Efficient multi-hop broadcasting in dense nanonetworks

Thierry Arrabal*, Dominique Dhoutaut* and Eugen Dedu*
*FEMTO-ST Institute, Univ. Bourgogne Franche-Comté, CNRS
Numérica, cours Leprince-Ringuet, 25200 Montbéliard, France
Email: firstName.lastName@univ-fcomte.fr

***Abstract*—Broadcasting is a widely used dissemination technique in wireless multi-hop networks. The simplest broadcasting technique, the pure flooding, is quite inefficient in terms of number of packets generated. Whereas numerous ways to optimize it in classical networks have been proposed, we found no proposition appropriate for very dense networks (nodes having thousands of neighbors or more) with resource-constrained nodes. Typically, nodes have to prevent the broadcast storm problem using only a partial or even no information about their neighborhood due to memory scarcity. At the same time, they have to avoid the die out problem commonly found in broadcasting techniques. In this article we propose an efficient broadcasting scheme for dense nanonetworks. It combines an estimator of the number of neighbors, a backoff window and a counter of packets. We explain why this method is efficient in dense networks. Based on simulation results, we show that on a random dense network it reduces by 71% the number of packets exchanged in the network compared to an optimized method found in the literature, while at the same time avoiding the die out problem.**

I. Introduction

Multi-hop networks usually rely on broadcasting as a way to convey information to all nodes in the network. In low density networks, i.e. where nodes have only tens or maybe hundreds of neighbors, the multi-hop broadcasting is relatively simple and has been extensively studied, with optimal or almost optimal solutions proposed. But in (very) dense networks the usual solutions have issues. Pure flooding has a prohibitive cost. Probabilistic flooding methods may work, but they need many packets exchanged in order to achieve a good coverage of the network. Moreover, counter-based methods (introducing a form of controlled redundancy) do exist in macro networks, but are not transposable as is in dense networks.

We define dense networks as networks with so many neighbors that the classical protocols handle common tasks inefficiently. For instance, the Wi-Fi DCF channel access method is already quite inefficient when 100 nodes need to send packets at the same time. The initial 32 slots backoff window [1] gives a high probability of collisions, and progressively increasing the backoff window up to 1024 time slots, even if eventually allows nodes to send their data, leads to a lot of wasted time. A second example is the classical problem of RFID tags counting. To handle large number of tags, usual methods are very ineffective. Methods coping with high number of neighbors have been proposed, involving either static parameters [2] or, better, dynamic ones [3].

978-1-5386-7659-2/18/$31.00

Nanonetworks are a recent incarnation of dense networks. They are formed of nodes of nanometric size, which we call nanonodes. Nanonetworks potential uses include programmable matter, advanced health monitoring, and wireless network on chip (WNoC). Due to their tiny size, the energy, memory and computation capacity are extremely scarce. Classical modulations techniques consume too much energy (if only for the carrier), and a specific TS-OOK modulation has been proposed [4], where a sender highly synchronized with the receiver sends an energy pulse as bit 1, and silence as a bit 0. In this context, a dense network is a network where nodes have at least a few hundreds neighbors (with many thousands still plausible).

Existing broadcast scheme for other dense networks, such as Vehicular Ad hoc NETworks (VANET) [5] and Wireless Sensors Networks (WSN) [6], are not applicable to nanonetworks. Nanonodes cannot use geolocation techniques which would allow them to find out the most suitable forwarder. Moreover, their small memory cannot contain the potentially huge list of their neighbors.

Last but not least, in nanonetworks, nodes do not sequentially access the channel. Different packets may overlap. In the example of packet counting (used in this article), nodes take an action depending on the number of copies received. In Wi-Fi and most macro scale protocols, the access to the channel is sequential [1]. Thus, each time a node is allowed to transmit, it knows precisely how many copies have already been sent. On the contrary, in nanonetworks, packets naturally overlap. At a given time, a node may not be aware of the copies already being sent (as they or at least their headers have not been fully received). This leads to nodes eventually receiving much more copies than intended, and is only one example of why current schemes are not suited to dense nanonetworks.

The contributions of this article are the following. We propose a method to efficiently broadcast information in dense networks. It combines a density estimator, a backoff window, and a packet counter. We explain why this method would be efficient in such networks. We analyze the efficiency of two parameters: the counter value (redundancy) and the backoff window size. Finally, we compare it with probabilistic flooding and pure flooding from two points of view, network coverage and packets exchanged, and show its superiority.

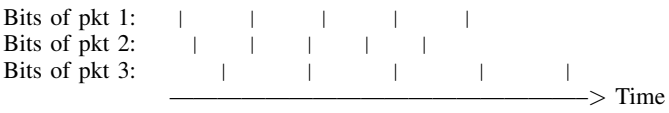

Fig. 1. In TS-OOK, packets may overlap by bit interleaving.

The article is organized as follows. Section II presents the background, and Section III presents related work. The flooding method we propose for dense networks is detailed in Section IV, and Section V shows and analyzes simulations results. Section VI draws the conclusions.

II. Background

A. Electromagnetic nanocommunications

A nanonetwork is a network composed of submicrometric nodes [7] with communication capabilities. Due to high signal attenuation and very low transmitting power, nanonodes are able to receive bits from only a few milimeters away [8].

Nanonetworks are a good example of potentially dense networks. For example, for a 3D isomorphic communication range of $r = 5\,\mathrm{mm}$ and nanonodes of volume $v = 10\,\mu\mathrm{m}^3$, a node can have up to $\frac{4\pi}{3} r^3 \frac{1}{v} \approx 5 \times 10^{10}$ neighboring nodes.

A modulation scheme (TS-OOK) has been proposed [4], which uses an electromagnetic pulse of duration T_p to transmit a bit "1", and silence (no transmission) for bit "0". The time between (starting of) two consecutive bits is T_s. Due to hardware and power constraints, the spreading ratio $\beta = T_s/T_p$ can be very large. Using T_p=100 femtosecond and $\beta = 1000$ [4], the total available bandwidth is very high, in the order of terabit per second.

