Ray tracing routing using packet reception timing in dense nanonetworks

Eugen Dedu, Masoud Asghari

Ray Tracing Routing Using Packet Reception Timing in Dense Nanonetworks

Eugen Dedu[a], Masoud Asghari[b,*]

[a] *Université de Franche-Comté, CNRS, institut FEMTO-ST Numérica, cours Leprince-Ringuet, 25200 Montbéliard, France.*
[b] *Department of Computer Engineering, Faculty of Engineering, University of Maragheh, P.O. Box 55136-553, Maragheh, Iran.*

Abstract

This paper presents a novel routing method called ray tracing. The method targets wireless nanonetworks, which are networks of nanometer-sized nodes with applications in various domains such as future medicine and metamaterial. Due to their tiny size, nanonodes have quite limited resources, in particular energy, and it is crucial to use as few resources as possible. In multi-hop nanonetworks, resource consumption for routing depends on several factors, such as the number of nodes that forward packets, and the number of packets sent and received. To reduce the energy used, in the proposed ray tracing routing, only the nodes that receive duplicate packets at the same time forward them. This leads to the selection of the forwarding nodes over a quasistraight line between the source and the destination, which can bend at the edges of the network. The proposed method is a major improvement over its previous version, aiming to fix several crucial issues such as double propagation, large forwarding angle, and die-out. Moreover, the applicability of ray tracing in heterogeneous and 3D nanonetworks is also analyzed and a practical application for the protocol is proposed. Extensive simulation results demonstrate the fixing of these issues, and the superiority of the proposed improved ray tracing algorithm over other routing methods in the long and narrow nanonetworks, such as blood vessels and branches.

Keywords: Forwarding, Electromagnetic nanonetworks, Routing, TS-OOK.

Declarations of interest: none.

1. Introduction

Recent developments in nanotechnology explore the possibilities of creating integrated nanodevices that are very small and have new applications that are not possible on larger scales [1, 2]. Nanodevices are made from nanometer-sized pieces with very limited resources and energy, so they cannot do much by themselves. Therefore, they need to work together in a distributed network of nanonodes to achieve complex tasks over a large area. The paper focuses on electromagnetic nanonetworks, which are networks of nanonodes that are

*Corresponding author.
Email addresses: eugen.dedu@univ-fcomte.fr (Eugen Dedu), mas.asghari@maragheh.ac.ir (Masoud Asghari)

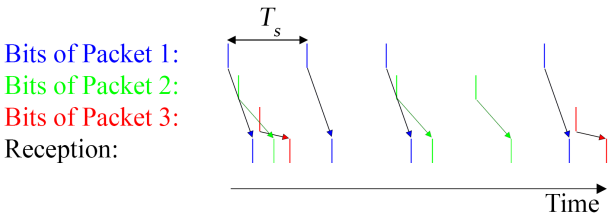


Figure 1: Bits of each packet in TS-OOK are sent with T_s interval and can be received with the same T_s interval after some propagation delay.

smaller than 10 µm and communicate in the terahertz band. These nanonodes usually have a nanoprocessor, a nanomemory, a nanosensor, and a nanotransceiver (including a nanoantenna) for wireless communication [1, 2]. The nanonodes are self-powered by harvesting energy from their environment via a piezoelectric nanogenerator, for example [1, 2]. Nanonetworks can contain numerous nanonodes in a small space, which makes it hard for traditional routing methods to work. Also, because of their tiny size and small transmission range, normal network techniques are not suitable.

The fundamental modulation proposed for wireless electromagnetic nanonetworks is Time Spread On-Off Keying (TS-OOK) [3], which only uses simple extremely short (e.g., $T_p = 100$ fs [3]) pulses without any signal carriers. A bit 1 is transmitted by a pulse of power burst, and a bit 0 is "transmitted" as silence. Due to hardware constraints, a nanonode cannot send all the bits of a packet consecutively: it needs to wait for $T_s = \beta \times T_p$ between transmission of each bit of the packet [3]. An example of the time spreading ratio is $\beta = 1000$ [3]. Since the interval between consecutive bits of a packet is T_s, a receiver can detect the bits of the packet by reading bits at T_s interval, e.g. at times x, $x + T_s$, $x + 2T_s$, ..., whereas the bits received at other times belong to other packets. Figure 1 shows an example of transmitted 1 bits as pulses and the detected bits at different times at a receiver. For simplicity, it is assumed that any T_s interval is divided into β time slots that are repeated every T_s interval. In this analogy, all the bits of a packet are received at the same time slot number.

Nanonetworks can interact with the larger world through gateway nodes, creating the Internet of Nano-Things (IoNT). When the nanodevices are introduced into the human body, for example by injecting them into the bloodstream, they can sense different parameters and pathogens in the body. The nanodevices can communicate with each other inside the human body, creating intrabody nanonetworks [4, 5]. This can enable applications in future medicine, such as detecting and reporting diseases at the molecular level, delivering drugs with high accuracy, monitoring health conditions, and performing neurosurgery. Nanodevices can also create Software-Defined Metamaterials (SDMs), which can change their shape and electromagnetic properties through commands received from nanodevices. Nanonetworks have applications in other fields such as military, intelligent agriculture, and programmable matter such as wireless modular robots.

Nanodevices have a very short communication range, usually less than 1 cm, because they use the THz band, which is highly affected by water, such as in blood. The water makes the signal drastically weaker,

and nanodevices do not have enough power (because they are very small) to send the signal further. Therefore, they need to use multiple hops to send data packets across the nanonetwork, and a routing protocol. However, the traditional routing methods that are used in other networks are not suitable for nanonetworks (for example, cluster-based approaches would quickly deplete clusters' batteries). Nanonetworks should use routing methods that are designed for their specific conditions (such as the size and shape of the network, the location and movement of the nanodevices, the energy source and limitations of the nanodevices, and the applications they are designed for). This can help reduce the number of packets that are sent in the wrong direction and the number of nanodevices that forward the packets. Therefore, it can make the nanonetwork work more efficiently and last longer.

