figure 3. Division in areas for the DDFD.

that, neighbors selected as next forwarders must keep their data packets during at least, the 3 possible rounds of re-selections.

Finally, when the third re-selection fails the whole process is repeated. The forwarder node re-sends the *DATA* packet and the neighbors start again the contention process. This retransmission process can be tried up to 10 times. After that, the packet is dropped. Our experiments show that in BOSS the retransmissions are rarely used. As BOSS uses the data packet to determine which neighbors take part in the contention process, only those having strong links send their responses and can be selected. Therefore, as the *RESPONSE* and *SELECTION* messages are significantly smaller than the *DATA* packets, their probability of being transmitted successfully is higher. Thus, the probability using retransmissions is lower.

3.4 Discrete Dynamic Forwarding Delay (DDFD)

As we have already commented, all the neighbors wait for a period of time before answering the forwarding node. Moreover, the time to wait is related to their position. This behavior has two important goals: avoiding collisions and determining the forwarding strategy. Letting all the neighbors answer immediately increases exponentially the probability of colliding answers because the *DATA* message arrives almost at the same time to all of them. On the other hand, in BOSS the forwarding node selects as next forwarder the neighbor which replies first. Therefore, the forwarding strategy is clearly controlled by the way the timers work. Additionally, by forcing some neighbors to wait more

than others we can reduce the number of possible answers and thus, the bandwidth consumption. The key then is to design a function to determine the timeout value in such a way that the most promising neighbors answer in the first place.

To determine the forwarding strategy we, as some other protocols in the literature, define the progress of a candidate neighbor as a way to measure its goodness as next forwarder. In our case, we define it as follows:

$$P(n, d, c) = dist(c, d) - dist(n, d) \qquad (1)$$

where c and d are the forwarding and the destination nodes respectively and $dist(a, b)$ represents the Euclidean distance between the positions of the nodes a and b. Obviously, the maximum progress possible is equal to r, the radio range, and the minimum one is $-r$, achieved by the neighbors placed farther from the destination than c.

Our Discrete Dynamic Forwarding Delay (DDFD) function assigns smaller delay times to the neighbors providing the maximum progress toward the destination. To do that, instead of using directly the value of the progress function in each neighbor, we divide the neighborhood in sets of neighbors providing a similar progress (see Fig. 3). Concretely, we determine the Number of Sub Areas (NSA) in which we want to uniformly divide the whole coverage area and then, taking into account that the maximum difference in progress between two neighbors is $2r$, each neighbor determines in which Common Sub Area (CSA) it is placed. To do that it uses the following equation:

$$CSA = \left\lfloor NSA \times \frac{r - P(n, d, c)}{2r} \right\rfloor \qquad (2)$$

here, the value of CSA falls between 0 and $NSA - 1$ corresponding 0 to the area placed closest to the destination and $NSA - 1$ to the farthest one. Given the CSA, each neighbor computes its delay time according to the next equation:

$$T = \left(CSA \times \frac{T_{Max}}{NSA} \right) + random \left(\frac{T_{Max}}{NSA} \right) \qquad (3)$$

here, T_{Max} is a constant representing the maximum delay time that a forwarding node will wait for answers of its neighbors and, $random(x)$ a function obtaining a random value between 0 and x. By its construction, this function assigns half the total T_{Max} delay to neighbors in the PPA and half to the others. That allows the forwarding node to determine whether there are PPA neighbors or not because a PPA neighbor will always answer before $\frac{T_{Max}}{2}$ seconds. Additionally, the neighbors in the same CSA can wait different amount of times thanks to the random function. Neighbors from consecutive CSAs will never wait the same amount of time because the base time is determined by the CSA index.

Unlike other solutions proposed in the literature, our DDFD function combines a uniformly distributed value that depends on the progress with a random value generated so that the total delay does not mixes responses from nodes in different sub-areas. Thus, it manages to to reduce the number of responses and the probability of generating simultaneous responses from nodes which are in the same sub-area. The problem of other functions is that, usually, their value only depend on the progress to the destination. In this case, the forwarding node could have several neighbors providing a similar progress and, therefore, replying simultaneously causes collisions between the responses.

