On the Improvement of the Enhanced Distance Based Broadcasting Algorithm

Patricia Ruiz* and Pascal Bouvry

Faculty of Science, Technology and Communication (FSTC)
University of Luxembourg,
6, rue Richard Koudenhove-Kalergi,
L-1359, Luxembourg
E-mail: patricia.ruiz@uni.lu, pascal.bouvry@uni.lu
*Corresponding author

Abstract: Energy efficiency has attracted a lot of attention during the recent years, especially in mobile ad hoc networks where devices rely on batteries. Moreover, in this kind of networks the cooperation among devices forwarding packets or acting as a router is crucial for the network performance. Thus, reducing the energy consumption of each sending is a key feature that has been widely studied. In this paper we extend the study made for EDB, an enhanced distance based broadcasting algorithm that aims at minimising the transmission power needed to broadcast a message in the network, by adding different network densities and values for the thresholds. Results showed that reducing the transmission power does not cause any detriment in the network connectivity but it can also increase the performance of the dissemination process.

Keywords: Energy efficient, mobile ad hoc networks, distributed system, cross layer design, distance based broadcasting.

Reference to this paper should be made as follows: Ruiz, P. and Bouvry, P. (xxxx) 'On the Improvement of the Enhanced Distance Based Broadcasting Algorithm', *Int. J. of Communication Networks and Distributed Systems*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Patricia Ruiz received her degree in telecommunication engineering from the University of Málaga (Spain), and she received a national award for her master thesis. She is currently working as a junior researcher at University of Luxembourg, but previously she was working for the public research centre Henri Tudor at Luxembourg. Her main research interests are mobile and vehicular ad hoc networks, crosslayer design, energy efficiency, broadcasting algorithms and wireless communications.

Pascal Bouvry earned his Ph.D. degree (94) in computer science at the University of Grenoble, France. He is now Professor at the Faculty of Sciences, Technology and Communication of the University of Luxembourg and heading the Computer Science and Communication research unit (http://csc.uni.lu). Pascal Bouvry is specialised in parallel and evolutionary computing. His current interest concerns the application of nature-inspired computing for solving reliability, security, and energy-efficiency problems.

1 Introduction

Communicating mobile devices using wireless technology at any time and any moment without the need of a previous existing infrastructure is attracting the attention of the research community. This kind of networks is called mobile ad hoc networks, MANETs hereinafter. In order to be able to deliver a message to a node outside the transmission range, each node must act also as a router. These networks have many challenging aspects like the appearance and disappearance of devices due to the limited transmission range, the battery life, the mobility of devices, obstacles in the environment, the network partitioning, etc. All these undesirable behaviours are being tackled by many researchers.

As we said before, MANETs are composed of portable devices that move, and that rely on batteries. The energy consumption is, therefore, a key feature in this kind of networks because when devices run out of battery, the number of nodes in the network decreases and the number of partitions increases reducing the capabilities of the network. This topic has been extensively addressed in the literature, some examples are the ones presented by X. Li. (2003); J. Gomez and A.T. Campbell (2007); M. Cagali et al. (2002, 2005); W. Liang et al. (2009).

The intrinsic broadcast nature of wireless networks makes broadcasting one of the most suitable algorithms for neighbour discovery, routing, etc. Moreover, broadcasting is considered as one of the basis for many high level applications and even other protocols assume the existence of a broadcast service. That is the reason why many researchers try to optimise these algorithms by maximising the number of nodes reached and minimising both the network and device resources as it was done by E. Alba and B. Dorronsoro (2008).

During the dissemination of a message *the broadcast storm problem* presented in S. Y. Ni et al. (1999) may occur. Many different techniques for minimising these effects like the probabilistic, the counter based, distance based, location based and cluster based schemes, have been proposed by S. Y. Ni et al. (1999). All these approaches try to minimise the number of forwarding nodes in the dissemination process. In this work, we focus on one of these approaches: the distance based (DB), that limits the number of rebroadcast in terms of the distance. Indeed, we make a more detailed study of the enhanced distance based broadcasting algorithm presented by P. Ruiz and P. Bouvry (2010) (EDB hereinafter), considering more network densities, and different values for the thresholds.

This paper is an extension of the one proposed by P. Ruiz and P. Bouvry (2010), therefore the contributions are: (1) adding energy efficiency features to the distance based approach by reducing the transmission range of the forwarding nodes, (2) analysing the influence that reducing the transmission power has over the network performance in terms of the number of collisions, (3) making an intensive study on the effect that changing the technique used for setting the delay has on the behaviour of the algorithm and, finally (4) studying in depth the values for the thresholds.

The remainder of this paper is organised as follows. Next Sect. presents a small state of the art in the topic. Section 3 presents the original distance based broadcasting algorithm, and the improvements added to it are explained in Sect. 4. The parameters used for the simulations and the results obtained are shown in Sect. 5 and Sect. 6, respectively. Finally, Sect. 7 concludes the paper.

2 Related Work

In this work we are considering an algorithm of the state of the art: the distance based approach (DB) presented by S. Y. Ni et al. (1999). It is possible to find different proposals in the literature, such as D. Liarokapis et al. (2009), where authors designed a distance based broadcasting protocol that does not exchange any *hello* messages. Moreover, the transmission range varies according to the number of retransmissions heard of the same packet.

In C. Zhi-yan et al. (2007) an improved version of DB is presented adding counter based features to the border-aware scheme. It also considers the remaining energy of the nodes.

As we mentioned before, the energy consumption in mobile ad hoc networks is a hot topic, since devices can run out of battery provoking the network degradation. Some of the solutions that have been proposed for saving energy in broadcasting algorithms dealing with ad hoc networks are mentioned below.

J. Gomez and A.T. Campbell (2007) showed that a variable transmission range can outperform a common range transmission approach in terms of power saving, increasing as well the capacity of the network. They also claim that there is an optimum setting for the transmission range, not necessarily minimum, which maximises the capacity available of the nodes in presence of mobility.

X. Chen et al. (2003) proposed an algorithm where nodes exchange information in the beacons in order to know the transmission power needed to reach the two hop neighbours. The source node examines whether it is worth or not to exclude the furthest node from the one hop neighbourhood and reduce the transmission range to reach the new furthest neighbour.

An approach to estimate the local density using an analytical model is used in X. Li. (2003) to set the transmission range according to this estimation.

Extensive studies on energy efficient algorithms for finding the minimum-energy broadcast tree (MEBT) have been proposed like M. Cagali et al. (2002, 2005), and also in W. Liang et al. (2009) where a shared multicast tree built in a distributed fashion with minimum energy is presented. The transmission power is either fixed or adjustable.

In S. L. Wu et al. (2002) each node continuously monitors, records and updates the transmission power level it needs to reach all its neighbours by overhearing all messages, even the ones that are not intended for it. This is not a broadcasting approach, so when sending a message, the source node will use the power needed to reach the intended neighbour (via unicast).

P. Ruiz and P. Bouvry (2010) proposed an enhanced distance based broadcasting algorithm, that not only considerably reduces the number of collisions but also reduces the energy consumption in the broadcast process without degrading the network connectivity.

In vehicular ad hoc networks it is also a tendency to adjust the transmission range used. In H. Reumerman and M. Runi (2005) nodes exchange periodically beacons containing information about the path loss. Neighbours are sorted according to the average path loss and when a broadcast message is received, the node checks the transmission power necessary to reach a targeted number of nodes.

