Efficient Density Estimation Algorithm for Ultra Dense Wireless Networks

Thierry Arrabal, Dominique Dhoutaut, Eugen Dedu

Efficient Density Estimation Algorithm for Ultra Dense Wireless Networks

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*—Extremely dense wireless network topologies gradually become a reality, especially through wireless sensors networks and more recently nanonetworks. Electromagnetic nanonetworks are expected to allow a very large amount of extremely small and capability-limited devices to communicate with each others. In nanonetworks, even in a communication range limited to tens of centimeters, thousands of neighbors can be found. Information diffusion and routing protocols would greatly benefit from having an accurate estimation of the density of nodes. However, in this context, most traditional wireless communication protocols are not suited. We propose Density Estimator for Dense Networks (DEDeN), a distributed algorithm able to provide the required density estimation. It allows confidence tuning and can cope with an extreme range of local densities. A formal analysis of DEDeN is provided and corroborated by extensive simulations. DEDeN interest is then demonstrated through application to two information diffusion protocols tailored for very dense networks, and also to a routing protocol specific to nanonetworks.**

I. Introduction

Continuous progress in miniaturization of electronic and mechanical components has led to machines composed of nanometer size elements. Using nano CPUs, nanomemories, nanosensors, nanoactuators, machines measuring up to a few micrometers can be built. But such tiny machines have a very low computation capacity and not enough power to handle common tasks alone [1].

A network of nanomachines, called a nanonetwork, is much more capable. Electromagnetic nanonetworks have been a field of interest for a few years now. Given the limited size of nanomachines, graphene nano-antennas have been proposed, which radiate in the THz band. The characteristics of this band and the very low energy found in nanomachines allow only a very short communication range compared to macro scale systems, in the order of centimetres or tens of centimetres [2]. The communication range can be further increased by using multi-hop nanonetworks, so they can achieve more complex or spread out tasks. As such, nanonetworks can be very large compared to the small size of the devices composing them. Those nanonetworks could be used to design a new generation of applications in fields of importance such as biomedical, environment or military [3].

Some applications envisioned for nanonetworks, such as programmable matter [4], can require a high nodes density over a large surface. Neighborhood cardinality can largely exceed thousands, while the diameter of the network (in terms of hops) can still be high. These values are well above those of usual radio networks.

The communication in a dense (nano)network has some peculiarities. Routing and information broadcasting have to be done while being very constrained in memory and processing power. Simple, pure flooding protocols do not work because of the extremely high redundancy and channel usage they cause. More elaborate routing protocols are often limited by the individual memory capacity that prevents a node to even maintain a lists of its neighbors.

The contribution of this article is to present DEDeN (Density Estimator for Dense Networks), an algorithm tailored for wireless nanonetworks that allows a node to estimate the number of its neighbors (also called node degree, node density, neighbor density, neighborhood density, local density, or simply density). DEDeN has the remarkable property to be tunable. The confidence and the error range of the estimation can be adjusted to the requirements of the user with a predictable overhead. Depending on how it is initiated, it enables a unique node to estimate its neighborhood cardinality, or propagate to the whole network and allow all nodes to get an estimation in one, more efficient pass. The algorithm may be executed each time this estimation is needed, and in particular during network setup (before using it). It has a tiny memory footprint: only a counter is needed. Simulations show that it gives high quality estimations with few exchanged messages, much fewer than a pure hello method. Our density estimator can cope with low density networks, but is the first proposed algorithm well suited to estimate the density of a very dense nanonetwork (thousands of 1-hop neighbors).

The remaining of this paper is organized as follows. Section II introduces nanonetworks and their specificities. Section III presents related studies. Section IV explains our algorithm and its design choices. Section V presents *BitSimulator*, a powerful tool we developed to validate our theoretical studies. Section VI experiments with our algorithm and analyses results. Section VII shows how upper layers broadcasting or routing protocols can benefit from using DEDeN. Section VIII concludes the paper.

II. Background: Communication in nanonetworks

At nano size, electromagnetic waves may be produced by graphene nano-antennas [3], [5]. Despite their extremely small size and in contrast to similarly sized classical metallic

antennas, graphene nano-antennas are expected to resonate in the terahertz band, between 0.1 and 10 THz. The THz band behaves differently from standard wireless channels like 2.4 GHz and 5 GHz used for Wi-Fi and many others macro-scale communication standards.

Therefore, for nanonetworks, Jornet at al. proposed TS-OOK, a pulse based modulation scheme [6]. TS-OOK uses pulses to transmit information over the channel. "1" bits are encoded with a power pulse of duration T_p and "0" bits are encoded as silence. Because sending consecutive pulses need a hardware and power unavailable at such small sizes, consecutive bits are spaced with a duration T_s which is usually much longer than the pulses themselves. The spreading ratio $\beta = T_s/T_p$ can be very large (1000 for instance, and possibly much more). Because pulses are extremely short—Jornet and al. proposed a T_p=100 femtosecond duration—the total available bandwidth is very high, in the order of Tb/s.

As pulses from a given frame are scattered over a relatively large period, many concurrent transmissions may occur in TS-OOK without excessively rising the collision probability. This capability to time multiplexing many transmissions is very specific to nanonetworks. In fact, not only could multiple transmissions take place at the same time, but they could even be all correctly received at a given node, provided that bits do not collide. Collision occurs when the receiver is receiving a "0" bit (silence), but in the same time a "1" bit arrives, which effectively shadows the "0". Such collisions are hard to predict especially as the propagation delay, at the considered scale and durations, is usually larger than a pulse duration.