Such a modulation scheme has a few peculiarities. One of them, useful for our study, is time multiplexing of packets: numerous packets are being transmitted *at the same time*, as shown in Fig. 1. Otherwise said packets overlap by bit interleaving. This is similar to TDMA (time-division multiple access), however, whereas in TDMA the multiplexing is done at communication level, in nanonetworks it is done at packet level. These simultaneous communications do not excessively rise the collision probability, and they could even be all correctly received at a given node, provided that bits do not collide. Note also that collision of two bits does not always leads to an error: an error occurs only when the receiver is receiving a "0" (silence) and at the same time a "1" arrives, which hides the "0"; the three other cases (receiving two silences, or receiving a "1" at the same time with a "0" or "1") do not lead to a transmission error.

Those specificities along with the simplicity of nanomachines that prevents using the usual TCP/IP network stack oriented us to develop a specific simulator that will be briefly presented at the beginning of section V.

B. Node density estimation

In our *backoff flooding* method, similarly to adaptive probabilistic flooding, nodes need to know the number of nodes found in their communication range. In low density networks, nodes could simply send a *hello* message containing their unique ID, and count the packets received. However, in dense networks, this would lead to numerous collisions and transmission errors. Moreover, a node with several thousands of neighbors does not have enough space to store a list of all its neighbor IDs. Thus, despite having several estimators proposed in the literature [9], [10], DEDeN (Density Estimator for Dense Networks) [11] is, to the best of our knowledge, the only efficient density estimator in dense networks.

DEDeN does not find the exact number of neighbors, but allows to set a precision and a confidence in the estimation, which in turn influences its cost (in terms of packets sent). This is very useful, since for the flooding protocols used in this paper for example, the estimation does not need to be very precise (a 30% error margin with a confidence of 95% leads already to very good results), in which case the cost of DEDeN is quite small. Also, DEDeN works with any range of neighbor densities.

III. Related work

A. Sequential medium access assumption

Most related works consider a 802.11-like medium access. Here, the channel can be used by at most one transmitting node at a time, i.e. when a node sends data, the other nodes have to wait, the transmission of a packet being an "atomic operation" [1]. Additionally, packets are sent sequentially, one at a time. On the contrary, in nanonetworks, packets from different nodes can overlap. Therefore, many methods found in the literature cannot be used in our context.

B. Pure flooding

The simplest method to broadcast information to all the nodes in a network is flooding. The initial packet is sent by the source node, and each node receiving the packet forwards a copy of it (we call it *forwarder*). This method generates an immense overhead in terms of packets exchanged, a deficiency known as the broadcast storm problem [12]. As such, it is rarely used.

C. Adaptive probabilistic flooding

In probabilistic flooding, the nodes that receive a packet forward it with a certain probability. This greatly reduces the number of forwarders. The probabilistic flooding remains memoryless: the decision about forwarding is taken as soon as the packet is received, and packets do not need to be saved in memory.

A fixed probability yields good results only for the corresponding neighbor density (for example, for a neighborhood density of 100, a probability of 10% means an average of 10 forwarders per communication range). Adaptive schemes automatically adapt the probability based on some information.

Numerous (adaptive) probabilistic flooding methods have been proposed [13], their difference being in the information (such as number of neighbors, node speed or energy) and the formula used to compute the probability. For the purpose of our article, a typical example is when the probability of

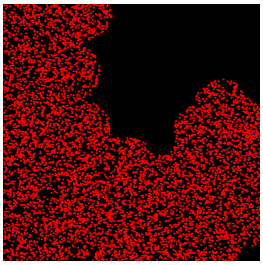

Fig. 2. Die out problem: incomplete broadcasting (black zone on top-right).

forwarding is set to k/n, with k an efficiency parameter to increase the reachability of the broadcast to the detriment of method overhead (number of packets sent), and n the number of neighbors [14]. In the results section we compare our method to this one.

All these methods have a common drawback, which lies in the use of probabilities: they do not insure that a packet will eventually be forwarded to all the nodes in the network. For example, in a zone with 100 neighbors and probability of 5%, usually 5 nodes will forward the packet, but it also could happen that *no* node chooses to forward the packet. This can lead to the *die out problem*, which appears when the packet does not reach some zones in the network, as shown in Fig. 2, plotted with our simulator.

D. Adaptive counter-based schemes

To avoid the die out problem, in adaptive counter-based schemes, nodes count the number of copies of a given packet to decide whether to forward the packet or not. Numerous methods of this family have been proposed [13].

For instance, GOSSIP3 [15] uses a fixed forwarding probability. If the decision taken is to not forward, the node waits a given time, and if it has not seen a given number of copies, calculated from the forwarding probability and the number of neighbors, it will still forward it. In AGAR [16], the forwarding mechanism is split in two phases. In the first phase nodes decide to forward or not a packet with a fixed probability. Nodes that have decided to not forward the packet wait, in the second phase, for a given time; if the number of copies received is below a given threshold, they will forward the packet with a probability depending on the initial probability and the number of neighbors.

The adaptive scheme proposed in [17] uses the number of neighbors to compute the required number of forwarders. In very sparse networks, several forwarders are required, because nodes need to cover large areas. On the contrary, in the nanonetwork context, as we will see later, because of higher node density, fewer forwarders are needed, and one will usually be enough.

Moreover, in 802.11-like medium access, as used in [17], the backoff window used for *broadcasted* packets is fixed and small. This is not appropriate in nanonetworks, where the number of neighbors varies widely, and as such we will dimension the backoff window before forwarding using the number of neighbors.

E. Geoforwarding and OLSR

Geoforwarding protocols are protocols that rely on geographic information about node positions to take the forwarding decision [18]. Geoforwarding is often used in WSN to select the furthest possible forwarder and then optimize the coverage and reduce the number of forwarders. Node positions can be obtained:

- directly through a GPS embedded in each node;
- by triangulation techniques, using relative positions of nodes obtained using anchors, beaconing or techniques relying on signal strength [19].