3 Distance Based Broadcasting Algorithm, DB

As we explained before, DB is one of the different schemes proposed by S. Y. Ni et al. (1999) for minimising the effects of the broadcast storm problem when disseminating information in wireless networks. The protocol needs to know the distance between the source node and the receiver (they do not specify how to obtain this metric). The idea is that a node receiving a broadcast message for the first time will compute the distance to the source node. If this distance is short, forwarding the message does not significantly add to the coverage, thus, the message is not rebroadcast. This is shown in Figure 1.

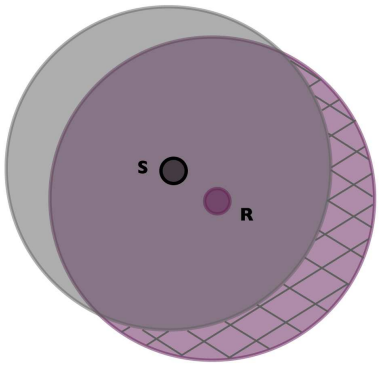

Figure 1 The additional coverage provided to the broadcasting process.

As stated in S. Y. Ni et al. (1999) the distance from source node to the receiver is clearly related to the additional coverage obtained in case of forwarding, so it can be used as a metric. Only nodes that are separated at least a minimum distance from the source node resend the message. This minimum distance is a predefined threshold, D. Moreover, the protocol includes a delay before forwarding a received message, and if the same message is heard more than once (during this waiting time), the delay is cancelled, the distance between the current node and the new sender is calculated and if it is higher than D, the message is resent.

The behaviour of the algorithm is represented in Figure 2. Considering node A broadcasts a message m, nodes B and C do not resend m because the distance from those nodes to A is smaller than D. Nodes E, F and G wait for a random number of slots. If node F finishes the waiting time first, it forwards the message and, thus, node E hears it and calculates the distance to node F. As the distance between E and F is smaller than D, node E drops the packet. The pseudocode of the protocol is shown in Algorithm 1.

4 Enhanced Distance Based Broadcasting Algorithm, EDB

4.1 Implementation

As we said before, when it is necessary to calculate the distance between a source and a destination, the most common technique is either assuming a GPS service

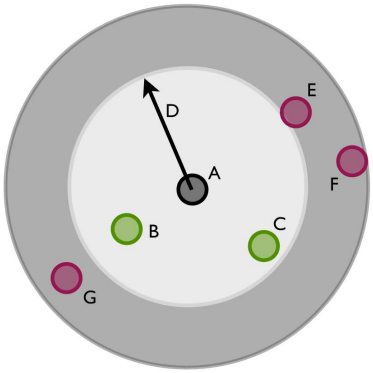

Figure 2 Mechanism of the distance based broadcasting protocol (DB).

Algorithm 1 Pseudocode of DB.

Data: m: the incoming broadcast message.
Data: r: the node receiving broadcast message.
Data: s: the node that sent m.
Data: d: the distance between r and s.
Data: $dmin$: the minimum distance between r and any s.
Data: D: the distance threshold.

```
1: if m is received for the first time then
2:     calculate d
3:     update dmin;
4:     if dmin < D then
5:         r → drop message m;
6:     else
7:         wait random number of slots;
8:         waiting = true;
9:     end if
10: else if waiting then
11:     calculate d
12:     resume waiting
13:     waiting = false;
14:     if d < dmin then
15:         update dmin;
16:     end if
17:     if dmin < D then
18:         r → drop message m;
19:     end if
20: end if
21: transmit m
```

or considering the signal strength. In this work, we do not assume that all devices in our network must provide a GPS service. Therefore, we use the signal strength of the received packets to estimate how far two nodes are.

There are some approaches where a propagation path loss model for the channel is selected and the distance between the nodes according to this model is estimated. C. Zhi-yan et al. (2007) considered free space propagation model, and used that equation to calculate the distance thanks to the reception power (assuming all devices use the same transmission power).

In this work, we are not assuming any concrete propagation path loss model. The received power is related to the distance, although we are not interested in the distance itself but in the energy lost during the transmission. Considering this,

a device close to the source node but with some buildings around (weakening the signal), will forward the message contributing to the process in an area where the dissemination of the message is not easy. Using this implementation we are also aware of the non perfect shape (generally considered as a disk) of the transmission range, that generally depends on the environment.

So for us, the threshold D is not in terms of distance (meters) but power (dBm). In our implementation, we called it *borders_Threshold* as it defines the nodes that are considered to be far from the source and therefore close to the border of the transmission range. The value we are using for this parameter is *-90 dBm*. This value was experimentally chosen and represents one third of the total transmission range. We also consider a node is not able to decode a received packet if the reception power is lower than *-95 dBm*, this is called the *end_Threshold*. Therefore, all nodes receiving a message whose reception energy varies from [-95, -90] dBm are candidates of forwarding the broadcast message.

Every device sends a *hello* message (or beacon) to alert devices within a close area about their presence. A device receiving these beacons is able to keep track of all neighbours around. We assume that all devices send the beacons with the same transmission power (default transmission power, 16.02dBm).

We are considering here a crosslayer design where the physical layer informs the upper layers about the received signal strength of each beacon and message received. In this situation, the algorithm is able to take decisions depending on this value. When a broadcast message is sent, the receiver will check the reception power, if it is below the *borders_Threshold* (*-90 dBm*), it will consider itself as a border node (locate in the forwarding area and thus, candidate for resending the message) and thus, it will set the delay.

4.2 Enhancements

As we are dealing with ad hoc networks where devices depend on battery (energy consumption is a critical aspect), we are interested in saving energy in order to extend the lifetime of the devices. One of the new features added to DB is reducing the transmission power, thus energy used. Moreover, we also consider and analyse different settings for the delay established when a message is received.

4.2.1 Reducing transmission power

In any wireless transmission, as the electromagnetic wave propagates through the space, the power of the signal suffers from path loss attenuation causing a reduction in the signal strength. The relation between the transmitted power and the power finally received at the destination directly depends on the loss suffered during the transmission. Equation 1 represents the relation in terms of dB.

$$receivedPower = transmittedPower - loss \tag{1}$$

If we assume that all nodes send the *hello message* with the same transmission power (16.02 dBm), a node receiving a beacon will be able to estimate the loss that packet suffered during its transmission, using the reception power detected at the physical layer.

Every node keeps and updates the reception power of each of its neighbours in a list, so that, when a device wants to send a broadcast message, it will be able to estimate the loss the packet will suffer (we assume a packet traversing in a direction will experiment the same loss as another traversing in the opposite direction).

If a node can estimate the loss the packet is going to suffer, it will be able to reduce its transmission power and use only the necessary one to get the furthest one hop neighbour. Thus, reducing the transmission power for sending the broadcast message provides some reduction in the energy consumption of the device, without degrading the performance of the broadcasting process since we do not consider loosing the connection with any neighbour.

When the loss the packet suffered due to the propagation is calculated, the node can estimate the transmission power needed to reach the furthest neighbour in the one hop neighbourhood. If we are reducing the transmission power, the furthest node is receiving the packet with the minimum possible reception power allowed to correctly decode the message. That means, its reception power should be the *end_Threshold*. The new reduced transmission power can be calculated as shown in Equation 2.

$$transmissionPower = loss + end\_Threshold \tag{2}$$

Once the new transmission power is estimated, in terms of the reception energy stored using the beacons, it is necessary to consider that the devices do move and the information can be out of date since beacons are sent every 1 second. Therefore, we are considering a margin of error (*margin_Forwarding*) that is added to the estimated transmission power. This value was experimentally chosen and was estimated considering the loss a packet at the border of the transmission range might suffer when the node moves for 1 second (as beacons are sent every second). Its value is 0.5 dBm. Therefore the new transmission power is shown in Equation 3.

$$newTransmissionPower = transmissionPower + margin\_Threshold \tag{3}$$

From Equation 2 it is possible to estimate the maximum transmission power needed to reach the furthest neighbour in the one hop neighbourhood. If it is less than the default transmission power, we reduce it in order to save energy, as the extra energy used is useless. This is shown in Figure 3, where it is possible to see that all the one hop neighbours are close to the current neighbour. Therefore, reducing the transmission range from r to r' decreases the energy consumption with no detriment of the network connectivity or the broadcast performance.