III. Related work

Multi-hop wireless networks have been subject to intensive research in the last two decades and many protocols have been proposed for ad hoc, sensor, and vehicular networks along with many other forms of networks not relying on a fixed infrastructure. Nanonetworks mostly differ from those by:

- The very low computation power and memory available.
- The sheer number of potential neighbors a node can have (thousands and even millions can be envisioned).
- The ability to multiplex many frames over the same period of time.
- The unavailability of global positioning mechanisms.
- The ability to harvest energy from the environment and thus mitigate the cost of communications if they can be spread out over a large period.

The algorithm we propose allows nodes to discover the local density (i.e. how many neighbors they have) while sticking to the constrains of nanonetworks. Specifically, distributed algorithm in nanonetworks should not build a potentially huge list of their neighbors to count them, nor send many messages that could exhaust their energy or saturate the channel. In macronetworks several solutions exists but are not well suited for nanonetworks. For a presentation of related work less suitable to our context, such as mobile phones and sensor networks (low density networks), one can refer to the related work given in [7].

A. Estimation of competing nodes

Bianchi et al. [8] proposed a method to estimate the number of nodes competing for the channel access in a 802.11 network. This estimation is based on the observation of 802.11 busy time slots and time slots where collisions occur. Then using mathematical tools such as collision probabilities (deduced from observations) and Kalman filter, authors are able to estimate the number node of competing for the channel access. The method works at run-time and gives an estimation continuously in real time.

We note that this method estimates the nodes which are transmitting data. In nanonetworks, the number of idle neighbors can be very high, which could heavily falsify the results. We need an estimation method which should work even with no traffic running on the network. As such, this method is not suited for nanonetworks.

B. DIP

Density Inference Protocol (DIP) [9] is a MAC-based protocol that estimates the number of neighbors of each node in macro wireless sensor networks.

DIP uses the channel contention to infer the local density. Nodes estimate the density during some interval of time, which is divided in equal time slots. Each node sends a packet during a random slot of equal probability. To avoid bias given by interferences (several packets sent in the same time), DIP bases its estimation on the number of slots where the channel is free (no communication). DIP uses several iterations, i.e. several intervals of time. Nodes calculate the number of neighbors n statistically using the following formula [9]:

$$n = \log \frac{E(0,n)}{m} / \log \left(1 - \frac{1}{m}\right) \quad (1)$$

where $E(0,n)$ is the number of free slots noticed by the node, and m the total number of slots.

DIP is not appropriate for our work. In nanonetworks based on TS-OOK, the definition of a free channel is fundamentally different than in standard macro wireless network (e.g. 802.11): several messages can be transmitted at the same time (cf. section II), and as such there is no such concept as free channel. Regarding the overhead, the total number of packets used in DIP for the estimation is the number of nodes multiplied by the number of iterations, whereas in our estimation method the number of packets can be smaller than the number of nodes for sufficiently high network densities. Since it uses contention, DIP cannot work with background traffic, i.e. all the traffic needs to be stopped in order for a node to estimate its neighborhood. The formula above needs floating point operations and high precision computing, whereas our solution uses integer numbers only.

C. Estreme

Estreme [7] is a statistic method to estimate the number of neighbors. It works in networks where all nodes perform periodic but random events within a given period, such as sensor networks. The key idea of the method is that the time

difference between two consecutive events captures the density of the neighborhood: the shorter the time difference, the higher the density. It targets networks where nodes have "one hundred neighbors" [7]. All nodes estimate concurrently the density.

As is, Estreme cannot be used in our context, since there is no periodic event in the network. Still, it could be implemented in a similar way to our density estimator, in a separate phase during which each node estimates its neighborhood. In this case, to avoid collision, periods need to be very long, since *all* the nodes send packets. Also, the probabilities used by Estreme are not adaptive, and the range of densities it handles (one hundred) is much smaller than our density estimator. Thus, Estreme is not suited to very high density networks.

D. RFID cardinality estimators

Zero-One Estimator Protocol (ZOE) [10] is a probabilistic protocol that estimates the cardinality of a large RFID system, composed of a reader and numerous tags. In ZOE, the reader uses several rounds, in each round it sends a request to all tags. The tags answer the reader with a fixed probability. The reader estimates the number of RFID tags using the response probability of tags and the proportion of "empty" rounds (without any response).

Bloom Filter based Cardinality Estimator (BFCE) [11] works similarly, but uses an additional initial phase to get a rough estimation of the fixed probability to be used.

RFID methods are not suitable to our problem. An RFID system is a particular 1-hop network. In contrast, DEDeN works in multi-hop networks. Also, the probability used by readers in ZOE is fixed in advance, and the formula to compute it in BFCE is complex, inappropriate for constrained nanonodes. In contrast, DEDeN, by changing the probability dynamically, works with a great range of node densities.

IV. Density estimation algorithm

As explained in section III, because nodes do not have much memory nor processing power, elaborated routing protocols that try to find optimum forwarder(s) cannot work well if at all. On the other hand, flooding and geocasting protocols are simple but inefficient in their pure form if the density is very high. Hopefully, they can be optimized by using the local density of the network to limit their retransmission rate. Some applications are given in section VII.

A. Context and expected properties

We propose a distributed algorithm that estimates the local density, with a particular emphasis on nanonetworks, given that they are the densest networks up to date.