Node positioning methods do not fit the nanonetwork requirement. Due to their small size and low energy available, nanonodes can neither embed a GPS, nor compute their position without an underlying infrastructure.

As mentioned before, nanonetworks may be extremely dense. This huge amount of neighbors and the low memory available on nanonodes make useless all kinds of protocols that rely on detailed neighborhood knowledge or routing table, such as OLSR [20] and its variants.

F. Clustered networks

Hierarchical (or clustered) routing protocols are often used in WSN [21]. The network is split into several clusters, in which a node assumes the role of leader, commonly called *cluster head*. Cluster heads are similar to routers in IP networks, in that the information generated by nodes is routed by them, generally towards a gateway or a sink.

Clusterization is not suitable to nanonetworks for several reasons. First, additional protocols are needed, for cluster head election for example. Also, given that cluster heads are points of failure, such a system needs to include techniques to deal with fault tolerance. Moreover, since cluster heads treat data from the nodes in their region, plus data from other clusters heads, they consume much more energy, and need more resources, such as higher computation capacities and much more memory (for waiting queues), than the other nodes. This does not meet the requirements of nanonodes.

To conclude, classical broadcasting methods are not applicable to nanonetworks. Moreover, since nodes have low memory and processing capability, they cannot maintain a complete image of their neighborhood, useful to find optimal forwarders. Maintaining even a one-hop neighbor list becomes indeed excessively costly when neighbors are of the order of thousands. Therefore, we propose a new adaptive density- and counter-based method, detailed in the next section.

IV. Backoff flooding

Backoff flooding works as following. When a node receives a packet for the first time, it starts a time counter. If it has received a given number of copies of the packet before the timeout, it simply discards the packet. Otherwise, at the end of the period, it forwards (by broadcast) the packet and discards it. In both cases, it memorizes the ID so that it does not treat

again a copy of that packet. Note that nodes never *re*transmit a packet (i.e. send multiple copies).

To operate, this method combines a counter of packets, a waiting window and a density estimator.

Let us recall the importance of the counter of packets. Depending on the network density and communications, the broadcast propagation may lead to problems. For example, packets may be lost due to collisions. Also, because multi-hop broadcasting methods tend to randomly select forwarders, non optimal ones may be chosen, and in the worst case a forwarder may not forward the packet to new nodes, effectively halting its propagation. To avoid this, backoff flooding uses a *redundancy* (packet counter) parameter, whose role is to guarantee that a message is forwarded at least a given number of times. It will be analyzed in the simulation section.

The estimator, discussed in Section II-B, is used in the waiting window. Finally, the waiting window is described in the next section.

A. Waiting window size

An essential part of our method is the use of a carefully-sized backoff window. Let us recall that because of the high temporal multiplexing capability of TS-OOK modulation, many copies of a packet can start being sent before nodes decode them. This issue can be solved by adding a waiting time to packet forwarding, commonly called *backoff*, or *RAD* (Random Assessment Delay) in [13], which also helps to drastically reduce collisions.

It was stated in [13] that "there is no a clear evaluation of the role played by RAD. There is no evaluation on the optimal value of RAD in counter-based schemes." Here, we consider a RAD chosen uniformly between 0 and t. Intuitively, the selection of the upper bound t has the following impacts:

- A small value of t when the local nodes density is high means a high probability of concurrent copy transmissions, and even collisions.
- A large value of t means a uselessly long backoff period, hence a higher end to end delay in the transmission.

Because of this, we decided to tie t to the local nodes density, obtained through an estimator, as presented in section II-B. Contrary to classical backoff, where the average real delay added at each hop is the average between 0 and the window size, in our case the packet is forwarded with the *lowest* backoff drawn.

To find out t, let $t_{dmax} = \frac{c}{cr}$ be the time for a bit to reach the edge of the communication range (c is the speed of light and cr is the communication range), $t_{pkt} = T_s \times s$ the transmission time of a packet (T_s is the time between two consecutive bits, cf. Section II, and s is the number of bits of the packet), i.e. the time elapsed between the sending of the first and of the last bit of the packet, and t_{proc} the processing time required to decode the packet and decide to forward it. Without backoff, the time after which a node can be sure that a neighboring node has forwarded a packet is:

$$t_{wait} = 2(t_{dmax} + t_{pkt} + t_{proc}) \quad (1)$$

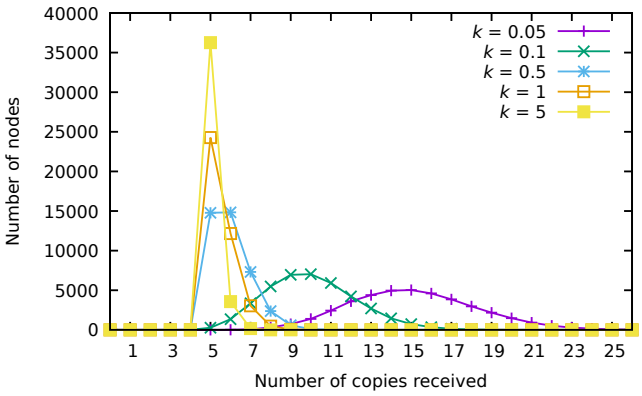

Fig. 3. Distribution of the number of copies received due to packet multiplexing for an intended redundancy of 5 and various values of k: for large values of k, almost all packets are received exactly 5 times per node; for small values of k, packets are often received 6 times or more.

which accounts for the packet and its forwarding copy to go back and forth to the furthest node in the communication range. It should be noted that t_{proc} can be treated as a constant and does not impact the behavior of the protocol. Then the ideal maximum backoff t is:

$$t = k * n * t_{wait} \quad (2)$$

where n is the estimated number of neighbors given by the estimator and k is a multiplying factor affecting the distribution of the number of forwarders, studied in the following.