Reducing the transmission power for sending broadcast messages not only improves the energy consumption in wireless networks but also reduces the interference level of devices in a close area.

There is a minimum amount of reception energy needed to decode the message, that minimum level is called the *end_Threshold*. If the received signal strength is lower than this value, the device is not be able to recover the data transmitted. The problem is that this signal is still received and will be considered as noise, increasing the interference level of the device (see S. Basagni et al. (2004) for a detailed explanation).

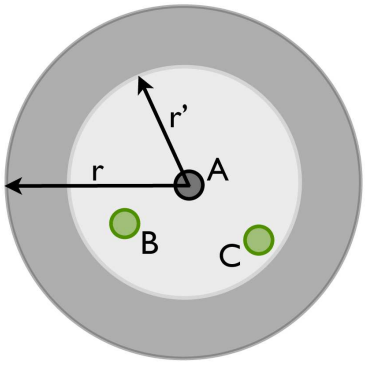

Figure 3 Reducing the transmission power (thus, the transmission range) of the node.

We can observe on the left side of Figure 4 that both nodes A and B use the default transmission range r. In this situation node C is not in range either with node A or B but it suffers from their interference. However, if both nodes A and B reduce their transmission range to the maximum needed (r') to reach the furthest node in the one hop neighbourhood, as it is shown on the right side of Figure 4, node C will not receive anything from A or B, and thus, the interference level will not affect C or at least it will be reduced.

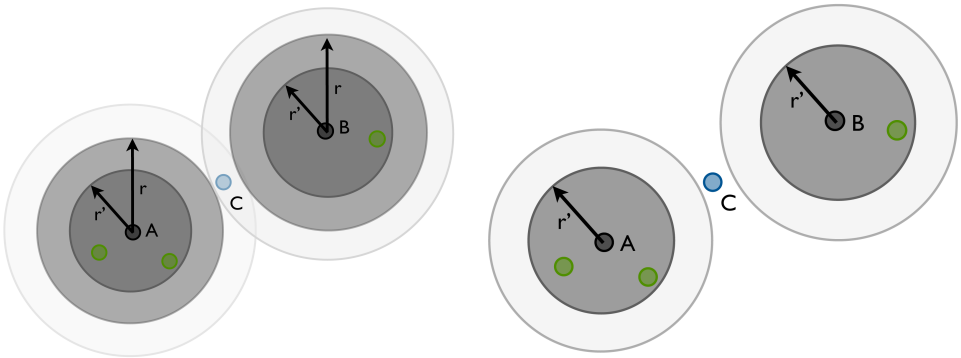

Figure 4 Reducing the interference level.

According to V. Kawadia and P. Kumar. (2005), the transmission power affects many aspects of the network just as the transmission range, and thus, the connectivity of the network (the lower the power, the smaller the transmission range), the performance of the medium access, since it depends on the number of nodes within range, the capacity of the network, etc. The network connectivity in this case is not decreased, since nodes try to reduce the transmission power but considering all neighbours in range. Neither the contention for the medium access is, exactly for the same reason explained before. Also according to them, the capacity is increased as the transmission power level is reduced by decreasing the interference area (it is proportional to the square of the transmission range).

Reducing the transmission power not only reduces the energy consumption of the device but also helps in the dissemination of the message when dealing with distance based broadcasting algorithms. In DB if the source node only has one neighbour and it is not located in the forwarding area the message is not rebroadcast, even if that node has many other neighbours around (as shown in Figure 5 B will not rebroadcast), thus, the dissemination process is stopped. A graphical explanation is provided next. On the one hand, we can see in Figure 5 that node B is not in the forwarding area of node A, therefore, the message is not forwarded and the dissemination is finished even if many nodes did not receive the message. On the other hand, if node A is reducing the transmission power so that the furthest neighbour (B) is reached, node B will be in the forwarding area, thus, it resends the message and the rest of nodes receive the message.

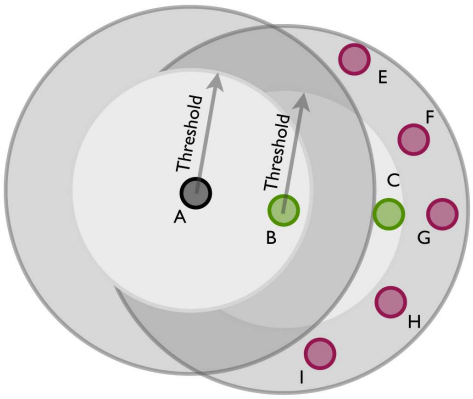

Figure 5 Advantage of reducing the transmission range.

4.2.2 Using different delay techniques

In the original implementation DB stops the random delay when a repeated message is heard. Then, if the distance from the new source node is smaller than the threshold D, the message is discarded and no retransmission is performed. Otherwise, the forwarding starts.

In our implementation, we consider keeping track of the received energy and continue the delay, instead of stopping the delay when a repeated message is heard. Once it is finished, the forwarding decision is taken according to the received signal strength of all the copies heard of the same message. In P. Ruiz and P. Bouvry (2010), it was proven that this mechanism highly reduce the number of collisions (the densest network studied was 700 devices).

In DB the delay is randomly chosen from a predefined interval. We also propose to adopt a similar scheme as the one presented by A. Benslimane et al. (2004) where the delay is fixed and inversely proportional to the distance between the receiver and the source node. In this situation, a node closer to the source will rebroadcast later than a node far from it.

In our work, we are studying the behaviour of two different techniques and we are comparing them to the original proposal of DB:

1. the first proposal considers a fixed delay inversely proportional to the received power. The process for calculating the delay is shown in Equation 4:

$$powerDelay = \frac{-1}{rxPower - borders\_Threshold - 1} \tag{4}$$

   In Equation 4, the procedure to calculate the delay in terms of the reception power is shown. If a node is setting a delay, it means, the node is a *border node*, otherwise the node is not considered a candidate to forward the message, and therefore, no delay is set. All *border nodes* receive the message with a reception power that can vary between the *borders_Threshold* (*-90 dBm*) and the *end_Threshold* (*-95 dBm*). Therefore, considering Equation 4, we can check that the delay varies between 0.167 and 1 second. The higher reception power (closer neighbours), the longer the delay, and vice versa;

2. the second proposal considers a random delay chosen from an interval whose size also varies with the reception power. That is, the waiting time will be chosen between [0, *powerDelay*] calculated as in Equation 4.

We are comparing different techniques: (1) *RandomDelay*: the delay is chosen randomly from the interval [0 1] s; (2) *FixedDelay*: the delay is fixed with the value *powerDelay*; and finally (3) *PowerDelay*: the delay is chosen randomly from the interval [0 *powerDelay*] s.

The pseudocode of the enhanced distance based broadcasting protocol proposed in this work is shown in Algorithm 2.