Finally, as our proposed function works in both routing modes (greedy and perimeter) it is not necessary to initiate a new process (broadcast, answering) for the cases when greedy routing fails. In those cases, no PPA neighbors will answer during the first $\frac{T_{Max}}{2}$ seconds but, after that, the answers from the NPA neighbors will arrive. The same situation occurs when routing in perimeter mode. If there exists a node closer to the destination than the position where perimeter routing started, that node answers before any of the NPA nodes. The *SELECTION* message issued by the forwarding node will then cancel all the responses from those neighbors.

4 Experimental Results

In this section, we provide a comparison between BOSS and two well known beacon-less algorithms: Beacon-Less Routing [18] (BLR) and Contention Based Forwarding [8] (CBF). BLR has different variants corresponding to different forwarding areas. Three possible areas can be used, namely Sector, Reuleaux triangle and Circle. We use Reuleaux triangle as forwarding area because, according to authors, Reuleaux triangle obtains better performance than Sector and Circle in terms of packet duplications and average progress in each hop. By CBF we refer to the version of the protocol using the active suppression method since this method is the one achieving the best performance.

Finally, in our simulations we consider a realistic MAC layer with collisions and interferences. That is, a message sent out by a node, might not be received by some nodes in its radio range. To do that, we use the results of our empirical experiments commented in section 3.1. We use them to build a function that computes the probability for that packet to be received given a distance and a packet size.

The simulation scenario is a $500 \times 500 m^2$ area in which a varying number of nodes (from 150 to 700 nodes) are deployed. This results in scenarios with different network densities (mean number of neighbors/node). We have considered 8 different mean densities to represent a wide spectrum of scenarios, from the sparse to the very dense ones. On the other hand, the source and destination nodes are al-

ways placed respectively a (0,0) and (500,500) coordinates. Thus, using a radio range of $r = 50$, the theoretical minimum number of hops is $\frac{500\sqrt{2}}{r} \simeq 14$. That is useful to determine the deviation from the better path of each protocol tested. For each scenario our results are the average over a total number of 200 simulation runs in order to achieve a sufficient small 95% confidence interval.

For BOSS we use: $T_{Max} = 600ms$ and $NSA = 10$. In CBF there are two different timers, one for greedy routing and another for perimeter routing. The two are set to $300ms$. The same occurs in the case of BLR, it is configured to use $300ms$ per timer. To make a fair comparison the three protocols are configured to behave in the same way when losses occur. The SELECT/ACK process can be repeated up to 3 times and the maximum number of retransmissions is set to 10. BLR can only do that in perimeter mode because in greedy mode there is no defined mechanism to do that. Finally, we have modified the CBF protocol to force the selected neighbor to use an ACK to inform the forwarder. That process is also retried up to 10 times. Otherwise, CBF provided a really poor performance in realistic error-prone WSNs.

4.1 Performance metrics

We considered the following metrics during the evaluation of performance of the algorithms:

- Total transmissions. This metric accounts for the total number of packets transmitted during the process of routing a message from the source to the destination. Includes also the messages not received and the transmissions made by duplicate packets due to errors in the protocols.

- Hop Count. This metric accounts for the number of point-to-point links in a transmission path. The number of hops is the average number of intermediate nodes between the source node and the destination node.

- Duplicates. This metric accounts for the number of packets received by the destination node for each one sent by the source in those cases where all the algorithms reach the destination.

- Packet Delivery Ratio (PDR). This metric shows the effectiveness of the protocols. It determines the percentage of packets that reach the destination node. This is an important performance metric in scenarios with realistic conditions.