B. k multiplying factor effect on the number of forwarders

As specified above, the backoff value depends on the number of neighbors in order to avoid concurrent forwarding copy transmissions and collisions. It should be noted however that, fundamentally, backoff flooding does not guarantee an exact number of forwarders, for two reasons: packet overlapping and geometry of node positions.

To see the influence of the first reason on the number of forwarders, let us recall that backoff flooding involves a random number, potentially allowing several nodes to choose very close backoff values. Due to nanocommunication specificities (collision and packet multiplexing), multiple copies of a given packet may be sent simultaneously. The larger k, the smaller the probability to have concurrent copies being transmitted. This is illustrated in Fig. 3 for an intended redundancy of 5, 1150 neighbors, $t_{wait} = 8$ nanoseconds, and for various values of k. In this scenario, nodes ideally should receive exactly 5 copies of a given broadcasted packet. But with small k, it happens frequently that the reception of a previous copy is not finished (and thus still ignored) by the moment a node decides to forward another copy. The end result is that nodes often receive far more than 5 copies.

The number of copies received by nodes increases further because of the geometry of the area covered by each copy transmission. The phenomenon is illustrated in two dimensions on Fig. 4 for a redundancy of 1. An omnidirectional communication radius is considered for an original transmitter T

(Fig. 4a). Among all the nodes in the reception area, the one that picked the shortest random backoff duration ($R1$) forwards the packet (Fig. 4b). The nodes in the darker area having received a copy, they cancel their scheduled copy transmissions. Among the nodes touched by the initial transmission, a crescent has not yet received a copy. A node $R2$ in this crescent will then reach the end of its backoff period and sends a new copy (Fig. 4c). By doing so, part of the nodes will get a second copy, while the area that has not yet seen a copy will be reduced. A third transmitter $R3$ sends a copy, finally ensuring that all nodes in the initial transmission area have seen a least one copy (Fig. 4d). As the message further propagates, it can easily be seen on Fig. 4e that most nodes will have received more copies than intended.

The influence of both reasons mentioned above can be seen in a simulation (using the BitSimulator software presented in the next section) to count the effective number of copies received by nodes for various values of k. As previously, the average neighborhood is 1150 nodes, the intended redundancy is 5, and $t_{wait} = 8$ nanoseconds. DEDeN provided the neighborhood estimation with a 95% confidence to have a 30% maximum estimation error. Fig. 5 presents the results. Compared to Fig. 3, the curves are flatter (maximum is around 9000, compared to around 36 000), the reason being that the geometry is taken into account too. Similarly, the required redundancy is always met (5 copies at least). Also, on the one hand, the values of k smaller than 0.5 induce an increase in the number of copies transmitted. On the other hand, big values of k increase backoff window size, cf. (2), and as such increase delay. Consequently, from now on, we will use $k = 0.5$ as it represents a good tradeoff between number of copies and delay.

C. Delay incurred

In backoff flooding, upon first reception of a packet, a node chooses a backoff value inside the backoff window t. The packet is delayed either until that backoff (when the packet is forwarded), or until it receives *redundancy* copies of the packet (and the packet is discarded), whichever occurs first. During that time, the packet needs to be stored in node's memory.

In the following we analytically compute this delay for a redundancy of 1. We model this time as being the minimum value among n values (n is the number of neighbors) drawn at random in a window of size t femtoseconds.

The probability for a node to draw a backoff b superior to an arbitrary value $v \in [0, t[$ (with b and v integers) is:

$$P(b > v) = \frac{t - v}{t} \tag{3}$$

On the other hand, the event described by "the minimum backoff drawn is superior to v" is equivalent to the event "the backoffs drawn by the n nodes are superior to v". As such, the probability of the latter event is given by:

$$P(b_{\min} > v) = (P(b > v))^n = \left(\frac{t - v}{t}\right)^n \tag{4}$$

where $b_{\min}$ is the mimimum backoff drawn. Then, the probability for $b_{\min}$ to be inferior to v is:

$$P(b_{\min} \leq v) = 1 - P(b_{\min} > v) \tag{5}$$

The probability for $b_{\min}$ to be in an interval $]v_1, v_2]$ can be computed from (4) and (5):

$$\begin{aligned} P(v_1 < b_{\min} \leq v_2) &= P(b_{\min} \leq v_2) - P(b_{\min} \leq v_1) & (6)\\ &= \left(1 - \frac{v_1}{t}\right)^n - \left(1 - \frac{v_2}{t}\right)^n & (7) \end{aligned}$$

Fig. 6 shows the probability for the minimum backoff to be in a 1% interval of the backoff window, i.e. between $xt/100$ and $(x+1)t/100$. This probability is shown for several densities (10, 100 and 1000 neighbors) and hence for several backoff window sizes, since the backoff window's length directly depends on the number of neighbors, as given in (2). Fig. 6 shows, for instance, that for $n = 100$ the probability for $b_{\min}$ to be inferior to 3% of t is greater than 0.95. Looking at (2), 3% of t is a small value, at the same order of magnitude than back and forth time t_{wait}. Additionally, the greater the density, the smaller the part of the backoff window used by $b_{\min}$.

D. Memory usage (amount of time packets are stored in nodes' queue)

In backoff flooding, a node that receives a packet has to store it until it receives a given number of copies of the packet, but without exceeding the duration of the backoff. We are interested in the amount of time this packet is stored in node because, due to their tiny size, nanonodes have a small amount of memory. Nodes also memorize the IDs of the broadcasted packets, so that they do not treat them again; however, we neglect this, given that only an ID is memorized and only for a short time.

We formalize the amount of time as following. We need to compute for any number of participating nodes n, any desired redundancy r, and any value $v \in [0, t]$, the probability P of the event "r values are in the interval $[0, v[$ and the others $n - r$ values are in the interval $[v, t]$".