5 Simulations

We are using ns-3 simulator, proposed by M. Lacage et al. (2006), for experimentally evaluating the performance of the new features we are adding to the distance based broadcasting protocol. ns-3 is a discrete event simulator written in C++ and with an optional Python scripting API. To validate the enhancements we are providing to the algorithm, we are comparing the original DB to EDB in terms of different parameters:

- the number of collisions due to the broadcasting process;
- the energy used per forwarding when reducing the transmission power of the source node;
- the coverage achieved by the broadcasting message;
- the network usage, in terms of the number of nodes forwarding the broadcast message.

Moreover, we also analyse the behaviour of EDB with different values of the thresholds in terms of:

- the energy used;
- the coverage achieved;

Algorithm 2 Pseudocode of EDB.

Data: m: the incoming broadcast message.
Data: r: the node receiving broadcast message.
Data: s: the node that sent m.
Data: p: the received signal strength of m sent by s.
Data: $pmin$: the minimum signal strength received for m from any s.
Data: $borders\_Threshold$: the signal strength threshold.

```
 1: if m is received for the first time then
 2:     calculate p;
 3:     update pmin;
 4:     if pmin > borders_Threshold then
 5:         r → drop message m;
 6:     else
 7:         waiting = true;
 8:         wait time RandomDelay/FixedDelay/PowerDelay;
 9:         goto 17;
10:     end if
11: else if waiting then
12:     calculate p;
13:     if p > pmin then
14:         update pmin;
15:     end if
16: end if
17: if pmin > borders_Threshold then
18:     r → drop message m;
19: else
20:     estimate power to reach furthest neighbour
21:     transmit m;
22: end if
23: waiting = false;
```

- the broadcast time.

As we are dealing with mobile ad hoc networks, it is necessary to set a mobility model for the devices. In this case, we are using the *random walk* also known as *brownian motion mobility model* presented in R. B. Groenevelt et al. (2005), in which nodes move with a speed and direction randomly chosen during a fixed amount of time. After, new random speed and direction are chosen. In this work, we are considering 20 seconds (*time moving* hereinafter). If a node hits one of the boundaries of the area, it rebounds on the boundary with a reflexive angle and speed. The simulation environment used is a square area of 2000 m size (4 Km^2) . The speed of the nodes can vary from 0 to 2 m/s (between 0 and 7.2 Km/h). We measure our experiments with different network densities: the number of nodes varies from 100 up to 1000 in steps of 100 devices. In Table 1, we present a summary with the configuration we are using for the simulations.

Table 1 Configuration used

Number of devices	100-1000
Speed	[0 2] m/s
Size of the area	2000 x 2000 m
Transmission power	16.02 dBm
end_Threshold	-95 dBm
borders_Threshold	-90 dBm
margin_Forwarding	0.5 dBm
Delay interval	[0 1] s
Time moving	20 s

Table 2 Different variants of DB and EDB

StopRandomDelayDB	original DB with random delay ∈ [0, 1]
StopRandomDelayEDB	DB with energy estimation (EDB) with random delay ∈ [0, 1]
StopFixedDelayDB	DB with fixed delay = *powerDelay*
StopFixedDelayEDB	EDB with fixed delay = *powerDelay*
StopPowerDelayDB	DB with delay ∈ [0, *powerDelay*]
StopPowerDelayEDB	EDB with delay ∈ [0, *powerDelay*]
NonStopRandomDelayDB	DB with random delay ∈ [0, 1] & not stop delay
NonStopRandomDelayEDB	EDB with random delay ∈ [0, 1] & not stop delay
NonStopFixedDelayDB	DB with fixed delay = *powerDelay* & not stop delay
NonStopFixedDelayEDB	EDB with fixed delay = *powerDelay* & not stop delay
NonStopPowerDelayDB	DB delay ∈ [0, *powerDelay*] & not stop delay
NonStopPowerDelayEDB	EDB delay ∈ [0, *powerDelay*] & not stop delay

As we explained before we are comparing different variants of the broadcasting distance based protocol (DB) and the new EDB. In Table 2, all the different proposals are explained.

6 Results

For obtaining reliable results we are considering 100 different independent topologies for each of the different densities we are using. We are considering very partitioned networks with 100 devices in 2000 × 2000 m (4Km^2) and we are increasing the number of nodes by 100 in each simulation until we reach to 1000 devices. That means the density varies from 25 devices/Km^2, which is a really sparse network, to 250 devices/Km^2. In the experiments, the network evolves for 30 seconds (so that devices are uniformly distributed all over the simulation area), and at that moment, a node starts the broadcast process. The simulation stops 20 seconds later. Statistical analysis were made to all the results presented in this paper. This statistical study was done using either the Anova or the Kruskal-Wallis tests, depending on whether the data follow a normal distribution or not. In order to know if the data follow it, we apply the Kolmogorov-Smirnov test. In light grey the proposal that shows the worst performance with significant differences all over the others is marked, and in dark grey the one with, statistically, the best behaviour. In bold font the highest average value is shown. In tables, the last row shows whether there are statistical differences between any variant in each density. A + for the p-value means there are significant differences at least between two algorithms.

6.1 Collisions

We consider a collision can be produced due to the reception of a packet when the device is already synchronised or transmitting. In Table 3, the number of collisions produced due to the broadcasting process only is shown for each density and the two variants of algorithms (DB and EDB). The values presented there represent the number of devices detecting the collisions, that is, for example, in a neighbourhood composed of 10 devices (all in range), if a message is sent while another node is still transmitting, all devices will detect the collision (it is not one collision but 10 devices detecting a collision).

Table 3 Average number of collisions in the dissemination process

Variants of Protocols	100	200	300	400	500	600	700	800	900	1000
NonStopRandomDelayDB	0.02 ±0.20	0.25 ±1.54	0.97 ±3.19	5.19 ±7.96	15.81 ±18.49	37.22 ±25.03	74.66 ±48.04	124.00 ±59.59	**173.13** ±78.30	248.81 ±102.88
NonStopRandomDelayEDB	**0.00** ±0.00	0.22 ±1.41	1.55 ±3.98	**5.30** ±7.79	**19.03** ±19.73	**38.92** ±28.06	**69.96** ±48.85	**114.59** ±57.46	193.84 ±74.82	**238.81** ±99.41
StopRandomDelayDB	0.09 ±0.73	1.01 ±4.31	16.60 ±25.11	101.98 ±85.64	391.55 ±183.51	810.55 ±275.51	1428.18 ±416.31	2384.31 ±640.28	3857.89 ±959.43	5768.80 ±1401.34
StopRandomDelayEDB	0.08 ±0.56	1.02 ±4.62	23.54 ±32.88	128.26 ±96.66	367.17 ±147.08	812.67 ±289.21	1603.83 ±434.92	2673.45 ±652.44	4126.23 ±909.85	6117.96 ±1804.76
NonStopFixedDelayDB	**0.00** ±0.00	0.09 ±0.73	**0.67** ±3.16	8.40 ±10.99	29.43 ±24.07	55.29 ±33.78	102.13 ±54.40	142.24 ±69.22	208.75 ±98.45	300.71 ±109.14
NonStopFixedDelayEDB	**0.00** ±0.00	0.09 ±0.64	1.53 ±4.18	9.62 ±11.57	27.86 ±23.97	55.02 ±34.73	100.17 ±52.64	143.55 ±58.73	238.63 ±109.92	307.59 ±126.12
StopFixedDelayDB	0.04 ±0.40	2.37 ±8.33	19.18 ±25.23	116.10 ±85.05	379.34 ±150.99	853.52 ±225.14	1526.04 ±448.75	2603.09 ±657.98	4067.83 ±1138.21	6363.69 ±1520.48
StopFixedDelayEDB	0.04 ±0.40	2.67 ±7.59	16.85 ±23.33	116.72 ±90.84	421.88 ±179.17	932.97 ±275.46	1644.51 ±395.78	2697.93 ±752.37	4483.52 ±1254.70	6563.00 ±1818.84
NonStopPowerDelayDB	**0.00** ±0.00	0.10 ±0.64	2.31 ±4.94	17.30 ±19.36	46.00 ±35.79	128.35 ±60.79	197.53 ±84.35	356.54 ±134.30	474.41 ±181.83	600.86 ±207.42
NonStopPowerDelayEDB	**0.00** ±0.00	**0.07** ±0.50	1.33 ±3.55	17.92 ±22.78	46.76 ±36.22	144.35 ±72.01	230.86 ±107.36	365.85 ±148.45	503.95 ±177.88	661.81 ±224.86
StopPowerDelayDB	0.12 ±0.86	1.13 ±4.31	13.72 ±20.76	121.49 ±96.05	408.75 ±190.54	943.61 ±318.06	1688.47 ±458.08	2851.90 ±819.83	4566.63 ±1277.47	7685.82 ±2116.07
StopPowerDelayEDB	0.14 ±1.00	1.64 ±5.25	17.43 ±28.63	151.34 ±113.40	426.24 ±211.19	960.23 ±318.92	1769.64 ±444.99	3121.64 ±868.90	5069.38 ±1450.58	8159.67 ±2443.67
p-value	-	+	+	+	+	+	+	+	+	+