We previously proposed a routing method appropriate to nanonetworks, which introduced the ray tracing principle and a basic algorithm [6]. Ray tracing uses the features of TS-OOK to send packets in a quasistraight line from the source to the gateway without sending the packets to all the nanodevices in the network. This can save energy and resources by reducing the number of nanodevices that receive and forward the packets. In ray tracing forwarding, only nanodevices that get the same packets at the same time will forward them further. The forwarding nanodevices form a nearly straight line, but they can bend at the edges of a long nanonetwork. Unfortunately, for various reasons, presented below, this algorithm does not give good results in some cases (double or multiple propagation appears, the propagation line has large deviation angles, or the propagation stops).

Therefore, in this paper, we propose a new ray tracing routing for long and narrow nanonetworks (e.g., pipes, blood vessels, and gastric tract where the other routing methods used for nanonetworks do not work well) that uses a small number of forwarders to save the limited resources of the nanonodes. This protocol improves the basic version through the addition of four control packets (RTF, CTF, FORGET, and FORGET2). Through a thorough analysis of the algorithm, we show that it avoids most of the issues of the basic algorithm (as analyzed in Section 4): RTF and CTF packets fix the double propagation issue, and FORGET and FORGET2 packets allow retrieval of the transmission to fix the die-out cases. By preventing the selection of too close forwarders, the issue of large forwarding angle is also fixed. We model the forwarding angle mathematically and analyze the impact of nanonode distance and transmission range. We implement the improved protocol in a nanonetwork simulator, validate the simulator's results, and use it to evaluate the performance of the improved ray tracing and to compare it with the related routing methods. Moreover, the applicability of ray tracing in heterogeneous and 3D nanonetworks is also analyzed and verified by simulations. As a proof of concept, a practical application for the proposed protocol in the human cardiovascular system is envisioned.

1.1. Motivations and novelties

The paper for our previous version concluded that it "presented preliminary results of the ray tracing forwarding", and opened a few perspectives, such as to "analyze the algorithm in less dense scenarios, where

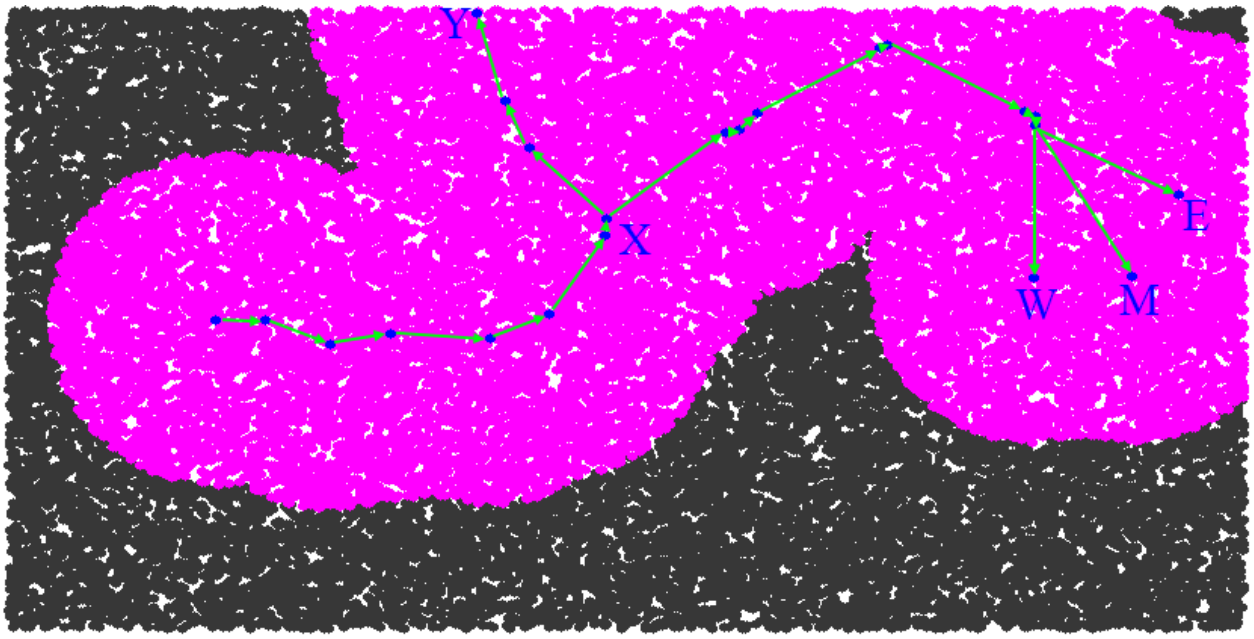


Figure 2: If the forwarding angle is too wide so that some potential forwarders do not hear each other directly, the propagation can duplicate (node X) or can even stop (nodes W, E, and M). Also, if propagation is perpendicular to the network border, the propagation can stop (node Y).

sometimes there is no node in the forwarding direction, [and] in even denser networks, where forwarding can take several directions" [6].

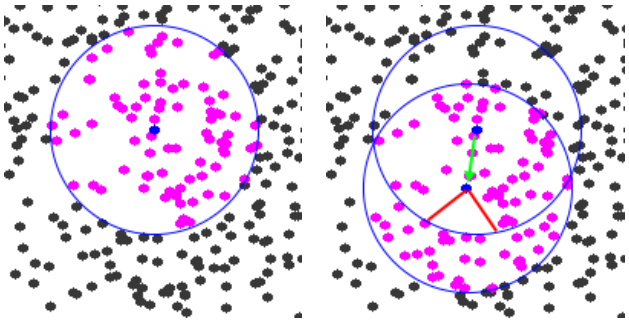
In alignment with these conclusions, we made numerous tests with the basic algorithm [6] by varying the RNG seed used for the backoff before forwarding. We mainly used 30 000 nodes in the network (which gives an average density of 2803 neighbors), but also tested a less dense network with 2000 nodes. The transmission range is 0.8 mm, but also tested with 0.5 mm. We noticed several issues, described in the following.

1.1.1. Double propagation caused by distant forwarding nodes

Figure 2 shows the double propagation problem. Compared to one propagation only, the double propagation uses more senders, whose number we want to keep small, and sometimes can even stop the propagation.

To explain the double propagation, we notice that the size of the surface where potential forwarders are located can be so large that some potential forwarders do not hear each other. In this case, two distant nodes can forward the packet, leading to double propagation (we also noticed triple propagation during simulations). This can be seen in the middle of the figure after node X, with two next forwarders, to the top-left and top-right, which are too far to hear each other.