Each of the n values can be either strictly inferior, or superior to v. This means that there 2^n different cases. Among them, $\binom{n}{r}$ cases have r values strictly inferior to v. Now, the cases are not equiprobable: the probability of a case is the product of the probability of each value to be inferior or superior to v. The probability p for a value to be strictly inferior to v is v/t, and to be superior to v is $1-v/t$. Therefore, the probability for each of the $\binom{n}{r}$ cases is $p^r \times (1-p)^{n-r}$. From that we can finally compute P as:

$$\begin{aligned} P &= \binom{n}{r} \times \left(p^r \times (1-p)^{n-r}\right) & (8)\\ &= \frac{n!}{r!(n-r)!} \times \left(\frac{v}{t}\right)^r \times \left(1 - \frac{v}{t}\right)^{n-r} & (9) \end{aligned}$$

Fig. 7 shows (9) for $t = 199\,804$ ns, $n = 1000$, redundancies r from 2 to 7, and various values of v. For these parameters, nodes very rarely need to store the packet for

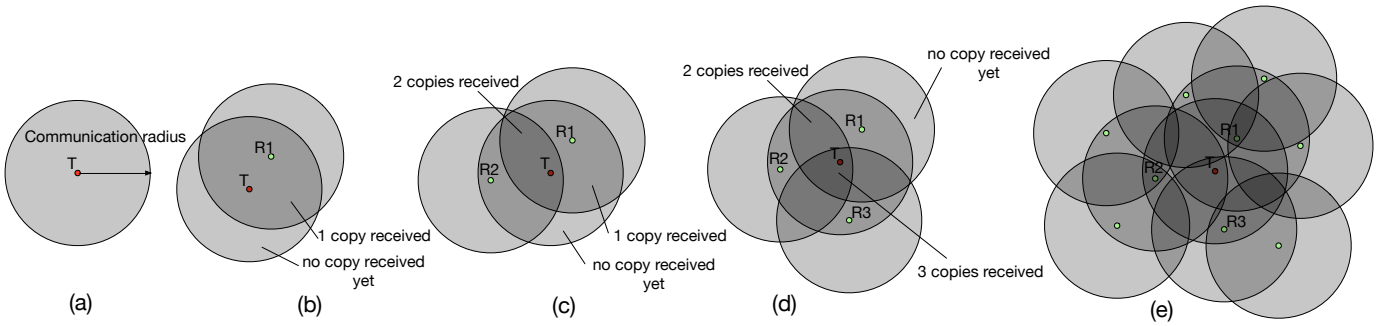

Fig. 4. The number of copies received with backoff flooding depends on node positions.

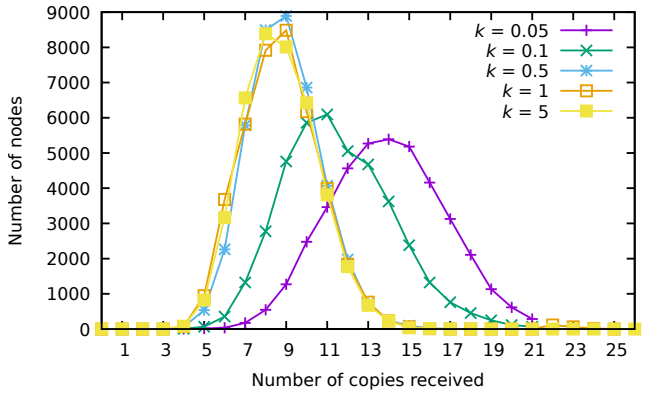

Fig. 5. Distribution of the number of copies received due to packet multiplexing *and* geometry for an intended redundancy of 5 and various values of k.

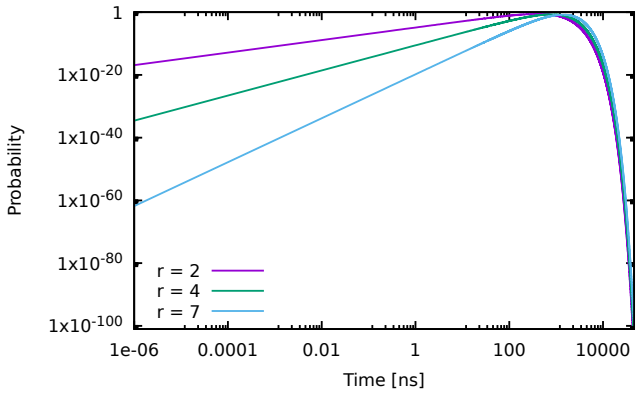

Fig. 7. Probability for r values to be smaller than time v in a window of 199 804 ns.

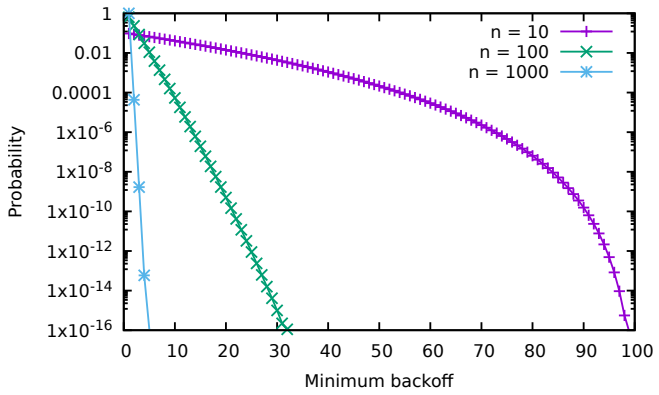

Fig. 6. Probability of $b_{\min}$ to be in a 1% interval of the backoff window t.

a time greater than 800 ns. After this value, the probability decreases abruptly: for example, the probability for four values ($r = 4$) to be smaller than 800 ns (which is approximately 0.4% of t) is 0.19, while the probability to be smaller than 1600 ns is 0.05.

Fig. 7 shows that the curves are not monotonic. To explain this, let us take the example of two values ($r = 2$). For small v (close to 0), the probability to have the two values smaller than v increases with v. Similarly, for big v (close to t), the probability to have $n - 2$ values greater than v (i.e. 2 values smaller than v) increases with the decrease of v. The increasing curves both from 0 and from t explain the non monotonicity of the curves.