It is possible to check that in the sparse densities the number of collisions is very low, i.e. with a network composed of 100 devices the maximum average number of collisions is 0.14 for any protocol. We must remark the big difference between two different versions of the protocols, the one that stops the waiting time and the one that does not (no matter which value of the delay is set). In all protocols that do not stop the delay when a copy of the message is heard, the number of collisions is much lower than in the ones stopping. That was an expected result as devices in a close area receive a duplicate copy of the message almost at the same time, and therefore, they stop the delay and forward the message (if this is the case) at the same moment.

There is a huge difference in the average number of collisions between the proposals that stop and the ones that do not stop the delay. For example, the original DB (*StopRandomDelayDB*) in a network of 1000 devices has an average number of collisions of 5768.80 while the same protocol just finishing the delay (*NonStopRandomDelayDB*) has an average number of collisions of 248.81, what means 95.68% less. Moreover, the same configuration but reducing the transmission power (*NonStopRandomDelayEDB*) detects less number of collisions 238.81.

The variant that chooses a random delay is the one detecting less number of collisions. That is logical because the value that each device sets for the delay is randomly chosen, meanwhile in the other cases, it depends on the distance between the source and the current node. Meaning that, devices nearby will have a similar value. Moreover, in the *NonStopPowerDelayEDB* variant the number of collisions is higher than in the others because the value of the delay is randomly chosen between [0, *powerDelay*]. In this case, the maximum possible value is *powerDelay*, which is smaller as the neighbours are closer to the limit transmission range (0.167 seconds). So that, for dense networks where the probability of nodes close to

the limit transmission range is high, nodes will have a small value for the delay, increasing therefore, the probability of collisions.

The statistical tests have been applied to all the variants of the protocols and all the densities for comparing the average number of collisions. Results showed that for 100 devices there is no statistical differences in any case. For 200 devices, both variants of *StopFixedDelay* are worse than all the *NonStopDelay* ones. For a network of 300 devices, all the different proposals of *StopDelay* are statistically worse than the *NonStopDelay*, and this behaviour is also presented for all the others densities. For 500 up to 1000 devices, not only the *StopDelay* variants are worse than the *NonStopDelay* ones, but also *NonStopRandomDelay* is statistically better than *NonStopPowerDelay*.

From these first results showing the difference in the performance between stopping the delay or not when a duplicate copy of the message is heard we can consider the possibility of discarding all the stopping variants from the comparison.

6.2 Energy

In Figure 6, the evolution of the average energy used per forwarding is shown in logarithmic scale. The amount of energy used is increasing up to a moment where it remains almost constant (little reduction is performed) with a value close to the default transmission power. This is due to the high density of nodes in the network, as the possibility of having a neighbour close to the limit transmission range is high, and therefore, no energy reduction is performed.

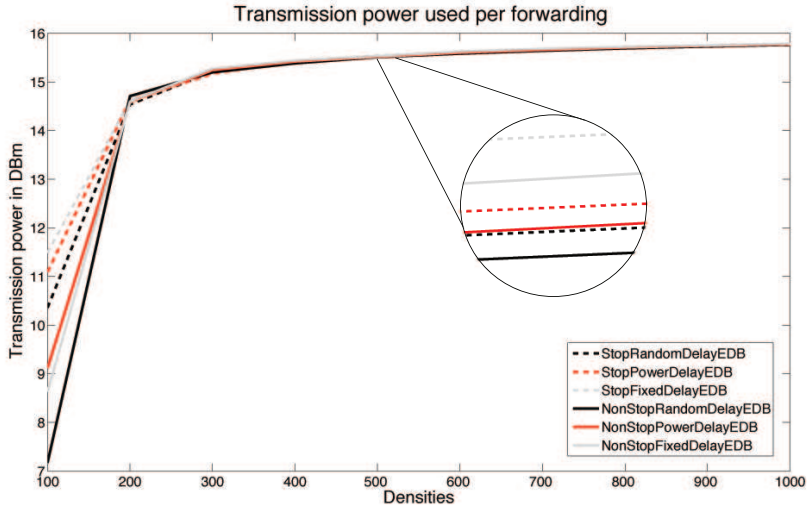

Figure 6 Energy used in each forwarding of the broadcasting process.

As we can observe from Figure 6, difference between the two main proposals (*NonStopDelay* and *StopDelay*) in terms of the energy consumption is similar. Moreover, as explained before the number of collisions is highly reduced for the (*NonStopDelay* approaches. Therefore, for now on, we will only focus on these

variants of the algorithms, the (*StopDelay* approaches have been discarded from our analysis.

Table 4 Average of the energy used per forwarding message in mWatts

Variants of Protocols	100	200	300	400	500	600	700	800	900	1000
AnyDB	40.00 ±0.0	40.00 ±0.0	40.00 ±0.0	40.00 ±0.0	40.00 ±0.0	40.00 ±0.0	40.00 ±0.0	40.00 ±0.0	40.00 ±0.0	40.00 ±0.0
NonStopRandomDelayEDB	24.12 ±12.87	**31.35** ±7.47	**34.04** ±4.32	**36.30** ±2.33	**37.39** ±0.46	**38.03** ±0.42	**38.46** ±0.29	**38.81** ±0.29	**39.07** ±0.22	**39.27** ±0.18
NonStopPowerDelayEDB	**21.11** ±13.03	31.98 ±6.26	34.38 ±4.57	36.50 ±0.93	37.48 ±0.69	38.09 ±0.40	38.57 ±0.30	38.96 ±0.25	39.15 ±0.21	39.35 ±0.16
NonStopFixedDelayEDB	23.30 ±12.81	31.36 ±7.74	34.99 ±4.22	36.66 ±1.69	37.60 ±1.93	38.39 ±0.32	38.81 ±0.28	39.13 ±0.22	39.28 ±0.48	39.46 ±0.16
p-value	+	+	+	+	+	+	+	+	+	+

To clearly see the differences presented in the energy consumption between the protocol variants, we show in Table 4 the average of the energy a node uses in each forwarding of the dissemination message (in Watts). It is marked with bold font the variants that consume, in average, less for each network density.