The algorithm can be executed during the network deployment or at hard coded intervals of time, if nodes have sufficiently accurate clocks. Alternatively, upon reception of a 1-hop signal from a macro equipment, if it exists. Finally, it can be executed whenever some node needs to estimate its number of neighbors, based on the application dynamics for example (node movement, node entering or leaving the network). In this latter case, it broadcasts a message (beacon) in the network, which gives for the node receiving it the beginning of the algorithm. This beacon can be sent as pure flooding or by an optimized protocol, such as probabilistic or backoff flooding, with conservative settings in order to reach all the nodes, as presented in section VII.

It is important to note that nodes need to be only locally synchronized. Certainly, in a big multi-hop network, the propagation delay from the initiating node to a distant one can be significant, which means that nodes in the *whole* network do not start the algorithm at the same time. However, our algorithm does not need a global synchronization, but a local one (synchronization among neighhbors), which is guaranteed because the rounds, explained later, are sufficiently long to make beacon reception delay insignificant to neighbors[1].

The algorithm performs properly when started by several nodes concurrently. A node executing the algorithm ignores any new beacon. Also, given that the sender is not taken into account in the algorithm, having two concurrent beacons in different parts of the network just makes the algorithm start sooner in some parts of the network.

The algorithm works in the presence of concurrent background traffic too. In this case, the only requirements are to not experiment too many collisions (very busy channel) and to be able to distinguish our packets from those of other protocols, for example by using a specific identifier in the packet header. When used without background traffic, 1 bit per packet is sufficient, as packets do not carry data. For simplicity, in this article we have not used background traffic.

The algorithm has a statistical behavior: it will not usually give the exact number of neighbors, but a good estimation of it. It works on any density, from a few to millions of neigbours. Even if somewhat unoptimal in sparse networks in terms of number of exchanged messages, its efficiency rapidly increases with the number of neighbors. It can cope with heterogeneous local densities (i.e. some parts of the network much denser than others). It is also designed to benefit from the high multiplexing capability of nanonetworks to obtain its estimation in a very short time, but could be used on any other network. No matter the network density, it never saturates the radio channel and does not require much memory, nor computations. The only memory needed is a counter to store the number of packets received so far. It uses integer numbers through the use of fixed-point arithmetic.

B. Algorithm

When initiated, the algorithm has a fixed maximum duration after which an estimation of the density will be provided. It consists in several rounds R_i of equal duration. For the duration of each round, each node may send a unique `DensityProbe` packet with a probability p_i.

This probability p_i starts with a value close to 0 and increases at each round R_i. At each R_i, the node counts the packets received from its neighbors. Since it knows the

[1] A bit travels through a communication range (of 0.5 mm) in approximately 167 fs, which is negligible to the neighborhood compared to the round duration (10 μs).

transmission probability, it estimates the number of nodes in its transmission range. As the probability progressively increases (multiplied by the *growthRate* parameter at each round), the average number of probes sent during a round increases accordingly. Starting with a very low p_1 ensures that even if many neighbors are present, only a few will effectively send a probe, thus keeping the channel usage and collision risk very low. The lower the initial probability, the higher the potential number of neighbors the algorithm can cope with.

The round duration can be known by each node from the beginning or can be specified in the beacon message. To prevent a large number of collisions to occur at the beginning of each round, nodes select a random waiting time (backoff) between 0 and round duration before transmitting their probes. Collisions between probes are further made unlikely by choosing an adequate round duration. A bit (or pulse) duration being 100 fs, the transmission of a 10 bits probe represents a 1 ps activity on the medium. In our tests, as we expected a maximum of around 1000 probes per round, we used a 10 μs round duration. This makes collisions very unlikely, even among 1000 probes. Note that the maximum number of expected probes depends on parameters set by the user, such as the required confidence interval and the maximum number of neighbors expected.

The algorithm reaches the final estimation when any of the following two conditions is met:

- Transmission probability reaches 1. In that case, at the end of the round, the number of packets received directly gives the number of neighbors. This can occur if the number of neighbors is very low.
- The number of probes that have been received during the current round exceeds a given threshold th_i. As detailed in section IV-C, values of th_i for each R_i can be computed offline and stored in a table in each node; they provide a desired confidence value when estimating the number of neighbors. Additionally, the node will continue to send probes with an increasing probability for a few remaining rounds. In non homogeneous environments, this allows neighboring nodes that did not reach their threshold yet to correctly estimate the density.

A summary of DEDeN is presented in algorithm 1.

Algorithm 1 Density estimation algorithm executed by nodes.

Input: p_1, growthRate > 1, th_i for each round
Output: estimation

```
i = 1
while TRUE do
    send a probe with probability p_i using a backoff
    wait for the round R_i to end
    k = number of received probes in this round
    if k > threshold th_i then
        estimation = k/p_i
        end algorithm
    end if
    if p_i ≥ 1 then
        estimation = k
        end algorithm
    end if
    p_{i+1} = p_i × growthRate
    i++
end while
```

C. Estimation confidence and threshold computation

From a node point of view, probe receptions can be modeled as a binomial distribution with parameters:

- k (the number of successes), the number of probes observed.
- p^{trans} (the probability of success in a Bernoulli trial), the transmission probability.
- n^{real} (the number of trials), the real number of neighbors to estimate.

The natural approach is to compute the number of neighbors $n^{estimated}$ as the average of a binomial distribution:

$$n^{estimated} = \frac{k}{p^{trans}} \tag{2}$$

Its value is therefore the *most likely* number of neighbors. But k comes from a random process. Its value may change between instances of the experiment, causing undesirable variations in the estimation.

The confidence in this estimation depends on the spread of the binomial distribution. However n^{real} is not known, so neither is the effective binomial distribution.