- End-to-end delay. This metric accounts for the total time required for the first message sent by the source to make it to the destination.
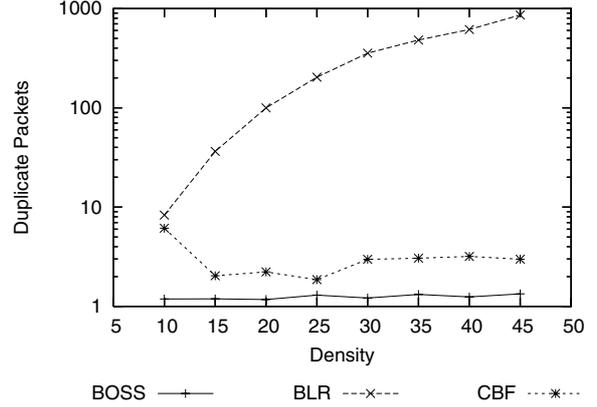

Figure 5. Duplicate packets

4.2 Analysis of Results

Fig. 4(a) shows the mean number of messages transmitted by each protocol at increasing densities. CBF and BOSS transmit much more messages for the scenarios with lower densities. Both algorithms reach their normal behavior when the density is above 20 (scenarios where routing is performed mostly in greedy mode). In all the tested scenarios BOSS transmits less messages than CBF. In most scenarios, BOSS only needs half the transmissions than CBF. The global performance of BLR is very bad in comparison with the other two algorithms. That proves the inefficacy of the BLR scheme, where neighbors select themselves as next forwarders.

Fig. 4(b) shows the number of transmissions made during perimeter routing. The scenarios with lower density force the three protocols tested to use the perimeter mode. That means, the number of messages is higher in those sparse scenarios. But unlike the other two algorithms, BLR continues using the perimeter mode when the density increases. The reason is that the higher the density, the higher the collisions produced by the neighbors forwarding the message because timers expire almost concurrently. Those collisions prevent the rest of neighbors to cancel their timers and the forwarder to hear the retransmissions from its neighbors. Thus, it changes to perimeter mode although it is not necessary and, in parallel, new branches are created.

In those sparse scenarios, the strategy of sending the data packet in the first place allows us to discard the neighbors with lossy links. In CBF, the probability of choosing a bad link is higher as the number of packets routed in perimeter mode confirms.

Fig. 5 shows the mean number of duplicate packets arriving to the destination for each one sent by the source. As it can be seen, in BLR the increase in the number of duplicates generated grows exponentially with the density. CBF has a
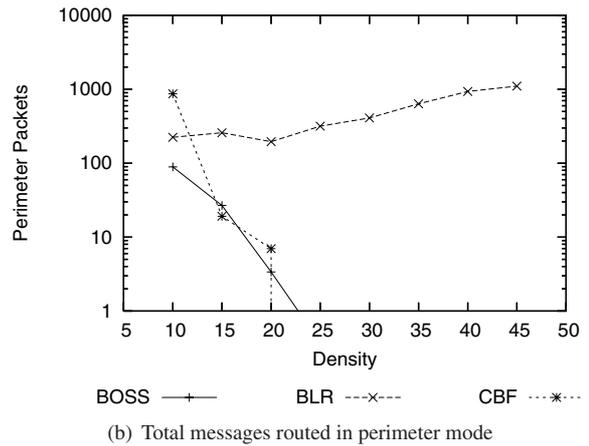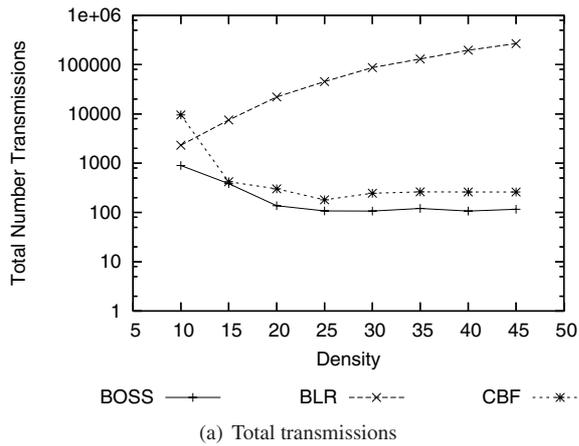
(a) Total transmissions