To resume, for a redundancy of 4 and a backoff window of 199 804 ns, in 95% of cases the packet is stored in node's memory for at most 1600 ns, i.e. 0.8% of the window.

V. Simulation

A. BitSimulator overview

Many technological issues need to be solved before building nanomachines and nanonetworks. Also, even if they existed, it would be practically impossible to create scenarios involving a very large number of nodes. Therefore, we cannot carry out experiments to test our proposed method, and we use simulations.

Currently, four nanonetwork simulators exist:

- *Nano-Sim* [22] is a plug-in for the well-known NS3 network simulator[1]. It does not take into account the signal propagation delay, neither the actual payload when computing collisions, whose impacts are significant in nanonetworks.

[1]https://www.nsnam.org

- *Vouivre* [23] is a discrete event simulation library which simulates individual packets and propagation delay. However, it uses a statistical model for packet error rate and does not take actual payload into account.
- *COMSOL Multiphysics*[2] is a simulator for various physics and engineering applications, and as such it allows to simulate THz nanocommunication too. This very precise simulator would take too much time to simulate the large networks used in our tests.
- *BitSimulator* [24], a small and very focused simulator developed by us. It uses a discrete event model and has the following main features useful for our study:
  - Propagation delay: packet arrival time on a node depends on the distance between the sender and the receiver.
  - Collisions: the TS-OOK model (see the background section) and packet payload are used to compute collisions.
  - Its simple design allows it to scale up to hundred of thousands of simulated nodes on a usual laptop.
  - The simulations use reproducible random numbers, giving reproducible simulations.

For our work, BitSimulator is the most appropriate, and as such we used it in this article.

B. Scenarios and parameters

In all the simulations we use the following scenarios. We simulate a 2D environment as a square of side $s = 6\,\text{mm}$. We use an all-or-nothing propagation model with a communication range of $cr = 0.5\,\text{mm}$. Packets are 1000 bits long, $\beta = 1000$, and $T_p = 100\,\text{fs}$ [4]. Nodes are always placed randomly. A node placed in the center of the network broadcasts a packet to all the nodes in the network.

Given the environment size and the communication range, the furthest possible receiver is in the corner, at $\sqrt{2} \times s/2 = 4.2\,\text{mm}$ away from the source, i.e. $4.2/cr = 8.4$ times the communication range, leading to 8 hops (or more, depending on the connectivity of the network) for the packet to reach the furthest possible node. The average density of nodes in a communication range (number of neighbors) can be simply computed as (n_b is the total number of nodes):

$$density = n_b \frac{\pi cr^2}{s^2} = 0.022 \times n_b \tag{10}$$

In the following we compare three methods: our method with DEDeN, probabilistic flooding with DEDeN, and pure flooding. Instead of DEDeN, one can choose to use another estimator, as presented in the background section.

We define as the probabilistic flooding with DEDeN the method presented in [14], i.e. the flooding with the following forwarding probability:

$$p = \frac{r}{n_e} \tag{11}$$

[2]https://www.comsol.com

where $r \leq n_e$ is the intended average number of forwarders (i.e. the redundancy), and n_b the estimation of the number of neighbors for each node, as provided by DEDeN.

The pure flooding and the probabilistic flooding do not work well in nanonetworks for the following reason. When a node broadcasts a packet, because of the huge speed of the light c, all its neighbors receive it almost at the same time, more precisely in an interval of time kT_p, with k small (depending on the communication range), and $T_p = 100\,\text{fs}$ as the duration of a pulse, i.e. of a bit, cf. the background section. More specifically, $k = cr/(cT_p)$, which in our scenario gives $k = 0.5 \times 10^{-3}/(3 \times 10^8 \times 100 \times 10^{-15}) = 16.5$. In pure/probabilistic flooding and dense network, all these forwarding neighbors forward this packet as soon as they receive it, i.e. in a very small window, of k bits, which generates numerous collisions at receivers. To avoid this disastrous effect, we have added a backoff window of 10 000 fs to pure and probabilistic flooding.

The metrics used in the comparison are the reachability and the cost. We define the reachability as the ratio of nodes having received the packet divided by the number of nodes in the network. A reachability of 1 means that all the nodes in the network have received the packet. As such, a disjoint network never achieves a reachability of 1. We define the cost as the total number of packets sent in the network divided by the number of nodes in the network. Our goal is to have a high reachability with a small cost.

In the following, a redundancy of n means that the node will forward the packet if and only if it has received at most n copies. Otherwise said, when a node receives $n + 1$ copies during the backoff, it discards the packet.

In the following, each point represents the average of 15 simulations, which differ only by the seed of the random generator used for the probability in probabilistic flooding, and for the backoff value inside the given window for backoff flooding, which results in different forwarders. We consider this to be sufficient, given the high number of nodes, and the size of the network compared to the communication range.

C. Results

We simulate 10 000 nodes, which, according to (10), gives an average density (number of neighbors) of 218, which in turn means that all the nodes are certainly reachable by the sender. The results are shown in Fig. 8.

Since the backoff flooding insures that at least *redundancy* packets are sent in a given communication range, a redundancy of 1 should be sufficient to obtain a reachability of 1; simulations confirm it: for a redundancy varying from 1 to 20, reachability is always 1, i.e. all the nodes receive the packet.

For probabilistic flooding, the reachability depends heavily on the redundancy, and for low redundancies the reachability is quite low. Also, for low redundancies, a large variability can be noticed: the reachability can span as much as from close to 0 to close to 1 (for redundancy of 5 in Fig. 8). The reachability close to 0 in one of the simulations corresponds to the fact that the initial packet, for probabilistic reasons, has

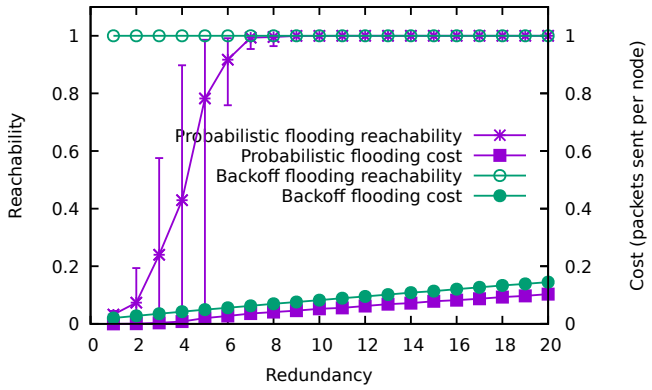

Fig. 8. Reachability and cost of probabilistic flooding and backoff flooding in dense networks. Pure flooding, not shown in the figure, has both reachability and cost equal to 1.