Intuitively, for a given p^{trans}, receiving more probes means a higher number of neighbors, hence a higher number of attempts at the Bernoulli trial. Increasing the number of attempts decreases the spread of the binomial distribution and consequently increases the confidence in the estimation. For a given p^{trans} it is possible to choose a threshold for the number of received probes above which a desired degree of confidence is reached.

(3) computes the probability to observe k probes for a given number of neighbors n and a transmission probability p:

$$Pr(k, n, p) = Pr(X = k) = \binom{n}{k} p^k \left(1 - p\right)^{n-k} \tag{3}$$

In our context, k and p are known, but n is not. The confidence in the estimation of n can be obtained by considering the distribution of the probability to observe a given k against all values of n.

When observing k probes, the minimal possible value for n is k (it represents the possible but unlikely event where n nodes were present and all of them succeeded at the Bernoulli trial). The most likely value for n is k/p. And finally there is no maximum value for n, but the probability to observe only k probes gets very small when the number of potential senders (n) becomes very large.

We can chose an estimation interval which is likely to contain the real value of n with a predetermined high prob-

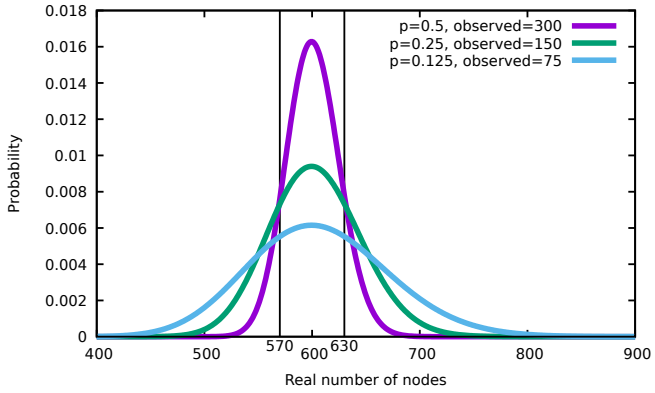

Fig. 1. Probability distribution of n^{real} for various values of k and p.

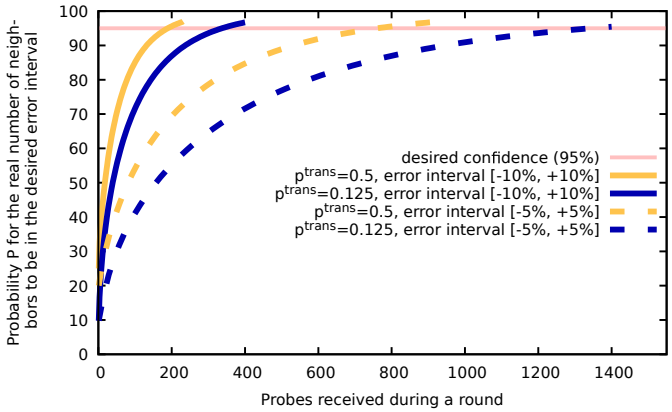

Fig. 2. Confidence depending on probability p and number of observed probes k.

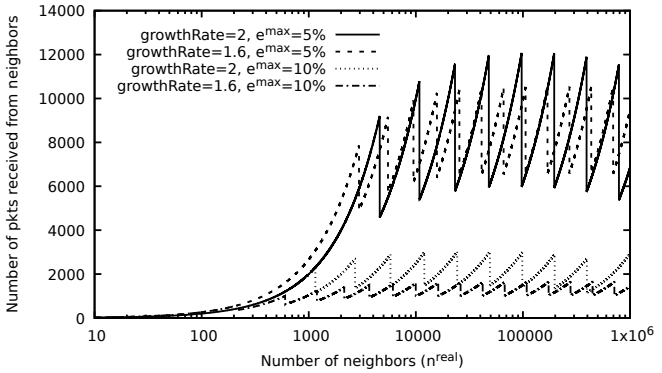

Fig. 3. Theoretical overhead (packets received per node) for various parameters.

ability (usually 95%). For an expected number of neighbors we choose an error threshold e^{max}. (2) becomes:

$$n_{min}^{real} = \frac{k}{p^{trans}}(1 - e^{max}) \tag{4a}$$

$$n_{max}^{real} = \frac{k}{p^{trans}}(1 + e^{max}) \tag{4b}$$

The confidence in an estimation is given by the probability P for the interval $[n_{min}^{real}, n_{max}^{real}]$ to contain n^{real}:

$$P = \frac{\sum_{n=n_{min}^{real}}^{n=n_{max}^{real}} Pr(k, n, p^{trans})}{\sum_{n=0}^{n=\infty} Pr(k, n, p^{trans})} \tag{5}$$

Fig. 1 illustrates (5) for various values of p and k. In this example, the curves show $n^{estimated} = 600$ as the most probable number of neighbors, which can also be computed from (2). The two vertical lines delimit a desired 5% error range $I = [n_{min}^{real} = 570, n_{max}^{real} = 630]$. Here, the confidences P have been numerically calculated as 0.36, 0.52 and 0.78 respectively for (p, k) values of (0.125,75), (0.25,150) and (0.5,300). It means that the real number of neighbors when $p = 0.5$ and observed number of probes $k = 300$ has only 78% chances to be in the estimated interval I.

For the same transmission probability p^{trans}, the confidence P increases with the number of probes received, as shown in Fig. 2. For a given p^{trans}, the threshold th is the number of probes receptions required for P to reach the desired confidence (here, a 95% probability for I to contain n^{real}). For example, receiving 190 probes means $n^{real} \in [171, 209]$ (error interval with $e^{max} = 0.1$) with a probability of 95%.