When the next forwarder is in the middle of the two forwarding nodes, the propagation can even stop. This can be seen in the same figure at right. We note by W, M, and E the last three nodes. W and E are too distant to hear each other, and both of them are forwarders. If a node close to the perpendicular bisector of the WE segment happens to be the next forwarder (i.e. M), no potential forwarder exists, because all of



(a) (b)

Figure 3: In less dense networks, the propagation can stop because no collinear potential forwarder is found.

them have received three copies of the packet (they are in the transmission range of the three nodes W, M, and E). Then the propagation stops, as shown in the figure.

The proposal of the current article avoids these problems related to multiple propagation through an RTF/CTF packet exchange: any potential forwarder sends an RTF first, and the current node answers with CTF approving the next forwarder. Even if several nodes send RTF, only one is approved by the current node, hence the multiple propagation problem is avoided. This is detailed in Section 3.2.3.

1.1.2. Large forwarding angle

We noticed that in the basic algorithm, sometimes the propagation line has large angle changes which leads to non-linear behavior and even the risk of inversion of the propagation direction. The large forwarding angles can be seen in Fig. 2, where the ray changes its initial direction from east to north (towards node Y) and south (towards node W). In this paper, the forwarding angle is reduced (i.e., controlled) by preventing the selection of too close forwarders as explained in Section 3.2.4.

1.1.3. Propagation stop (die-out)

During the propagation, sometimes no potential forwarder exists, and the propagation stops, leading to what is called a *die-out*. We present here two cases.

One case is when the angle of attack to the border is close to 90°. In this case, all the potential forwarders are out of the environment, hence the propagation stops, as shown in the top-left of Fig. 2 for node Y.

Another case appears often in less dense networks. Figure 3a shows the first forwarder (node in blue) along with its neighborhood (nodes in magenta), and Fig. 3b shows the second forwarder (node in blue on the south of the first forwarder) and its neighborhood. There is no node in the south of the neighborhood of the second forwarder (depicted by the red angle in Fig. 3b) that has received the packet from the first forwarder. Thus, there is no potential forwarder, and the propagation stops.

To avoid these problems, the current article proposes to make the propagation retreat in case of *die-out* through the rollback procedure, described in Section 3.2.5.

This section showed some issues with the basic algorithm [6] and mentioned how the improved algorithm in the current article solves them.

The remainder of this paper is organized as follows. Related work is given in Section 2. The improved method functioning including the basic ray tracing algorithm and its improvements are presented in Section 3. Validation of the implementation, and evaluation via simulations take place in Section 4. An envisioned application of the ray tracing routing in intrabody wireless nanonetworks is examined in Section 5. Finally, the conclusion is given in Section 6.

2. Related work

There are two ways to route data packets in nanonetworks: one is to send them to a specific node through the nodes in between (this is called unicast or zone-cast routing), and the other is to send them to all the nodes in the network (this is called flooding or broadcast). Nanonetworks have some challenges that make it hard to use traditional methods of addressing and routing, such as the shape of the network, the movement of the devices, and the very high number of homogeneous devices. Therefore, flooding is the only option that can work in most cases for nanonetworks. In *pure flooding*, all unique packets that are received by a node are forwarded by the node, leading to poor performance in dense networks due to resource wastage and broadcast storms. In *probabilistic flooding*, nanonodes forward packets with a certain probability that should be tuned to prevent broadcast storms and guarantee message delivery [7]. *Backoff flooding* is an efficient flooding strategy that utilizes a counter-based mechanism to count the number of received packets and adds a random waiting time (backoff automatically tuned based on the neighbor density) to packet forwarding [8]. Only those nodes that did not receive r (redundancy factor) copies of a packet during the backoff period forward the packet. Since in all flooding-based methods (including its variants such as probabilistic and backoff), the data packets are broadcasted to all the nodes in the network, not just the one that needs it, the efficiency of the network resources is low. Therefore, methods that can limit the number of packets and forwarders in the network can increase network efficiency.

Routing methods that use geographic information, such as geoforwarding, need to know the positions of the devices that can be obtained either directly through a GPS receiver in each node or by triangulation techniques using relative positions of the nodes obtained by anchors and beacons [9, 10]. Due to the high density, small size, and limited resources of the nodes in the nanonetworks, protocols that rely on location, or on a routing table and neighborhood knowledge cannot be used in nanonetworks [8]. In nanonetworks with short transmission ranges and its simulators (including TeraSim [11]), the THz band is considered as a single transmission window (almost 10 THz wide) with impulse radio type communications based on the transmission of femtosecond-long pulses. Since the path loss is very small at such short distances, nodes can rely on omnidirectional antennas (based on nanomaterial such as carbon nanotubes) to communicate [11, 12]. Some special routing methods such as RADAR, DEROUS, and SLR use geoforwarding only in special conditions and environments. In *RADAR* routing, the nanonetwork consists of a circular area with a central

node transmitting a directional signal within an angle [13]. The nanonodes that receive the signal are inside the radiation angle and they change their state to ON and the remaining nodes stay in the OFF state. A convex region and a symmetric perimeter with a special node in the center are required for RADAR to work. Similarly, *DEROUS* classifies the nanonodes by utilizing the transmitted packets of a beacon node positioned at the center of a 2D circular area [14]. In *SLR*, the network is in a static 3D cubic space with a few anchor nodes placed at the corners. SLR requires an initial setup phase with anchors transmitting packets in sequence allowing the nanonodes to find their coordinates as hop counts from the anchors [15]. After the setup phase, each node can find out if it is located on the linear segment connecting the sender to the recipient node and relay the packets accordingly [15].

Straight-line routing protocol is a form of random-walk routing protocol in wireless sensor networks (WSNs) [16, 17]. In Straight-line routing, it is assumed that each node can determine precisely its distance to the transmitter based on the signal strength. By using this distance information, a hop-by-hop linear path between the source and destination is constructed by those nodes that lie on the extended line of the path [16, 17]. The transmission range of the nodes in nanonetworks is much shorter than in WSNs. Unlike WSNs, nanonodes usually use TS-OOK modulation in the THz band (as described in the Introduction) by transmitting very short pulses and their very simple receiver can only detect THz band pulses if they are placed in the transmission range of the transmitter. Unlike WSNs, received pulse strength cannot be used to measure accurate distances between the nanonodes because of the short ranges, TS-OOK modulation, and oversimplification of the receivers, rendering the protocol incompatible for nanonetworks. Despite that, TS-OOK utilizes very short pulses with accurate pulse duration T_p and waiting interval T_s to detect consecutive bits of each packet allowing temporal multiplexing of multiple simultaneous packets. In the proposed ray tracing route, we employ these TS-OOK characteristics (pulse propagation and arrival times) to forward packets in quasistraight lines and reduce the number of forwards.