(b) Total messages routed in perimeter mode

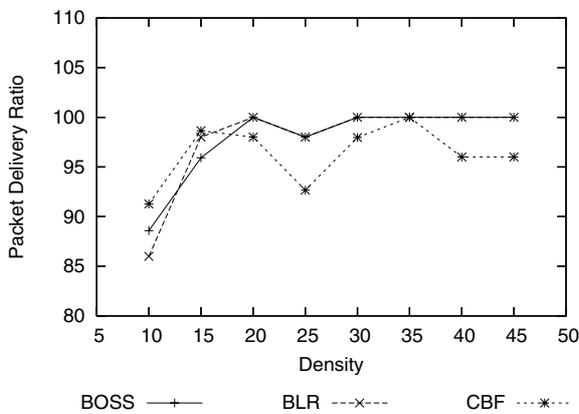**Figure 4. Comparing the three algorithms in terms of number o packets**

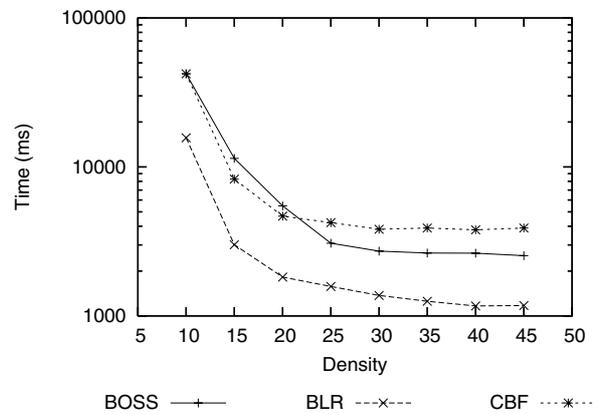**Figure 6. Packet Delivery Ratio**

Figure 7. End-to-end delay

mean of 3 duplicate packets while BOSS does not generate almost any duplicate.

CBF generates more duplicates than BOSS because of two main reasons. Firstly, the first response of a neighbor does not always arrive to every other candidate neighbor. Thus, more than one next forwarder can be chosen. Secondly, as CBF does not consider the quality of the links during the selection of the next forwarder, nodes with lossy links are likely to be chosen. Therefore, the ACK messages can be lost making the protocol to choose a different next forwarder although the first one is already forwarding the message. In BOSS, such nodes will never take part in the selection process.

Fig. 6 shows the Packet Delivery Ratio achieved by each algorithm at varying mean densities. The three algorithms have a high PDR because using 10 retries is enough in most of the situations. Nevertheless, BLR has the higher ratio due to the unacceptable number of duplicates generated. On

the other hand, CBF does not generate too much duplicates but its PDR is lower than the one of BOSS. The reason is that it chooses lossy links during the selection of the next forwarders. Finally, BOSS almost has perfect delivery for the scenarios with more than 20 neighbors per node. That is, when the packets are routed mostly in greedy mode. At the same time, as we have already seen, BOSS generates less duplicates than CBF and transmits thousands of times less messages than BLR.

Fig. 7 shows the End-to-end delay of the tested protocols. The graph shows that the density is strongly correlated with the end-to-end delay.The lower the density the greater the end-to-end delay. Obviously, routing in perimeter mode is the cause. Moreover, BLR generates lots of duplicates, but to measure the end to end delay we use the time of the first one arriving to the destination. That is the reason why BLR achieves the shorter End-to-end delay. Finally, BOSS, manages to outperform CBF. Here, the key
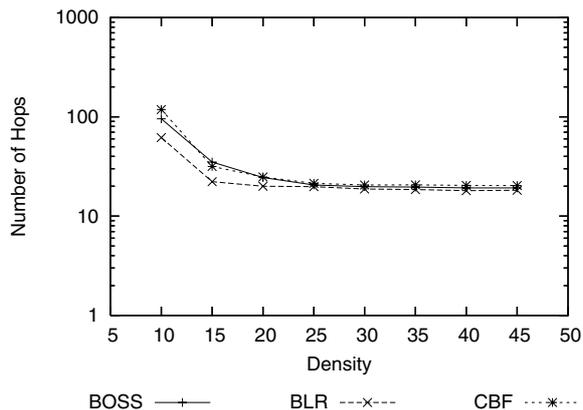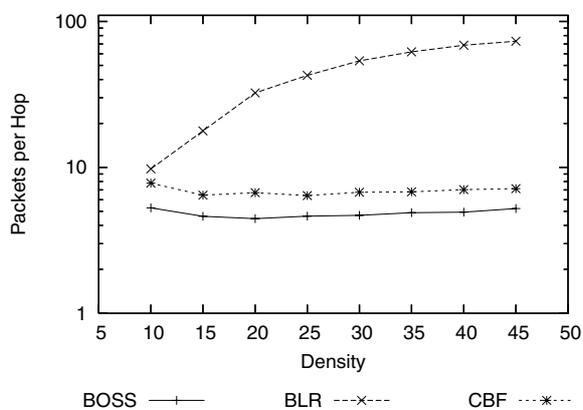
Figure 8. Mean Hop Count