For a given probability p^{trans} and number of neighbors n^{real}, the number of probes received k might not reach the desired threshold. In this case, the algorithm will proceed to the next round and multiply p^{trans} by *growthRate*. Now, with a higher p^{trans}, more probes will be sent. This process can be repeated until the confidence reaches the desired level.

Choosing a confidence interval has a strong impact on the overhead of the algorithm. Small values for e^{max} require a much higher threshold in the number of probes received (th). This is shown in Fig. 2 with curves using $e^{max} = 10\%$ needing much fewer probes to reach the desired confidence (here 95%).

The theoretical overhead of the algorithm is represented in Fig. 3. The overhead is expressed as the number of packets received by a node. It can be seen that the overhead does not grow anymore for a high number of neighbors. This means that for high densities, only a small proportion of the nodes have to send a probe to get a high quality estimation.

Fig. 3 also shows some spikes. They appear because each round uses a p^{trans} value suited for a range of node densities, and the number of probes received is smaller when the effective number of neighbors is close to the beginning of this interval than close to the end. The *growthRate* parameter affects the variations of the overhead. A smaller value of *growthRate* mitigates these variations, at the price of a higher number of rounds and thus a longer duration of the algorithm.

V. Implementation and Simulation Software

Many technological challenges need to be addressed before wirelessly communicating nanomachines become a reality. Therefore, we have to use simulations to test our algorithm. Three main existing simulators handle nanonetworks.

Nano-Sim [12] is a plug-in for the classical NS3 network simulator. It does not take into account the signal propagation delay, whose impact is significant because of the extremely short duration of the pulse. Indeed pulse are around 100 femtosecond long, and depending on the communication range several pulses can be "in flight" before the first pulse has reached a node. It neither takes into account the actual payload when computing collisions, which is also significant since only a "0" can collide with a "1", cf. section II. *Nano-Sim* proposes a clustered routing system that is not appropriate for our tests. Therefore, Nano-Sim was not really suitable for our work.

Vouivre [13] is a discrete event simulation library that can also work in stand-alone mode. It simulates individual packets and propagation delay. But it uses a statistical model for packet error rate and it does not take actual payload into account. Payload and as correct as possible collision and error calculations are critical for this study and upcoming work, so we preferred to put Vouivre aside.

COMSOL Multiphysics[2] is a simulator for various physics and engineering applications, and as such it allows to simulate THz nanocommunication too. It was used to model the terahertz channel of nanonetworks [14]. It focuses on physical properties to model the reception of each bit by taking into account molecular absorption, channel equivocation, etc. However, the simulation of "a single one-to-one nano-link" has a "very high computational complexity", and "in order to simplify the computational complexity, we simulate a 2D geometry instead of a 3D scenario" [14]. Consequently, this very precise simulator would take too much time to simulate our large networks with complex scenarios.

Inspired by these simulators, we developed a new, small and very focused simulator, *BitSimulator*[3]. It uses a discrete event model and has the following features useful for our study:

- Propagation delay: packet arrival time on a node depends on its distance from the sender.
- Collisions: computation of collisions uses the TS-OOK model (see section II) by checking the actual bit value of each packet currently being received at each node.
- While it allows for multiple concurrent receptions on a given node, it also recognizes that there is a limit to their number (that could be hardware processing power or buffer space for instance). In particular, this parameter has a strong impact on the behavior of upper layer protocols, as packets otherwise correct can be completely ignored.
- Being very focused on nanocommunications, its design is kept simple and efficient. It allows it to scale up to hundred of thousands of simulated nodes.
- As required from this type of simulators, the behavior is completely deterministic, but changed by seeding differently the various dedicated random number generators.

BitSimulator still has room for enhancements. It can be further improved to match more precisely THz nanocommunications by adding, for instance, the channel equivocation.

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

[3]The simulator is at http://eugen.dedu.free.fr/bitsimulator, GPL licence.

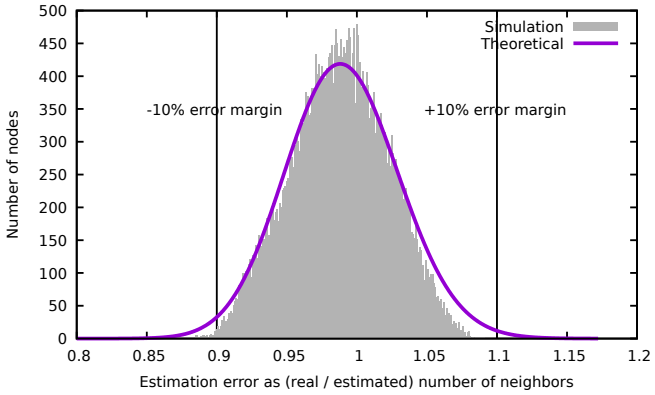

Fig. 4. Estimation error in a 40 000 nodes scenario.

VI. Density estimation results

DEDeN is a distributed algorithm that produces, for each node, an estimation of the number of neighbors. We are mainly interested in two parameters that define the efficiency of the algorithm: the error of the estimation and the number of packets sent during algorithm execution, which stand for the quality and the cost of the algorithm, respectively. It should be noted that the time required to reach an estimation is really short, due to the time scale of the communications in nanonetworks. If communications are not slowed down by a lack of energy, the whole process can terminate in μs.