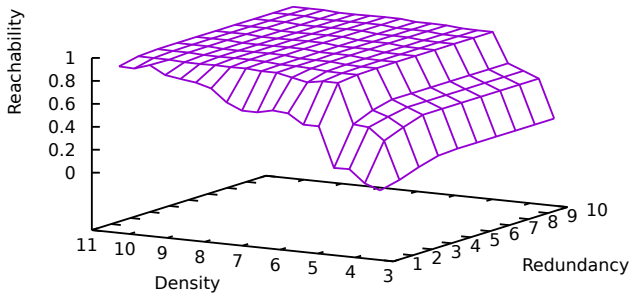

Fig. 9. Reachability in low density networks.

not been forwarded by any node, hence only the nodes in the initial communication range received it. In general, the die out problem (a reachability smaller than 1) appears when, for probabilistic reasons, all the nodes of a part of the network happen to not forward the packet, leading to the whole further region being dead out. This phenomenon does not appear in backoff flooding, because it is not probabilistic.

Another advantage of backoff flooding is that, given that a redundancy of 1 is sufficient, it is automatic, i.e. there is no parameter to tune for reachability, whereas in probabilistic flooding one has to discover which redundancy should be used to achieve a redundancy of 1.

As for the cost, for probabilistic flooding, the number of packets is proportional to the forwarding probability, which in turn is, according to (11), proportional to the redundancy. For backoff flooding too, the redundancy gives the number of packets in a communication range, hence the latter is proportional to the former. The simulations confirm these results: Fig. 8 shows that the number of packets increases linearly with the redundancy for both probabilistic and backoff flooding methods. Moreover, for a reachability of 1, the cost of backoff (obtained for a redundancy of 1) is smaller than the cost of probabilistic flooding (required redundancy of 9).

As for the pure flooding, it always achieves the same result: a reachability and a cost of 1. The small backoff we added to it reduces the number of collisions and allows all nodes to receive at least one copy of the packet, i.e. a reachability of 1. The reason for the constant cost of 1 is that all the nodes receiving a packet for the first time forward it. Compared to it, backoff flooding also gives a reachability of 1, but using much fewer packets.

We also simulated a network of 100 000 nodes, hence a density 10 times higher, i.e. 2180 neighbors. The conclusions on reachability and cost were similar to the dense network, hence we do not show them. Additionally, we noticed that the number of packets sent is similar in 10 000 and 100 000 nodes scenarios, even if the number of nodes differs by one order of magnitude. Hence, the higher the density, the higher the efficiency.

Both probabilistic flooding and backoff flooding drastically reduce the number of copies generated compared to pure flooding. However, backoff flooding has two advantages over probabilistic flooding, and two drawbacks. First, it is much less prone to incomplete coverage because the forwarding decision is *not* probabilistic, and if a given redundancy is needed then it insures it. Secondly, using k parameter, it can be tuned to have a low redundancy variation. On the other hand, it incurs an additional delay that increases with the reduction of the number of forwarders and the spread of their distribution. As the decision to forward or delete a packet needs to be postponed, it also requires to keep one copy of the packet in memory during that time. This additional time is however small, cf. section IV-C.

To conclude, in dense and very dense networks, backoff flooding is better than probabilistic and (obviously) pure flooding. To achieve a reachability of 1 in dense networks, pure flooding has a cost of 1, probabilistic flooding a cost of 0.068 after choosing carefully the redundancy parameter, and backoff flooding a cost of 0.020 (71% fewer messages than probabilistic flooding), as shown in Fig. 8. In denser networks, backoff flooding has a cost of 0.0021 and probabilistic flooding a bigger cost, of 0.0063.

Even if our method does not target low density networks, since e.g. pure flooding should have satisfactory results, we are interested to know the reachability given by our method in this case.

We simulate a total number of nodes varying from 150 to 500, which, according to (10), represents an average density (number of neighbors) between 3 and 11.

We notice (Fig. 9) that the reachability is still 1 or close to 1 for densities larger than 9.26 with a redundancy of 1. For densities from 9.26 to 5.13, a rendundancy of 2 is necessary to obtain a reachability close to 1. For even smaller densities, less than 5.13, a redundancy of 1 gives bad results and a redundancy of 10 gives subunit results; for a density of 3.76 for instance, the reachabilities are 0.15 and 0.6, respectively.

The reason for the subunit reachability is that for low

densities the network could be disjoint. For a density of 3.76, with a redundancy of 10, which is bigger than the number of neighbors (3.76), the reachability is still 0.6, which corresponds to the maximum number of nodes which can be reached by the sender, given the "sparsity" of the network.

To conclude, in low density networks, the maximum reachability could be obtained only by increasing the redundancy, and for very low densities not all the nodes receive the packet because the network is disjoint.

VI. Conclusion

This article presented backoff flooding, an efficient broadcasting scheme for dense networks. It combines a backoff window, a counter of packets and an estimator of the number of neighbors. We explained why this method should be efficient in dense networks. We run simulations to confirm this statement. Also, compared to probabilistic flooding, the proposed method is automatic, i.e. there is no parameter for the application to tune, and needs much fewer messages exchanged in the network. For instance, in a network with a density of 218 neighbors, it uses 71% fewer messages. While being efficient, it avoids the die out problem commonly found in broadcasting schemes.