All variants of the original DB protocol consume 40 mW (16.02 dBm), the default transmission power. As all these DB variants present the same behaviour for the energy consumption, we have grouped them in Table 4, and they are represented as *AnyDB*. According to the results obtained from the study of the number of collisions, we are only considering the variants of EDB that do not stop the delay. In general, the best performance is found in the *NonStopRandomDelayEDB*. It reduces the energy from 1.35% for the densest network up to 60.3% for the 100 devices network. In bold font, the proposal with the lowest average value of the energy used is marked. We can see that, generally, EDB consumes less.

For all densities *AnyDB* showed the worst performance with significant differences (marked with light grey background). Considering the behaviour between the other three variants no statistical differences where shown for the sparsest networks (up to 300 devices). From 400 up to 1000 devices *NonStopFixedDelayEDB* always showed statistically worse results than either *NonStopRandomDelayEDB* or *NonStopPowerDelayEDB*.

6.3 Coverage

In terms of the coverage achieved by the broadcasting process, Table 5 shows the average number of devices reached for each density after 100 runs for the different variants. As we mention before, the results shown correspond to the variants that do not stop the random delay. We compare them with the original DB (*StopRandomDelayDB*).

The coverage achieved by the broadcasting process is very good with reasonable use of the network resources. For the 100, 200 and 300 densities, the network is very sparse and the coverage achieved is very low (the minimum for 100 devices is 3.35%), but as the density grows, the coverage also increases. In networks with 600 and 700 devices, the minimum coverage achieved between all protocols is 95.72% and 98.35%, respectively. For networks with density equal or higher to 800

Table 5 Average of devices reached in the dissemination process

Variants of Protocols	100	200	300	400	500	600	700	800	900	1000
StopRandomDelayDB	3.35 ±3.39	17.40 ±16.44	89.50 ±70.48	265.56 ±122.31	436.96 ±114.12	574.36 ±84.40	693.86 ±6.01	795.14 ±5.60	896.63 ±3.42	997.32 ±2.91
NonStopRandomDelayEDB	3.94 ±3.27	**23.88** ±22.61	92.46 ±69.00	297.35 ±108.89	**472.19** ±30.71	584.07 ±57.76	694.30 ±6.02	**796.28** ±3.96	**897.75** ±2.77	997.74 ±3.03
NonStopPowerDelayEDB	**4.08** ±3.91	23.70 ±22.06	**111.28** ±86.13	**299.64** ±109.39	443.03 ±123.87	**590.82** ±8.69	688.46 ±63.17	796.03 ±4.96	897.42 ±2.69	998.01 ±1.99
NonStopFixedDelayEDB	3.74 ±3.37	21.34 ±21.64	107.13 ±82.35	282.19 ±118.41	457.67 ±96.79	586.21 ±20.86	**694.60** ±7.31	796.04 ±4.55	888.40 ±88.97	**998.12** ±1.87
p-value	-	-	-	+	+	-	-	-	+	+

the minimum coverage achieved is higher than 98.5%, being 99.39%, 98.71% and 99.73% for 800, 900, 1000 devices respectively.

Statistical tests have been applied to the results. There are only statistical differences for 4 network densities 400, 500, 900 and 1000 devices. In 400 devices density, *StopRandomDelayDB* behaves statistically worse than *NonStopPowerDelayEDB*. For 500 devices, *StopRandomDelayDB* is significantly worse than both *NonStopPowerDelayEDB* and *NonStopFixedDelayEDB*, meanwhile it is worse than *NonStopRandomDelayEDB* and *NonStopPowerDelayEDB* for 900 network density. For the last configuration, with 1000 nodes, *StopRandomDelayDB* reaches significantly less coverage than both, *NonStopRandomDelayEDB* and *NonStopFixedDelayEDB*.

These results lead to an important conclusion: reducing the transmission power for disseminating a message does not decrease the number of devices that finally receives the dissemination message. Moreover, as explained in Section 4, there are some cases in which reducing the transmission power promotes the dissemination process and increases the coverage. This happens when the source node does not have neighbours in the forwarding area, thus, no rebroadcasting is performed. However, when reducing the transmission power to reach the furthest neighbour, there is at least one node (the furthest), that is in the forwarding area and thus, the message is forwarded.

6.4 Network Usage

The percentage of nodes that after receiving the dissemination message forwards it is presented in Table 6. There, it is possible to see that from 500 devices on the number of forwarding nodes is decreasing as the network density increases. The protocol that uses the lowest average number of forwarding node is in bold font. In grey light it is marked the proposal that is statistically worst than all the others (the one using highest number of forwarding), and in dark grey the statistically best one.

On the one hand, when the network is sparse (up to 500 devices), over 30% of forwarding nodes are needed. On the other hand, in the densest network less than 19% is necessary for the energy aware proposals, and 25.56% for the original DB. The reduction in the percentage of the number of nodes rebroadcasting in the densest network is at least 7.13%.

Statistical tests showed that for 100 devices, *StopRandomDelayDB* is the best one, and that in 200 devices the original DB outperforms *NonStopRandomDelayEDB*. In 300 and 400 device networks, no statistical differences

Table 6 Average of forwarding nodes

Variants of Protocols	100	200	300	400	500	600	700	800	900	1000
StopRandomDelayDB	**1.59** ±2.15	**3.82** ±4.05	**11.98** ±10.02	**24.36** ±11.65	28.87 ±7.86	28.98 ±4.55	28.26 ±1.30	27.19 ±1.43	26.33 ±1.26	25.56 ±1.12
NonStopRandomDelayEDB	2.37 ±2.30	5.88 ±5.97	12.78 ±9.91	26.80 ±10.18	29.48 ±2.56	26.79 ±2.85	24.31 ±0.70	22.04 ±0.59	20.03 ±0.43	18.43 ±0.46
NonStopPowerDelayEDB	2.43 ±2.39	5.92 ±5.71	15.64 ±12.40	26.99 ±10.31	**27.38** ±7.85	26.93 ±0.95	23.97 ±2.37	21.69 ±0.61	19.80 ±0.48	18.17 ±0.41
NonStopFixedDelayEDB	2.32 ±2.01	5.32 ±5.82	14.88 ±12.03	25.09 ±10.69	28.21 ±6.18	**26.28** ±1.38	**23.91** ±0.71	**21.52** ±0.61	**19.45** ±2.02	**18.05** ±0.44
p-value	+	+	-	-	+	+	+	+	+	+

were found between any protocol. From 500 devices on, the original DB (*StopRandomDelayDB*) is always worse in terms of the percentage of forwarding nodes with statistical differences over all the other variants. Moreover, for 600 devices *NonStopFixedDelayEDB* is the best one with significant difference. In 700 and 900 devices network *NonStopFixedDelayEDB* outperforms with statistical confidence *NonStopRandomDelayEDB*, meanwhile for 800 and 1000 is worse than both *NonStopFixedDelayEDB* and *NonStopPowerDelayEDB*.