Figure 9. Packets per hop

point is the DDFD combined with the strategy of sending the DATA packet first. BOSS makes less retransmissions because the first selection of next forwarders tends to chose reliable links.

Finally, lets examine more closely the two better protocols, CBF and BOSS. As the authors of CBF does not consider the problems caused by lossy links, they do not define any method to check the delivery has been successful. Thus, to allow CBF work in our realistic scenarios we have added it a ACK message used to confirm the reception of the last message of the protocol (the data sent from the forwarder to the next forwarder). Therefore, the mean number of messages per hop is going to be at least 4 while in BOSS, by using a PACK strategy, only in the worst cases is going to be necessary an ACK. Fig. 8 shows the mean hop count of each algorithm while Fig. 9 shows the mean number of packets per hop. As it can be seen, the algorithms achieve a mean hop count near to 20 which is closer to the theoretical limit of 14. Obviously, we are accounting the hop count of

the first message arriving the destination and ignoring the duplicates arriving later. On the other hand, we can see that the mean number of packets per hop of CBF is 7 and 5 for BOSS. In theory, CBF has only one more message per hop than BOSS but in practice, the number of responses and retries is being higher in CBF than in BOSS. That means, the delay function included in CBF has a worst performance than our DDFD, so that, the number of responses generated is higher.

5   Conclusions and Future Work

In this paper, we propose and evaluate BOSS, a new beacon-less routing protocol for wireless sensor networks. It is designed to deal with errors, losses and interferences common in wireless transmissions. To do that, several empirical experiments have been made to confirm the direct relationship between the packet size and the probability of reception. Taking that fact into account, in BOSS we use a three way handshake protocol to determine the neighbors and select the next forwarder at each step of the routing. In BOSS the data packet being forwarded is sent first. The goal is to reduce the set of candidates to next forwarder nodes only to those neighbors able to receive it. Moreover, we propose a new delay function that reduces the number of responses generated by candidate nodes.

Several simulations have been performed to evaluate the performance of BOSS against two well-known beacon-less protocols (CBF and BLR). BOSS succeeds in achieving a much lower number of transmissions (totals and per hop) while keeping the delivery ratio above the 90%.

For future work, we will analyze BOSS and other alternatives of beacon-less protocols in real scenarios. In addition, we also plan to work on different delay functions to deal with energy efficiency.

Acknowledgment

References

[1] Tmote Sky: Low power Wireless Sensor Module Datasheet, 2005.

[2] B. Blum, T. He, S. Son, and J. Stankovic. IGF: A state-free robust communication protocol for wireless sensor networks. Technical report, Department of Computer Science, University of Virginia, USA, 2003.

[3] J. Bondy and U. Murty. *Graph theory with applications*. Macmillan London, 1976.

[4] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with Guaranteed Delivery in Ad Hoc Wireless Networks. *Wireless Networks*, 7(6):609–616, 2001.