In order to solve the die-out problem, we use a rollback procedure which is a form of backtracking algorithm. Backtracking is a known class of algorithms for solving computational problems (such as constraint satisfaction problems), that incrementally builds candidate solutions, and drops a candidate (i.e., backtracks) as soon as it detects that the candidate does not lead to a valid solution. In many wireless sensor network routing problems (such as [18]), the packet backtracks to the previously visited nodes when a local minimum is met.

In order to reduce the large forwarding angle, we prevent the selection of too close forwarders. This limitation resembles the ring in [19, 20, 21], where each transmitting node first selects potential forwarders that fall in a ring near the border of the node's communication range to reduce the number of forwarders.

3. Method functioning

3.1. Assumptions

The ray tracing mechanism uses the following features from TS-OOK modulation itself:

- The nodes should be able to listen at T_s interval and send bits at a given time without any random backoff at the physical level.
- Nodes must be able to know whether the difference in time between two incoming pulses (= bits) is less than $1\,T_p$ or not, modulo some number. This assumption is similar to the current assumption of TS-OOK, where nodes can match bits of the same packet to be able to use temporal multiplexing of TS-OOK [22]. Therefore, nodes need to be able to differentiate the reception time of packets with a high accuracy (T_p).

Additionally, the ray tracing mechanism works under the following assumptions:

- The propagation model is a unit disc.
- Nodes can send low-power pulses (with a transmission radius of around 1/3 of the regular transmission radius).

Note that the nodes do not need a time sync and each node has its own time reference (clock).

3.2. Ray tracing routing algorithm

The ray tracing aims to choose forwarders in a quasi-straight line. To achieve that, it uses the notion of collinearity. This section explains this notion, followed by the algorithm itself.

3.2.1. Node collinearity in nanonetworks

In mathematics and other fields, three nodes are collinear if one of them is located on the same line connecting the two other nodes, based on a *spatial* collinearity. In nanonetworks, this definition can be translated to *time* collinearity: one node sends a packet, the other node sends another packet at the exact time that it receives the first node packet, and a third node is collinear if it receives both packets at the same "time". The "time" has a specific meaning in nanonetworks, explained in the following.

First, it should be noted that when two bits arrive at the same time (more precisely, at a time difference less than $1\,T_p$) at a receiver, they generate a collision, which is to be avoided. Therefore, and given that all the bits are sent spaced by T_s in TS-OOK, we consider throughout the whole article that two packets are received *at the same time (T_p)* if the time difference between their reception modulo T_s is less than $1\,T_p$.

Thus, node collinearity in nanonetworks involves the *timing of packet receptions*: when two 1 bit pulses (note: 0 bits do not emit pulses, so they cannot be used) from two copies of the same packet sent by the first two nodes at the same time slot arrive at the *same* T_p, the receiver considers itself to be collinear with the two senders. As previously written, which 1 bits are taken from the packets does not matter, since consecutive bits of one packet are spaced by $1\,T_s$, which is zero modulo T_s.

This is illustrated in Fig. 4: node C, which is collinear to A and B, receives both packets at the same time slot $x + 7T_p$ (based on any node's time frame), whereas node D, which is not collinear, receives them in different time slots $x + 5T_p$ and $x + 7T_p$ respectively.

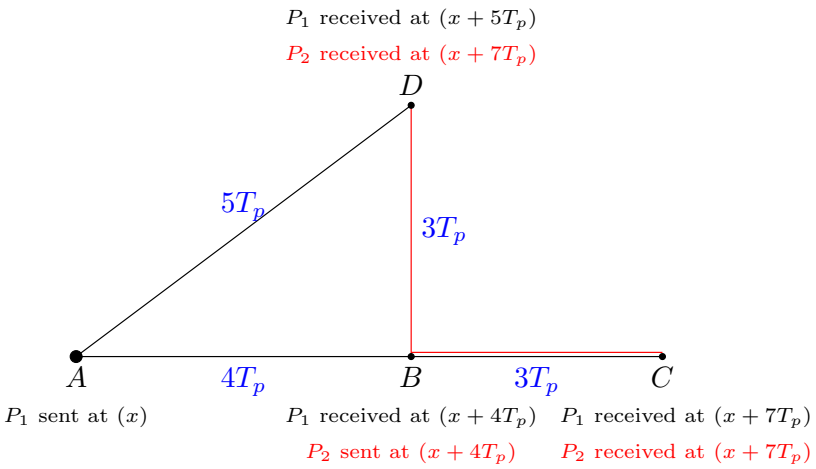


Figure 4: Node collinearity in nanonetworks.

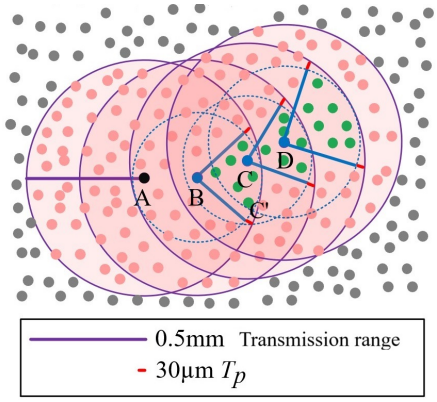


Figure 5: An illustration of ray tracing forwarding with potential forwarders that are "collinear" with the previous two transmitters in green, and effective forwarders (B, C and D) in blue [6].

It is important to note that, T_p being not null, not only C, but also nodes very close to C receive at the same time slot as C, i.e. at $x + 7T_p$, so they are collinear too.

3.2.2. Basic ray tracing forwarding algorithm [6]

This section presents the original algorithm (slightly reformatted), as described in our conference paper [6]. We will illustrate the algorithm using Fig. 5.

We describe the algorithm using only one packet, of a given id (identifier). For the general case of multiple packets and flows, packets are processed independently, and we suppose each packet generated by a source has a unique identifier in the network, to be differentiated (uniquely identified) from other packets, implemented as a combination of source address, source port or flow number, and packet sequence number fields for example.