To validate the theoretical results given in section IV-C, we use BitSimulator with different scenarios. The topology is kept simple with a 2.5 mm by 2.5 mm square area. The communications are omnidirectional with a 0.5 mm radius. Probe packets are 10 bits long. Between 100 to 40 000 nodes are randomly placed in this area using a uniform distribution.

A. Quality as error qualification

DEDeN is designed to produce estimations with a tunable error confidence. We first compare theoretical and simulated error distributions. The scenario has 40 000 nodes. DEDeN is configured so that a node estimation has at least a 95% probability to differ for less than 10% from the real number of neighbors. The estimation error of the algorithm is defined as $e = (n^{estimated} - n^{real})/n^{real}$.

Fig. 4 shows that the simulated error behaves as the theoretical one. In this scenario the simulated results are even better than the requested confidence. This is because the theoretical curve presented is computed at the exact confidence threshold, i.e. for nodes to estimate their neighborhood with an error of *at most* 10% and a confidence of *at least* 95%. In the simulation, the actual number of packets received exceeds this threshold and consequently the quality of the estimation is better.

To experiment further, we run over 200 scenarios with the same square area but with a neighbors count varying from 10 to 4200, *growthRate*=1.6, e^{max}=10%, and confidence=95%. Fig. 5 shows the estimation error distribution for all nodes in all scenarios. The shape is similar to Fig. 4 and proves the

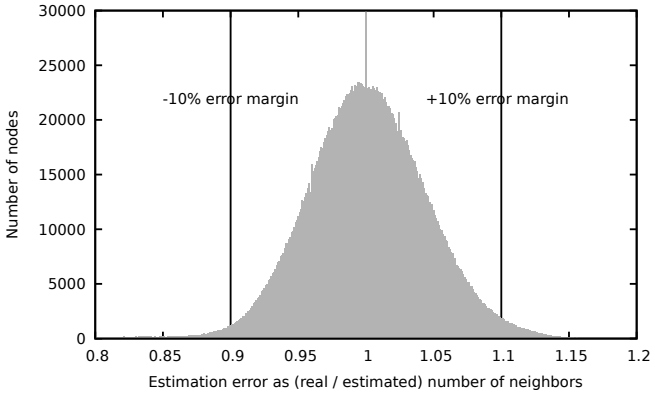

Fig. 5. Distribution of the average estimation error for *all* nodes in 202 different density scenarios.

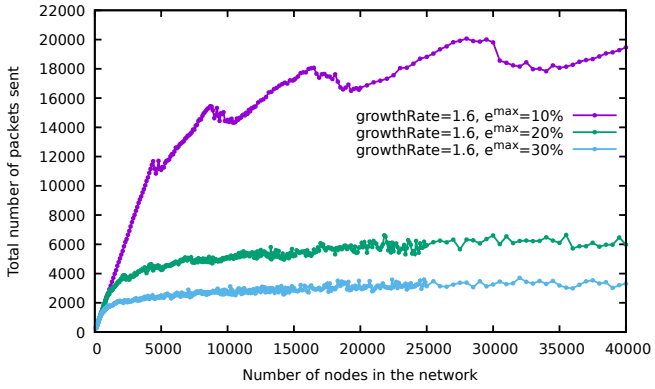

Fig. 6. Packets sent depending on network density and required error range.

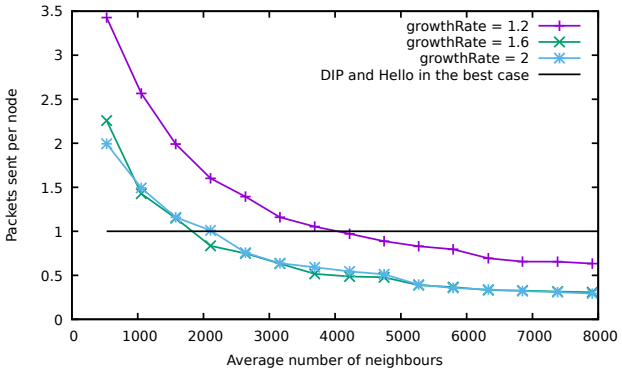

Fig. 7. Ratio $\frac{packetsSent}{numberOfNodes}$.

ability of DEDeN to produce estimations that at least provide the desired requirements no matter the network real density.

B. Overhead as number of packets sent

We define the overhead of DEDeN as the number of packets sent to obtain an estimation of the desired precision.

To observe this overhead we use at set of 200 scenarios with a varying number of nodes between 100 and 40 000. It should be noted that the density in the simulated scenarios is not homogeneous. Nodes near the border of the simulated area have indeed fewer neighbors than those in the center. This affects the results that will slightly differ from theoretical computations that consider an infinite homogeneous environment.

For DEDeN to work well in this non homogeneous environment, as previously mentioned, we let nodes participate in one more round after reaching an estimation with sufficient confidence. This increases the overhead, but is significant only when the neighbors count is relatively small. Fig. 6 shows the total number of packets sent to reach an estimation with *growthRate*=1.6 and various values of e^{max}. The curves present a similar shape to the theoretical one given in Fig. 3, but smoothened due to the non homogeneity of the topology. Fig. 6 also highlights the decrease in overhead when relaxing the constraint on e^{max}. Both Fig. 3 and Fig. 6, which present the number of packets required to reach a targeted confidence in the estimation, show that this number almost stops growing when the number of neighbors becomes very high.