Summarising all the results obtained we can say that all the variants that do not stop the delay upon the reception of a duplicate message have a better performance in terms of number of collisions than the ones that do stop the delay. Moreover, between the variants that do not stop the delay, *NonStopRandomDelayEDB* outperforms with statistical confidence *NonStopPowerDelayEDB*. For the coverage achieved there is not an algorithm that generally behaves better. Indeed, *StopRandomDelayDB* (the original DB) is never the best algorithm, what means that reducing the transmission power does not decrease the performance of the broadcasting process. Regarding the energy used per forwarded message, *NonStopRandomDelayEDB* has the lowest average values and along with the *NonStopFixedDelayEDB* are statistically the best ones. In the case of the number of the percentage of rebroadcast messages, *NonStopFixedDelayEDB* is the one that, for dense networks, generally behaves better in average.

6.5 Analysing different values of the thresholds

Broadly speaking, as a result of all these studies, we consider that *NonStopRandomDelayEDB* is the variant that, generally, provides better results. Only when considering the percentage of forwarded messages, *NonStopFixedDelayEDB* performs better results with statistical differences for 5 network densities than *NonStopRandomDelayEDB*, and also *NonStopPowerDelayEDB* for two of them. But, we must remark that the difference in the percentage of the number of devices in the case of *NonStopRandomDelayEDB* compared to the other variants, less than 1% in the case of *NonStopRandomDelayEDB*.

Therefore, we extend the experiments for this protocol variant: *NonStopRandomDelayEDB*, studying different values for the thresholds. The different configurations of the protocol are shown in Table 7.

In Figure 7 the average energy consumption (in mWatts) of *NonStopRandomDelayEDB* per forwarded message is shown for all the different densities using different values of the thresholds. As expected, in the cases where

Table 7 Variants of *NonStopRandomDelayEDB* with different values of the thresholds

Protocol variants	*Delay interval*	*borders_Threshold*	*margin_Forwarding*
NonStopRandomDelayEDB 01_90_0.5	[0, 1]	-90	0.5
NonStopRandomDelayEDB 01_90_1	[0, 1]	-90	1
NonStopRandomDelayEDB 01_92_0.5	[0, 1]	-92	0.5
NonStopRandomDelayEDB 02_90_0.5	[0, 2]	-90	0.5
NonStopRandomDelayEDB 02_92_1	[0, 2]	-92	1

margin_Forwarding is set to 0.5 dBm, the energy consumed per forwarded message is lower than the others approaches.

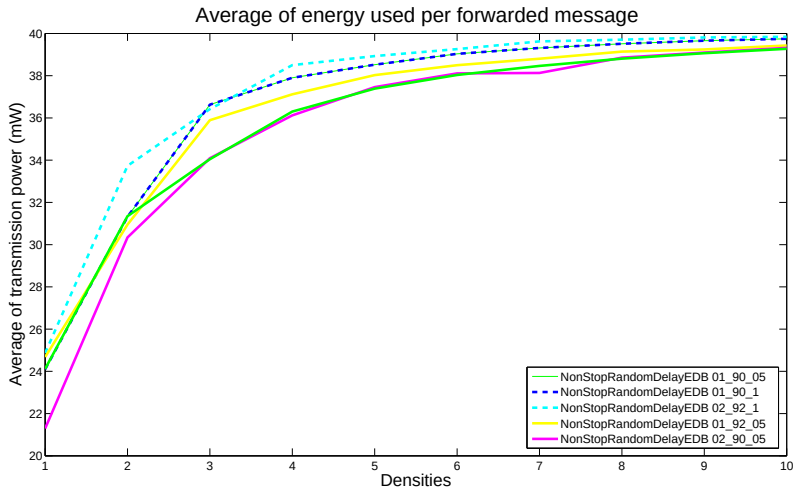

Figure 7 Average of the energy used in each forwarding of the broadcasting process.

The coverage achieved presented in Figure 8, is clearly lower in the case of the *NonStopRandomDelayEDB 02_92_1* and *NonStopRandomDelayEDB 01_92_0.5*, both configurations that set the *borders_Threshold* to -92 dBm. That is normal, as the lower this threshold, the smaller the forwarding area, and thus, the lower the number of potential forwarding nodes. From these results, we realised that the coverage achieved does not depend on the value of the *margin_Forwarding*, as there is almost no difference between *NonStopRandomDelayEDB 01_90_05* and *NonStopRandomDelayEDB 01_90_1*. In fact, this threshold was calculated to prevent the possible movements of nodes during 1 second (the periodicity of the beacons), so increasing it, should not provide extra coverage (as we can observe from Figure 8).

In Figure 9, the time necessary to broadcast a message in the network is shown. The behaviour of the different configurations is similar: in very sparse networks the broadcast process is very fast since it reaches only a few nodes of the network, but as density increases the broadcast time needed to spread the message increases too. This behaviour is shown until the network becomes denser (around 600 or 700 devices for the configurations with maximum value of 2 seconds delay, and 400

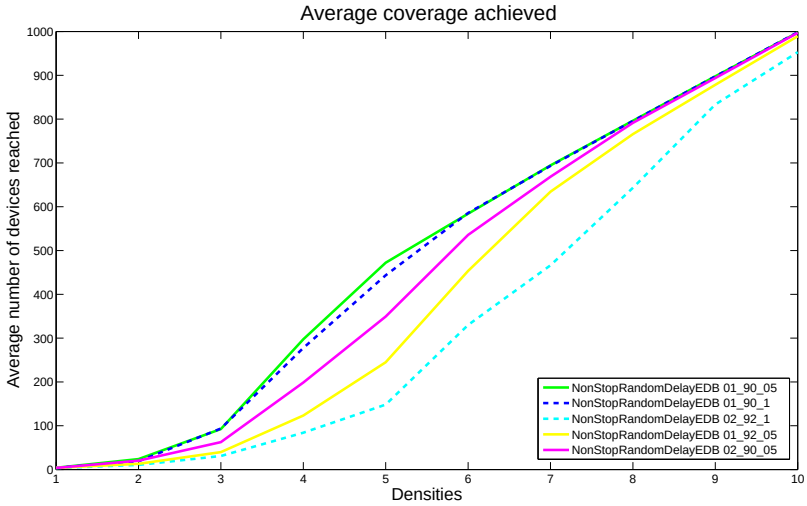

Figure 8 Coverage of the broadcasting process.

devices for the 1 second delay). After that, the process becomes faster as in denser networks the dissemination of a message is easier.

NonStopRandomDelayEDB 02_92_1 and *NonStopRandomDelayEDB 02_90_0.5* need a higher network density to start the drop of the broadcast time. These are the two variants that have higher value for the *delay interval* ([0 2]). As expected, the broadcasting process takes longer for those two variants.

For both *NonStopRandomDelayEDB 01_90_05* and *NonStopRandomDelayEDB 01_90_1* the behaviour is quite similar. But for *NonStopRandomDelayEDB 01_92_05* it takes shorter to disseminate the message in sparse networks, while longer in denser networks. This can be explained as the effect of having less forwarding nodes (due to a smaller forwarding area), what makes the dissemination longer.

After studying the behaviour of the protocol under different settings we can consider that *NonStopRandomDelayEDB 01_90_05* is the one that, generally, behaves better. It is one among all the variants consuming less energy, and achieving more coverage in less time. But it is not clearly always the best option among all the different variants. It means, that depending on the necessities of the application we could be able to tune the performance of the broadcasting protocol favouring one aspect (as energy consumption) among the others just by giving different values of the thresholds.