Only nodes having received *the first two copies* at the same T_p (modulo T_s) in their local time domain are potential forwarders.

All the nodes execute the same procedure, in a loop. Thus, we describe the procedure recursively: the base case (initialization), and the recursive step (general case).

Initialisation step: A source A, called the *sender*, sends a packet. All the nodes inside the transmission range receive it. One of them, B, which is called the *steering node* because it steers the forwarding direction, and forwards it.

Recursive step: We are at the time where node A has already sent a packet, and node B is currently sending a packet. This message will be received in all its transmission range. All the nodes in the network are in one of the following cases:

- It is the first packet it receives: node just memorizes the time slot.
- It is the second packet it receives: if collinearity (like node C) then schedule packet after some backoff.
  Details: If the current packet is *not* received in the same T_p as the first copy, then the node discards the packet and all the ensuing packets with the same id; otherwise (i.e. it is received in the same T_p), it uses a backoff: if it receives any copy of the same packet before the backoff (implying that another collinear node has volunteered), discard it, otherwise send it at the same T_p.

  The window from which the backoff is chosen is large, hence only one of the potential forwarders (nodes receiving the packets in the same T_p) will forward, whereas all the other potential forwarders will see the copy of this packet and as such will discard their packet. In this window, not all the values (backoffs) are eligible. Indeed, only backoffs so that the sending time is the same as the receiving time are allowed. This means that the eligible backoffs are the same modulo T_s. Also, note that a very large window does not add a very large forwarding delay. The delay comes from the *forwarding* node, which is the node having picked the smallest backoff, which is very small compared to the window size.

  Geometrically, all the nodes located in the intersection of the discs (transmission range) of the first and the second packets receive both packets. Some of them receive the two packets in the same T_p, depicted as green nodes in Fig. 5, and consider themselves as collinear with A and B; they form the *forwarding area*. The other nodes receive them in different T_p, and they discard the packet and forget about the communication (i.e. packets with this id).
- It is the third or ensuing packet it receives (like node C'): discard it (because its forwarding time has already passed, and the propagation is already in front of it).

Once node B sends the packet, the procedure is repeated (recursive step) with node B becoming sender A, and node C becoming steering node B.

The algorithm executed by each node upon reception of a packet data[id] is shown in Fig. 6.

3.2.3. First improvement: RTF/CTF

As explained in Section 1.1, the original proposal can lead to double propagation. The goal of the first improvement is to avoid it. For this improvement, two new control packet types are added: RTF (Request

```
Input:
  timefirst[] = null

Upon reception of data packet data[id]:
if timefirst[id] == null then
    // packet id seen for the first time
    timefirst[id] = (crttime % T_s) / T_p    // % is modulo operation
    timesecond[id] = null
elseif timesecond[id] == null then
    // 2nd packet id received
    timesecond[id] = (crttime % T_s) / T_p
    if (crttime – timefirst[id])%T_s < T_p or (crttime – timefirst[id])%T_s > T_s − T_p then
        // same T_p, so collinearity
        backoff = pktsize ×T_s× random int number in [0,100000)
        schedule event e[id] at crttime+backoff
        unscheduled[id] = false
    end
else // 3rd or subsequent packet
    if unscheduled[id] == false then
        unschedule event e[id]
        unscheduled[id] = true
    end
end

At execution of event e[id]:
  send packet data[id]
```

Figure 6: Ray tracing forwarding algorithm, basic version.

To Forward) and CTF (Clear To Forward). These packets need to contain the transmitter nodeid (similar to the source MAC address in traditional networks), and, for CTF packets, the receiver nodeid (similar to the destination MAC address). The difference with the basic algorithm is the following. Whereas in the basic algorithm, the current node (e.g. C) simply forwards the DATA packet, in the current version it instead sends an RTF packet (after waiting for backoff, as before), then the transmitter of the DATA (i.e. B) answers with a CTF packet (approving that C is the next selected forwarder), and finally the next node (e.g. C) forwards the DATA packet. Note that these packets are similar to RTS/CTS packets in Wi-Fi, used to avoid frame collisions introduced by the hidden node problem [23]. However, a notable difference is that whereas in Wi-Fi the RTS packet is sent by the *sending* node and CTS is sent by the *next* node, in the ray tracing algorithm RTF is sent by the *next* node and CTF by the *sending* (current) node to prevent hidden forwarding nodes and double propagation.

The improved algorithm is presented in Fig. 7 and is explained in the following.

At the reception of the data packet, like in the basic version: if it is the first copy, memorize the data slot; if it is the second copy and also collinear then schedule RTF after some backoff; finally, if it is the third or more copy, node (like $C2$) unschedules the RTF.

Input:
timefirst[id] = null
rtfSender[id] = null

Upon reception of data packet data[id]:
identical to the basic version, Fig. 6

At execution of event e[id]:
send packet RTF[id]

Upon reception of packet RTF[id]:
if has sent a data[id] packet **and** rtfSender[id] == null **then**
rtfSender[id] = sender of the RTF[id] packet
send CTF[id, rtfSender[id]] packet
end

Upon reception of packet CTF[id]:
// ignore it if less than 2 data packets are received
if timesecond[id] == null **then**
return
end
if destination of CTF[id] packet == me **then**
// use a backoff which ensures that the packet is sent in the same T_p
backoff = T_s – (crttime – timesecond[id] + T_p)%T_s
schedule event edata[id] at crttime+backoff
else
if unscheduled[id] == false **then**
unscheduled[id] = true
end
end

At execution of event edata[id]:
send packet data[id]

Figure 7: Ray tracing forwarding algorithm, RTF/CTF improvement.

At backoff expiration, i.e. execution of event e[id]: node (like C) sends an RTF packet (instead of sending the data packet in the basic algorithm).

At the reception of RTF (i.e. from C), the current node (last forwarder, i.e. B), only if this is the first RTF received after having forwarded the data packet, sends a CTF with the RTF's transmitter (i.e. C) as the destination. This informs all the neighbors that the next forwarder has been elected. All the other nodes (like A, neighbors of B and C) discard the RTF packet. Note that even if two nodes send an RTF at the same or close times, the node will send the CTF only to the first RTF received. This solves the double propagation issue because only one node (i.e. C) will forward.