Finally, Fig. 7 compares our algorithm to DIP for various values of *growthRate*. These curves are computed for $e^{max} = 10\%$, which as seen in Fig. 6 is a pretty conservative and costly value. It shows that for DeDen the packets sent per node ratio decreases with the density. In both DIP and Hello, each node sends a packet in each iteration, and we took the best case for them, i.e. only one iteration, so the packets sent per node ratio is 1. Depending on the value of *growthRate*, DeDen is better than DIP above 1800 (*growthRate*=2) to 4000 neighbors (*growthRate*=1.2). Therefore, for dense nanonetworks, our algorithm needs much fewer messages thanks to its ever-growing efficiency.

VII. Applications

Applications running over multi-hop wireless networks may have many different communication patterns, such as data collection, broadcasting, unicast flows or even mostly local exchanges. The flooding is the most basic packet forwarding scheme, where nodes retransmit packets they receive for the first time. Pure flooding in a very dense network (thousands of neighbors) is extremely inefficient as most retransmissions propagate the packet to few (if any) *new* nodes. Moreover, countermeasures have to be taken to prevent skyrocketing contention for channel access or collisions.

Consequently, more advanced routing protocols are used. Multi-hop routing protocols usually use proactive or reactive schemes, and sometimes a mix of both. Reactive routing schemes rely on flooding to find new routes, hence are vulnerable in very high density networks. Proactive schemes try to build and maintain routes, and usually need to have detailed knowledge of at least their direct neighborhood, consequently requiring a large amount of memory in high density networks.

DEDeN estimates the number of neighbors of each node. This information is especially useful as a base to optimize these routing protocols and other applications running on the network. In this section we first present how this estimation can be used to mitigate the overhead and the unreliability of information diffusion with two types of probabilistic flooding.

Then we consider Stateless Linear Routing (SLR) for 3D networks [15], a nanonetwork routing protocol, which can also be significantly enhanced by our estimation.

A. Simple adaptative probabilistic flooding

Multi-hop data broadcasting is a service of extreme importance in wireless ad hoc networks. It is required by a large number of applications and high level protocols, and also used to spread emergency information in disaster networks. The simplest method to implement it is flooding: each node forwards the packet it receives. But the flooding has two drawbacks: first it generates a considerable amount of messages, one packet per node; second, to prevent further retransmissions, each node has to memorize the id—supposed unique—of each packet.

Traditional optimizations involving the selection of subsets of forwarding nodes often suffer in very dense nanonetworks. This is caused by the inability to build a complete map of even the direct neighbors or because of too high memory requirements. A common solution to this issue is to probabilistically limit the number of forwarders, by giving each node a probability to forward a new packet (packets already seen are discarded). A survey on probabilistic broadcast schemes is presented in [16]. The schemes are divided in fixed and adaptive. In fixed schemes, a fixed probability of forwarding is used by all nodes. The broadcasting can further be optimized by taking into account network characteristics: in non-counter-based adaptive schemes (counter-based schemes are presented in the next section), the probability depends on several factors, such as density, distance, speed and others. Among these, "the number of neighbors is the most used density metric" [16]. Commonly, the forwarding probability is inversely proportional to the number of neighbours of a node, $p = k/n_b$, where k is the propagation factor and "n_b can be easily obtained via hello packets". [16] does not consider very dense networks, where this sentence *does not hold anymore*: hello method leads to numerous packets and needs much memory in nodes for id memorization. This is exactly what DEDeN provides: an estimation of the number of neighbors efficiently computed in high density networks.

We implemented in BitSimulator a simple adaptative probabilistic flooding and feed it with DEDeN estimations. The k factor is used along with the density estimation to control the *average* number of retransmissions in an area. Because of space limitation we do not provide here a complete analysis of the results, but prefer to point to the main risk of this simple method. Fig. 8 is produced from a 10 000 nodes scenario in which data packets are multi-hop-broadcasted. It shows the distribution of number of copies received by each node. In this case, the requested *average* number of copies was $k = 5$. But it can be seen that a large number of nodes have *not* received the packet: the propagation stopped too early, leaving parts of the network out of the broadcast. In turns out that k value is too small. Increasing it further increases the number of copies received. It would also be very costly to set it big enough

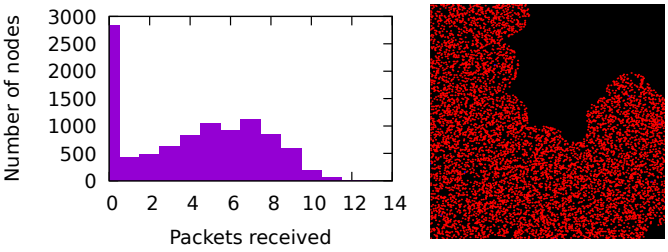

Fig. 8. Simple probabilistic flooding: distribution of the number of copies received (left), incomplete propagation area, in black (right).

to eliminate with a high confidence the risk of propagation interruption.

To conclude, in dense networks, especially with constrained resources, adaptive probabilistic flooding is better than fixed one. The local probability is derived from the density estimation provided by DEDeN and can be tuned to adapt to hostile environments (at the cost of a somewhat higher number of retransmissions). This simple probabilistic flooding requires very few memory (only for packet ids), giving it an edge over more complex adaptive floodings. Nodes take the decision to forward a packet immediately upon reception, and can delete it afterward.

B. Backoff-based adaptive probabilistic flooding

Another usual—and also more reliable—method to optimize multi-hop flooding is to count the number of retransmissions, also known as adaptive counter-based schemes in [16]. A node that receives a packet decides to effectively forward it if it has seen less than a given threshold copies of the packet. This threshold, called *redundancy* in the following, guarantees (collisions put aside) that a message is forwarded at least a given number of times.