[5] A. Cerpa, J. L. Wong, L. Kuang, M. Potkonjak, and D. Estrin. Statistical model of lossy links in wireless sensor networks. In *Proc. 4th international symposium on Information processing in sensor networks, (IPSN '05)*, page 11, Piscataway, NJ, USA, 2005. IEEE Press.

[6] M. Chawla, N. Goel, K. Kalaichelvan, A. Nayak, and I. Stojmenovic. Beaconless Position Based Routing with Guaranteed Delivery for Wireless Ad-Hoc and Sensor Networks. In *Proc. 19th IFIP World Computer Congress*, Santiago de Chile, Chile, August 2006.

[7] CC2420 2.4 GHz IEEE 802.15.4 / Zigbee RF Transceiver. Chipcon Product data sheet.

[8] H. Füßler, J. Widmer, M. Käsemann, M. Mauve, and H. Hartenstein. Contention-Based Forwarding for Mobile Ad Hoc Networks. *Ad Hoc Networks*, 1(4):351 – 369, 2003.

[9] K. Gabriel and R. Sokal. A New Statistical Approach to Geographic Variation Analysis. *Systematic Zoology*, 18:259–278, 1969.

[10] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. Complex Behavior at Scale: An Experimental Study of Low-Power Wireless Sensor Networks. Csd-tr 02-0013, UCLA, February 2002.

[11] S. Giordano, I. Stojmenovic, and L. Blazevie. Position Based Routing Algorithms for Ad Hoc Networks: A Taxonomy. *Ad Hoc Wireless Networking*, pages 103–136, 2004.

[12] M. Heissenbüttel, T. Braun, T. Bernoulli, and M. Wächli. BLR: Beacon-Less Routing Algorithm for Mobile Ad-Hoc Networks. *Elsevier's Computer Communications Journal (ECC)*, 27(11):1076–1086, July 2004.

[13] M. Heissenbüttel, T. Braun, M. Wälchli, and T. Bernoulli. Evaluating of the limitations and alternatives in beaconing. *Ad Hoc Netw.*, 5(5):558–578, 2007.

[14] J. Li, J. Jannotti, D. S. J. D. Couto, D. R. Karger, and R. Morris. A Scalable Location Service for Geographic Ad Hoc Routing. In *Proc. 6th annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '00)*, pages 120–130, New York, NY, USA, 2000. ACM Press.

[15] G. Toussaint. The Relative Neighborhood Graph of a Finite Planar Set. *Pattern Recognition*, 12:261–268, 1980.

[16] M. W. V. Turau. BGR: Blind Geographic Routing for Sensor Networks. In *Proc. of the Third Workshop on Intelligent Solutions in Embedded Systems (WISES'05)*, Hamburg, Germany, May 2005.

[17] M. W. V. Turau. The Impact of Location Errors on Geographic Routing in Sensor Networks. In *Proc. of the Second International Conference on Wireless and Mobile Communications (ICWMC'06)*, Bucharest, Romania, July 2006.

[18] M. uttel and T. Braun. A novel position-based and beaconless routing algorithm for mobile ad-hoc networks, 2003.

[19] A. Woo, T. Tong, and D. Culler. Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks. In *Proc. First International Conference on Embedded Networked Sensor Systems (SenSys '03)*, pages 14–27. ACM Press New York, NY, USA, 2003.

[20] J. Zhao and R. Govindan. Understanding Packet Delivery Performance in Dense Wireless Sensor Networks. In *Proc. First International Conference on Embedded Networked Sensor Systems (SenSys '03)*, pages 1–13, New York, NY, USA, 2003. ACM Press.

[21] J. Zhao and R. Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *Proc. of the first international conference on Embedded networked sensor systems*, pages 1–13, 2003.

[22] M. Zorzi. A new contention-based MAC protocol for geographic forwarding in ad hoc and sensor networks. In *Proc. of the IEEE International Conference on Communications (ICC '04)*, volume 6, pages 3481–3485, 2004.

[23] M. Zorzi and R. Rao. Geographic random forwarding (GeRaF) for ad hoc and sensor networks: energy and latency performance. *Mobile Computing, IEEE Transactions on*, 2(4):349–365, 2003.