At the reception of CTF: the packet is ignored if it has not received two data packets. The reason is that a CTF can be received *before* receiving the second data packet, and it needs to be ignored. Indeed, the

normal packet scheduling from the point of view of a node like B is the following:

1. Receives RTF from 0
2. Receives DATA from 0
3. Receives RTF from A
4. Receives CTF from 0
5. Receives DATA from A
6. **Sends RTF**
7. Receives CTF from A
8. **Sends DATA**
9. Receives RTF from C (and possibly others)
10. **Sends CTF**
11. Receives DATA from C
12. Receives RTF from D
13. Receives CTF from C
14. Receives DATA from D

Afterward, the node which is the CTF packet's destination (like C) sends the DATA packet in the corresponding T_p (the same as the DATA receptions); the other nodes unschedule their RTF (because a forwarder has been chosen, as informed by the CTF).

The sending time of RTF and CTF does not matter, what matters is the sending time of the DATA packet, which needs to be at the same T_p as the DATA packet received (ensuring collinearity), similar to the basic algorithm (previous section).

3.2.4. Second improvement: prevent too close forwarders

During simulations, we noticed that sometimes two consecutive forwarders are very close to each other. As will be shown later in Section 4.4.1, this increases the number of forwarders and the forwarding angle (where possible forwarders can be located) taking the risk of direction changes.

To avoid these two drawbacks, we modify the algorithm so that only nodes further than $(1/3)r$ (with r is the transmission range) from the transmitting node can be chosen as forwarders.

This limit can be implemented for example by making senders send an additional bit with a specific smaller power when sending data packets, so that only nodes at a distance less than $(1/3)r$ receive it, and the others do not receive it, i.e. see it as a silence (bit 0). Only the latter nodes (farther than $(1/3)r$) can be potential forwarders. Note that this assumption is very different than an alternative assumption, allowing nodes to get precisely the received signal strength and as such discover precisely the distance between a sender and a receiver (this assumption is much stronger than ours, and is unusable in nanonetworks).

The basic algorithm (Fig. 6) is changed as described in Fig. 8. When the second DATA packet is received, only nodes further than $(1/3)r$ are considered.

It is worthwhile to note that imposing that the distance between two consecutive forwarders is greater than $(1/3)r$ implies that this distance is also smaller than $(2/3)r$. Indeed, looking at Fig. 5 and noticing that node C receives DATA from both nodes B and A, we notice that $|AB| > (1/3)r$ and $|AC| \leq r => |BC| = |AC| - |AB| < r - (1/3)r = (2/3)r$. To conclude, the distance between two consecutive forwarders B and C is in the interval $((1/3)r, (2/3)r)$.

elseif timesecond[id] == null **then**

is changed to:

elseif timesecond[id] == null **and** distance $> (1/3)r$ **then**

Figure 8: Ray tracing forwarding algorithm, avoid too close forwarders improvement compared to the basic algorithm (Figure 6).

3.2.5. Third improvement: rollback algorithm

In this work, the selection of the forwarders and the direction of the propagation is quasi-linear and somewhat random. Therefore, the selected last forwarders sometimes lead to propagation die-out, as presented in Section 1.1. In this case, the last selected forwarders were improper (leading to a deadlock) and the solution can only be found by going back in the sequence of the selected forwarders and trying again with other forwarders.

In Fig. 5, nodes A and B have sent their DATA packet. When propagation goes well, a node like C will be the next forwarder. However, in some cases, as explained in Section 1.1, no forwarding node is found (the forwarding area has no node), and the propagation stops. To avoid this die-out, the ray tracing algorithm implements a ROLLBACK procedure (similar to the classical backtracking algorithm), allowing propagation to get back to the previous node (i.e. A) and try a next node different than B. This coming back is implemented as a rollback to the previous node state (before sending the DATA packet). It works as follows.

First, nodes need to keep track of the number of DATA packets received, in a variable called datacounter for example. The execution of event edata[id] in Fig. 7 needs also to be modified, as described below.

If a node (like B), after sending a DATA packet, does not receive any RTF after the max RTF reception time (the length of the window from which backoffs are chosen), then it implies a possible die-out and triggers the ROLLBACK procedure by sending a FORGET packet. It also prevents itself from sending the same DATA again, e.g. by increasing its datacounter by a high value; this is required in order to prevent it from being selected again and avoid an infinite propagation loop in that region.

All the nodes receiving this FORGET packet (e.g. from B) decrement (i.e. subtract 1) their datacounter (note there is no pending RTF anymore because of B's delay of max reception time above). Additionally, the previous node (e.g. A, which had sent a CTF to the node sending the FORGET packet, e.g. B), sends a FORGET2 packet too. (Note that relying on DATA packets sent is not useful, since not only A, but 0 too sent a DATA packet.)

At reception of a FORGET2 packet all the nodes decrement, as for FORGET packets, their datacounter (again, there is no pending RTF anymore because of B's delay). Contrarily to FORGET packets, no other FORGET2 packet is sent.

After having sent the FORGET2 packet, A resends the DATA packet in the same time slot as the second DATA packet received. The ROLLBACK procedure ends here. A node different than B will reply (by sending an RTF), and become the next forwarder.

The algorithm is presented in Fig. 9.

```
At execution of event edata[id]:
  send packet data[id]
  schedule event eforget[id] at crttime+maxBackoff

Upon reception of packet FORGET[id] or FORGET2[id]:
datacounter[id] – –
if packet == FORGET and rtfSender[id] == sender of the FORGET[id] packet then
    // I sent a CTF to B, so I am node A
    send FORGET2[id] packet
    // use a backoff which ensures that the packet is sent in the same T_p
    backoff = T_s – (crttime – timesecond[id] + T_p)%T_s
    schedule event edata[id] at crttime+backoff
end

At execution of event eforget[id]:
if rtfSender == null then // no RTF received
    send FORGET[id] packet
    // ensure that this node is not eligible again to send DATA, since it led to die-out
    datacounter[id] += 20 // pretend to have seen 20 packets
end
```

Figure 9: Ray tracing forwarding algorithm, ROLLBACK improvement.

It should be noted that the ROLLBACK procedure presented above allows several rollbacks at the same level (i.e. node A), and also at previous levels (i.e. node 0 or previous ones), which we call *cascading rollbacks*. This is because the rolled-back nodes prevented themselves from sending DATA again. The procedure has no limit on the previous level to retreat to, this limit depends only on how long the DATA packet is stored by the node after sending it.