However, wireless nanonetworks work differently. 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. This backoff period is picked in a window from 0 to t. t should be dimensioned in order to let nodes notice that a neighbor has already forwarded the packet before starting to forward its own copy. A small value of t when the local nodes density is high means a high probability of concurrent retransmissions, and even collisions. Hence, this duration t depends on the number of neighbors. Thus, since DEDeN provides an estimation of the number of neighbors, it can be used to dimension an ideal duration t.

Backoff flooding does not give any guarantee on the exact number of forwarders. It involves a random number, and due to the specificities of nanocommunication (collision and channel multiplexing) and the highly distributed nature of the algorithm, several nodes can use very close values of backoff and send a copy of the same packet simultaneously.

Like the probabilistic flooding presented earlier, backoff flooding drastically reduces the number of retransmissions. In comparison, it has two major advantages. First, it is much

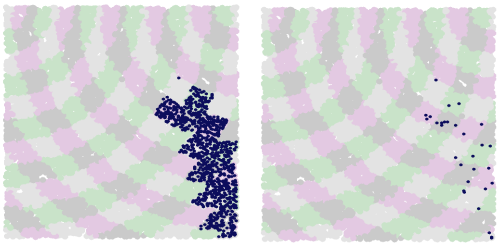

Fig. 9. Forwarding nodes in SLR scenario: without (left) and with probabilistic forwarding (right).

less prone to incomplete coverage because it tries to ensure the requested redundancy. Secondly, it can be tuned to have a low redundancy variation. On the other hand it implies an additional delay that increases with reduction of the number of forwarders and the spread of its distribution. As the decision to forward or delete a packets has to be postponed, this solution also requires to keep them in memory for some time. As we have seen, the maximum waiting time t depends on the number of neighbors. Consequently the backoff flooding in nanonetworks greatly benefits from DEDeN.

C. Stateless Linear Routing

Stateless Linear Routing (SLR) [15] is a routing scheme appropriate for nanonetworks. It works by forming groups of nodes depending on their distance (in hops) to a set of anchor nodes. The set of distances to the anchors form a topological address that can be shared by multiple nodes. These distances are obtained during the addressing phase, prior to any routing, through the flooding of beacons from the anchors. Subsequently, packets are routed in the network following a scheme similar to geocasting. Nodes choose to retransmit a packet if they are on the path between the source and the destination. As many nodes may share a topological address, the number of retransmissions can be very high. SLR heavily relies on broadcast, both for its addressing and routing phases and suffers in very dense networks. This makes it a good candidate for optimization through probabilistic or backoff flooding, as shown before.

We implemented SLR and modified its forwarding code to use density estimation from DEDeN. Fig. 9 illustrates how the number of forwarders drops when using probabilistic forwarding. In both cases the message reached its destination, but with 683 retransmissions (left case) and only 35 (right).

VIII. Conclusion

This article presented a distributed algorithm to estimate the node density in very dense networks, and in particular in electromagnetic nanonetworks, an appropriate example of such networks. The algorithm is executed by each node and is based on consecutive rounds. During each round, until a computed criterion is met, the probability of sending packets increases and the quality of the estimation too. This quality is completely tunable to the requirements of upper layer protocols, and the overhead in terms of exchanged messages is predictable. The memory requirements are tiny: only a counter is needed.

The usefulness of the algorithm was demonstrated on two probabilistic flooding algorithms and the SLR routing protocol. The proportion of nodes sending probes is much smaller compared to classical floodings, and strongly decreases with the density. Its application field is large and many other routing algorithms, protocols and applications can benefit from DEDeN. To the best of our knowledge, this is the first proposed estimation method suited for nanonetworks.

Future works include taking advantage of background traffic to reduce overhead, and develop new strategies based on our estimator to reduce congestion in saturated networks.

References

[1] I. F. Akyildiz and J. M. Jornet, "The internet of nano-things," *IEEE Wireless Communications*, vol. 17, no. 6, pp. 1–6, 2010.

[2] 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.

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

[4] S. Goldstein and T. Mowry, "Claytronics: A scalable basis for future robots," in *16th RoboSphere*. Mountain View, CA, USA: NASA, Nov. 2004, pp. 1–7.

[5] I. Llatser, C. Kremers, A. Cabellos-Aparicio, J. M. Jornet, E. Alarcón, and D. N. Chigrin, "Graphene-based nano-patch antenna for terahertz radiation," *Photonics and Nanostructures-Fundamentals and Applications*, vol. 10, no. 4, pp. 353–358, 2012.

[6] J. M. Jornet and I. F. Akyildiz, "Information capacity of pulse-based wireless nanosensor networks," in *8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*. Salt Lake City, UT, USA: IEEE, Jun. 2011, pp. 80–88.

[7] 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.

[8] G. Bianchi and I. Tinnierlo, "Kalman filter estimation of the number of competing terminals in an IEEE 802.11 network," in *22th IEEE International Conference on Computer Communications (INFOCOM)*. San Francisco, CA, USA: IEEE, Jul. 2003, pp. 1–9.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] J. M. Jornet Montana, "Fundamentals of electromagnetic nanonetworks in the terahertz band," Ph.D. dissertation, Georgia Institute of Technology, Atlanta, GA, USA, Dec. 2013.

[15] A. Tsioliaridou, C. Liaskos, E. Dedu, and S. Ioannidis, "Packet routing in 3D nanonetworks: A lightweight, linear-path scheme," *Nano Communication Networks*, vol. 12, pp. 63–71, Jun. 2017.

[16] 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.