Future works include further improving backoff flooding by incorporating techniques to enhance forwarders selection, and measuring how the improvements given by backoff flooding help to reduce congestion in networks.

Acknowledgment

This work has been funded by Pays de Montbéliard Agglomération.

References

[1] M. Gast, *802.11 Wireless Networks: The Definitive Guide*. O'Reilly, Apr. 2002.

[2] Y. Zheng and M. Li, "Towards more efficient cardinality estimation for large-scale RFID systems," *IEEE/ACM Transactions on Networking*, vol. 22, pp. 1886–1896, Nov. 2013.

[3] B. Li, Y. He, and W. Liu, "Towards constant-time cardinality estimation for large-scale RFID systems," in *44th International Conference on Parallel Processing (ICPP)*. Beijing, China: IEEE, Sep. 2015, pp. 1–10.

[4] J. M. Jornet and I. F. Akyildiz, "Femtosecond-long pulse-based modulation for Terahertz band communication in nanonetworks," *IEEE Transactions on Communications*, vol. 62, no. 5, pp. 1742–1753, May 2014.

[5] F. Cunha, L. Villas, A. Boukerche, G. Maia, A. Viana, R. A. F. Mini, and A. A. F. Loureiro, "Data communication in VANETs: Protocols, applications and challenges," *Ad Hoc Networks*, vol. 44, pp. 90–103, Jul. 2016.

[6] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer Networks*, vol. 52, pp. 2292–2330, Aug. 2008.

[7] I. F. Akyildiz and J. M. Jornet, "Electromagnetic wireless nanosensor networks," *Nano Communication Networks*, vol. 1, no. 1, pp. 3–19, 2010.

[8] J. M. Jornet and I. F. Akyildiz, "Channel modeling and capacity analysis for electromagnetic wireless nanonetworks in the terahertz band," *IEEE Transactions on Wireless Communications*, vol. 10, no. 10, pp. 3211–3221, 2011.

[9] M. Cattani, M. Zuniga, A. Loukas, and K. Langendoen, "Lightweight neighborhood cardinality estimation in dynamic wireless networks," in *13th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. Berlin, Germany: ACM/IEEE, Apr. 2014, pp. 1–11.

[10] N. Riga, I. Matta, and A. Bestavros, "DIP: Density Inference Protocol for wireless sensor networks and its application to density-unbiased statistics," University of Boston, MA, USA, Tech. Rep. 2004-023, May 2004.

[11] T. Arrabal, D. Dhoutaut, and E. Dedu, "Efficient density estimation algorithm for ultra dense wireless networks," in *27th International Conference on Computer Communications and Networks (ICCCN)*. Hangzhou, China: IEEE, Jul.-Aug. 2018, pp. 1–9.

[12] Y.-C. Tseng, S.-Y. Ni, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," *Wireless Networks*, vol. 8, no. 2–3, pp. 153–167, Mar. 2002.

[13] D. G. Reina, S. Toral, P. Johnson, and F. Barrero, "A survey on probabilistic broadcast schemes for wireless ad hoc networks," *Ad Hoc Networks*, vol. 25, pp. 263–292, Feb. 2015.

[14] J. Cartigny and D. Simplot, "Border node retransmission based probabilistic broadcast protocols in ad-hoc networks," *Telecommunication Systems*, vol. 22, no. 1–4, pp. 189–204, Jan. 2003.

[15] Z. J. Haas, J. Y. Halpern, and L. Li, "Gossip-based ad hoc routing," *IEEE/ACM Transactions on Networking*, vol. 14, pp. 479–491, Jun. 2006.

[16] Z. Shi and H. Shen, "Adaptive gossip-based routing algorithm," in *23rd IEEE International Conference on Computer Communications*. Phoenix, AZ, USA: IEEE, Apr. 2004, pp. 1–4.

[17] Y.-C. Tseng, S.-Y. Ni, and E.-Y. Shih, "Adaptive approaches to relieving broadcast storms in a wireless multihop mobile adaptive approaches to relieving broadcast storms in a wireless multihop mobile ad hoc network," *IEEE Transactions on Computers*, vol. 52, no. 5, pp. 545–557, May 2003.

[18] N. Abu-Ghazaleh, K.-D. Kang, and K. Liu, "Towards resilient geographic routing in WSNs," in *1st ACM international workshop on Quality of service and security in wireless and mobile networks (Q2SWinet)*. Montreal, Canada: ACM, Oct. 2005, pp. 71–78.

[19] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less low-cost outdoor localization for very small devices," *IEEE Personal Communications*, vol. 7, pp. 28–34, Oct. 2000.

[20] T. Clausen and P. Jacquet, "Optimized link state routing protocol (OLSR)," Oct. 2003, RFC 3626.

[21] M. Aslam, N. Javaid, A. Rahim, U. Nazir, A. Bibi, and Z. A. Khan, "Survey of extended leach-based clustering routing protocols for wireless sensor networks," in *14th IEEE International Conference on High Performance Computing and Communication and 9th IEEE International Conference on Embedded Software and Systems*. Liverpool, UK: IEEE, Jun. 2012, pp. 1–8.

[22] G. Piro, L. A. Grieco, G. Boggia, and P. Camarda, "Nano-Sim: simulating electromagnetic-based nanonetworks in the network simulator 3," in *6th International ICST Conference on Simulation Tools and Techniques (SimuTools)*. Cannes, France: ACM, Mar. 2013, pp. 203–210.

[23] N. Boillot, D. Dhoutaut, and J. Bourgeois, "Going for large scale with nano-wireless simulations," in *2nd ACM International Conference on Nanoscale Computing and Communication (NanoCom)*. Boston, MA, USA: ACM, Sep. 2015, pp. 1–2.

[24] D. Dhoutaut, T. Arrabal, and E. Dedu, "BitSimulator, an electromagnetic nanonetworks simulator," in *5th ACM/IEEE International Conference on Nanoscale Computing and Communication (NanoCom)*. Reykjavik, Iceland: ACM/IEEE, Sep. 2018, pp. 1–6.