The complete algorithm of ray tracing is more complex than what has been presented here and it can be accessed completely on the Web page allowing to reproduce the results, as shown in Section 4.2.

3.3. Discussion

3.3.1. Forwarding angle and area

The section presents the influence that the forwarding angle and the forwarding area have on the propagation, and how the algorithm we propose copes with it. Figure 10 is used for the calculations. Note that during $1\,T_p$, the pulse (light) travels $(3 \times 10^8\,\mathrm{m/s}) \times (100 \times 10^{-15}\,\mathrm{s}) = 3 \times 10^{-5}\,\mathrm{m} = 30\,\mu\mathrm{m}$. This is shown in the figure with the short orange lines. Now, let us consider A as the source node and B as the steering

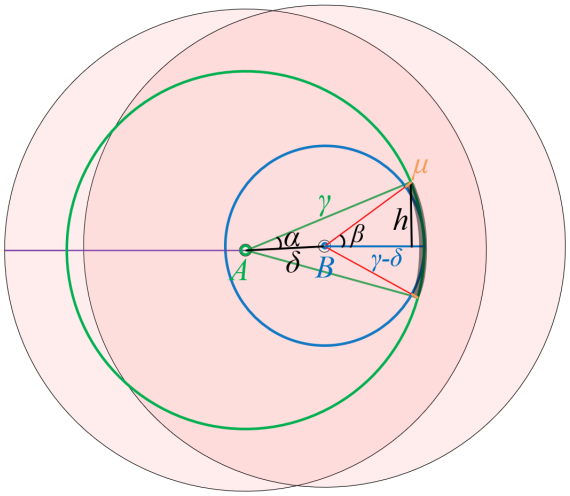


Figure 10: Forwarding angle and area calculation.

node, and the transmission starts from B at the exact T_p modulo T_s that the transmission of A arrived at it, i.e. at the same T_p. The forwarding area, from where the next forwarding nodes can be selected, is given by the overlapping circumferences (the grey arc region in the figure) of the two transmission disks from A (green circle with radius γ) and B (blue circle with radius $\gamma - \delta$) with thickness $\mu \sim 30\,\mu$m (i.e., traveling length of $1\,T_p$). This grey arc expands as the two transmission disks of A and B propagate simultaneously until reaching the maximum transmission range r of node A. In the figure, 2β is called the *forwarding angle* from node B at the shown transmission range of node B. The distance between nodes A and B is δ. First, to calculate the α angle in the figure, we use the cosine rule on the triangle with angle α and sides of γ, δ, and $(\gamma - \delta + \mu)$ as follows:

$$\cos\alpha \quad = \quad \frac{\gamma^2 + \delta^2 - (\gamma - \delta + \mu)^2}{2\gamma\delta}.$$

This gives us the following:

$$\alpha \quad = \quad \arccos\left(\frac{2\gamma\delta + 2\delta\mu - 2\gamma\mu - \mu^2}{2\gamma\delta}\right). \tag{1}$$

Next, based on the figure, we have:

$$h \quad = \quad \gamma \sin(\alpha). \tag{2}$$

By plugging α from (1), we will have:

$$h \quad = \quad \gamma\sqrt{1 - \left(\frac{2\gamma\delta + 2\delta\mu - 2\gamma\mu - \mu^2}{2\gamma\delta}\right)^2}. \tag{3}$$

Based on the figure, we can see:

$$\sin(\beta) \quad = \quad \frac{h}{\gamma - \delta + \mu}.$$

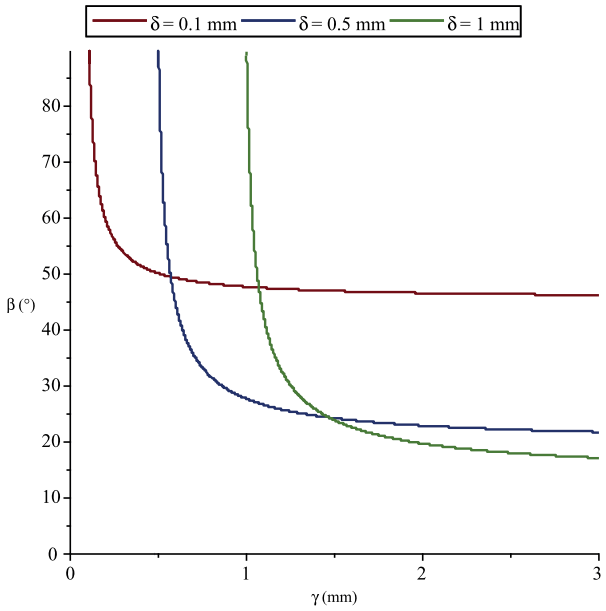


Figure 11: Values of the half of the forwarding angle for different δ.

By plugging h from (3), we will have β as follows:

$$\beta \quad = \quad \arcsin\left(\frac{\gamma\sqrt{1-\left(\frac{2\gamma\delta+2\delta\mu-2\gamma\mu-\mu^2}{2\gamma\delta}\right)^2}}{\gamma-\delta+\mu}\right). \tag{4}$$

Figure 11 shows the values of β (i.e., half of the forwarding angle) for some values of δ. As it can be seen from the figure, β is very high and almost 90° at the start of the propagation (i.e., when γ is very close to δ) and it reduces as δ and γ increase. Therefore, maximum values of the forwarding angle 2β happen with $\delta \to 0$ and $\gamma \to \delta$. This shows that the forwarding angle approaches 180° as the forwarding nodes get closer to each other leading the next forwarder node to be far from the linear path. To reduce this effect, we reduce β by imposing a minimum distance of $(1/3)r$ between the forwarders (Section 3.2.4).

Moreover, by considering β in (4), the surface of the forwarding area can be calculated as:

$$\text{area} = \int_{\delta}^{r} 2(\gamma-\delta)\arcsin\left(\frac{\gamma\sqrt{1-\left(\frac{2\gamma\delta+2\delta\mu-2\gamma\mu-\mu^2}{2\gamma\delta}\right)^2}}{\gamma-\delta+\mu}\right) \mathrm{d}\gamma. \tag{5}$$

Figure 12 shows the percentage of the forwarding area compared to the total communication area (πr^2) in different δ. It can be seen that as δ increases (i.e., the two previous nodes get farther from each other), the forwarding area reduces and the slope of the increase of the area with the increase of r also reduces. Thus, the higher the distance between two forwarders, the smaller the probability of finding the next forwarder, leading, in low node densities, to no potential forwarder, hence die-out. To prevent this, we proposed the rollback mechanism, which undoes the selection of the last forwarder and allows the selection of another forwarder.