7 Conclusions and Future Work

In this paper we are proposing an energy saving strategy for the well known distance based broadcasting algorithm, EDB. We are also studying some different configurations for establishing the waiting time before the nodes forward the

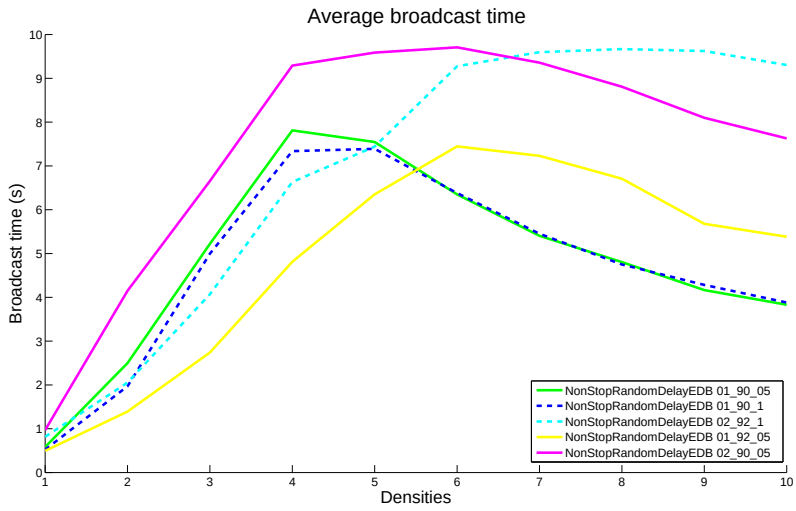

Figure 9 Broadcast time of the broadcasting process.

broadcasting message, and studying different values for the set of all the thresholds.

For reducing the energy consumption, we propose to decrease the transmission power when possible. That is, when the reduction in the transmission power does not imply the loss of any neighbour, and therefore, does not provoke network partitions. This is really useful when the network is not very dense reducing up to 60.3% in the best case, but when the number of devices is high, the node does not highly reduce the transmission power since there are usually nodes close to the border. However, in the worst case (densest network), this strategy of reducing the transmission power is saving at least 1.35% per forwarded message.

A study of different strategies for establishing the settings of the delay when a node receives a broadcast message was made. The original DB sets a random delay within a fixed interval, and when a message is heard for the second time, it cancels the waiting time and decides whether to forward the message or not. In this work, we are proposing different strategies: (1) allowing the node to finish the waiting time when the same message is received twice; (2) considering that the value of the delay is fixed, and inversely proportional to the distance from the source node (*powerDelay*); finally (3), the delay is randomly chosen from an interval which size varies from 0 to *powerDelay*. After some experiments, we can state that allowing the node to finish the waiting time highly reduces the number of collisions in the network. This reduction is getting more importance as the network is becoming denser.

As a result from the experiments performed, we can say that *NonStopRandomDelayEDB* is the one that generally behaves better. In terms of the coverage achieved we checked that reducing the transmission power does not decrease the number of devices reaches. Moreover, in some cases the coverage is increased. Regarding the energy consumption is the best one with statistical differences in many cases or does not have any with the best. And finally, in terms of the number

of collisions it is also the one that, statistically speaking, generally behaves better or has no differences with the best one.

After that, a deep study considering different values of the thresholds of *NonStopRandomDelayEDB* is done in terms of the energy used, the coverage achieved and also the broadcast time. The original proposed configuration showed in general, the best results. Additionally, we realised that depending on each situation, it can be more convenient to choose a set of values for the thresholds that promotes different objectives, i.e., a configuration that promotes the energy saving or a shorter broadcast time.

As future work, we plan to study a way to be able to highly reduce the energy consumption not only in sparse networks, but also in dense ones. The idea is to reduce the range of the transmission power discarding nodes from the one hop neighbourhood (the furthest ones) when the network is very dense. We would also like to optimise the protocol and find out the thresholds that maximise the coverage obtained by the broadcasting process, using the minimum energy consumption and broadcasting time.

References

E. Alba and B. Dorronsoro (2008), 'Cellular Genetic Algorithms', ser. Operations Research/Computer Science Interfaces. Springer-Verlag.

S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu (1999), 'The broadcast storm problem in a mobile ad hoc network', in *ACM/IEDBE Int. Conf. on Mobile computing and networking*, pp. 151–162.

X. Li, T. D. Nguyen, and R. P. Martin (2003), 'Using adaptive range control to optimize 1-hop broadcast coverage in dense wireless networks', in *SenSys*, pp. 314–315.

J. Gomez and A. T. Campbell (2007), 'Variable-range transmission power control in wireless ad hoc networks', *IEDBE Transactions on Mobile Computing*, vol. 6, no. 1, pp. 87–99.

M. Cagalj, J.-P. Hubaux, and C. Enz (2002), 'Minimum-energy broadcast in all-wireless networks: Np-completeness and distribution issues', in *Int. Conf. on Mobile Computing and Networking*, pp. 172–182.

M. Cagalj, J.-P. Hubaux, and C. C. Enz (2005), 'Energy-efficient broadcasting in all-wireless networks', *Wirel. Net.*, vol. 11, no. 1, pp. 177–188.

W. Liang, R. Brent, Y. Xu, and Q. Wang (2009), 'Minimum-energy all-to-all multicasting in wireless ad hoc networks', *Trans. Wireless Comm.*, vol. 8, no. 11, pp. 5490–5499.

D. Liarokapis, A. Shahrabi, and A. Komninos (2009), 'Diba: An adaptive broadcasting scheme in mobile ad hoc networks', in *Communication Networks and Services Research*, pp. 224–231.

C. Zhi-yan, J. Zhen-zhou, and H. Ming-zeng (2007), 'An energy-aware broadcast scheme for directed diffusion in wireless sensor network', *Journal of Communication and Computer*, Vol. 4,, No. 5, pp. 28–35.

X. Chen, M. Faloutsos, and S. V. Krishnamurthy (2003), 'Power adaptive broadcasting with local information in ad hoc networks', in *Conference on Network Protocols*. IEDBE Computer Society, p. 168.

S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic (2004), 'Mobile Ad Hoc Networking'. Wiley-IEDBE Press.

V. Kawadia and P. Kumar (2005), 'Principles and protocols for power control in wireless ad hoc networks', *Selected Areas in Communications, IEDBE Journal on*, Vol. 23, No. 1, pp. 76 – 88.

C. Zhu, M. Lee, and T. Saadawi (2004), 'A border-aware broadcast scheme for wireless ad hoc network', in *Consumer Communications and Networking Conference. CCNC 2004. First IEDBE*, pp. 134 – 139.

A. Benslimane (2004), 'Optimized dissemination of alarm messages in vehicular ad-hoc networks (VANET)', in *HSNMC*, pp. 655–666.

M. Lacage and T. R. Henderson (2006), 'Yet another network simulator', in *WNS2, workshop on ns-2: the IP network simulator*, p. 12.

R. B. Groenevelt, E. Altman, and P. Nain (2006), 'Relaying in mobile ad hoc networks: The brownian motion mobility model', *Journal of Wireless Networks*, pp. 561–571.

P. Ruiz, and P. Bouvry (2010), 'Enhanced Distance Based Broadcasting Protocol with Reduced Energy Consumption', *International Conference on High Performance Computing and Simulation (HPCS)*, pp.249–258.

S. L. Wu, Y. C. Tseng, C. Y. Lin and J. P. Sheu (2002) 'A Multi-channel MAC Protocol with Power Control for Multi-hop Mobile Ad Hoc Networks', *Comput. J.*, vol. 45, No. 1, pp. 101-110.

H. Reumerman and M. Runi (2005), 'Distributed power control for reliable broadcast in inter-vehicle communication systems', *2nd International Workshop on Intelligent transportation*.