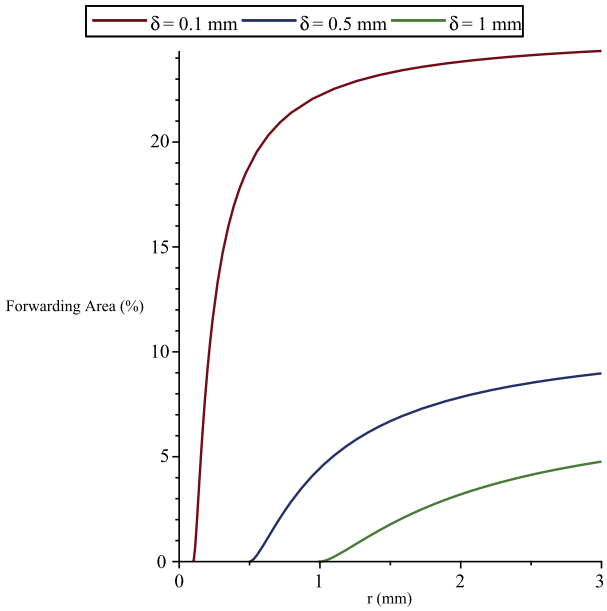


Figure 12: Percentage of the forwarding area to the total area in different δ.

As an example, suppose $r = 1.5\,\text{mm}$ and $\delta = 0.5\,\text{mm}$ (i.e. the two previous nodes are at a distance of $(1/3)r$ each other), then based on Fig. 12, about 6.7% of the neighbors will receive the packet. To avoid die-out, at least one forwarder per hop needs to be found. Given that the forwarder needs to be in the $((1/3)r, (2/3)r)$ range, it means that 3 forwarders per transmission range $(0, r)$ need to be found. Therefore, to avoid die-out in this example, a density of $\lceil 3 \times \frac{100}{6.7} \rceil = 45$ nodes per transmission range at minimum is needed, i.e. nodes must have at least 45 neighbors.

3.3.2. Influence of the number of hops on propagation direction

We recall that each forwarder is chosen randomly, and is found at an angle in the interval $(-\alpha, \alpha)$ (the forwarding angle). On several hops these angles "sum up" and constitute the "error" (the propagation goes sometimes up, sometimes down) compared to a straight line. However, in the long term, because of the central limit theorem (which states that the distribution of random variables tends to a normal distribution, where the average has the highest probability), the highest probability is to go more or less straight (as compared to heavily deviating or to retreat).

4. Evaluation

4.1. Scenario(s)

The evaluation environment is a long and narrow rectangular strip of $100 \times 15\,\text{mm}^2$ resembling human large arteries (e.g., iliac and femoral arteries [24]). The simulation parameters are shown in Table 1. In the environment, 10 000 identical nanonodes are placed randomly using a uniform distribution, plus 2 nanonodes placed manually, as shown below. Node antennas are omnidirectional, and the transmission range r is 3 mm,

Table 1: Simulation parameters.

Parameter	Value
Number of nodes	10 002
Network size	100 mm × 15 mm
Communication range r	3 mm
⇒ Neighbour density	188

i.e. the network is $15/3 = 5r$ high and $100/3 \approx 33r$ wide, large enough to test a routing protocol. Neighbor density (also called node or network density) is $10\,002 * \frac{\pi * 3^2}{100 * 15} \approx 188$ neighbors. The TS-OOK low-level parameters (described in the Introduction) are the ones proposed in the literature, i.e. $T_p = 100$ fs and $\beta = 1000$ (cf. "very short symbol duration T_p (i.e., ≈100 fs)" and "The ratio between the time between pulses and the pulse duration is kept constant and equal to β=1000" [3]).

The sender node, on the left of the network, and the steering node, a bit on its right, are manually placed to start the algorithm. The sender node generates one packet (with a random payload). The steering node retransmits it. Afterward, all the nodes execute the ray tracing algorithm.

The goal of the propagation is that it reaches the right border, i.e. when one of the nodes with abscissa greater than $L - r/10$ (with L the length of the network) receives a DATA packet.

Packet sizes are: 80 bits for DATA packet, 10 bits for RTF, 11 bits for CTF, 12 bits for FORGET, and 13 bits for FORGET2 packets.

We executed 100 different simulations (with seeds used for backoff from 0 to 99) and visually looked at all of them to understand the propagation.

4.2. Validation of the algorithm implementation, and the simulator

Given the high number of nodes involved, real experiments are not possible, hence we use simulations. Moreover, the ray tracing algorithm is complex, and its implementation can be tricky. To validate the implementation of ray tracing and the simulator itself, we present in this section the results of various checks: packet forwarding time, packet exchange, node collinearity, and cascading rollbacks.

4.2.1. Nanonetwork simulator used

Several nanonetwork simulators exist, but only BitSimulator [25] is scalable and allows to simulate more than around one thousand nodes [26]. BitSimulator implements routing and transport protocols, and considers some low-level bit transmission characteristics, such as transmission time and bit-dependent collision at the receiver. It implements TS-OOK modulation. BitSimulator is free software and has been used to validate the results of several articles[1].

[1] http://eugen.dedu.free.fr/bitsimulator

It is extremely important to note that for routing algorithms a *visualization* of the network is of paramount importance. Simulators usually provide statistics at the end of the simulation, and provide no clue about *what* happened in the network (nodes involved in packet routing). With only such data, and without visualization, we would not have discovered for instance the double propagation, nor the propagation stop at borders (shown in Section 1.1). In this respect, BitSimulator is very useful, as it comes with a visualization program, allowing to *see* visually what happens in the network, and providing a visual validation of the simulation.

We implemented the ray tracing forwarding in BitSimulator. The other routing protocols were also implemented in the simulator.

We provide a Web page[2] with the implementation of the algorithm (which is more complex than what is presented in this article), and the code allows us to reproduce all the results of this article.

4.2.2. Validation of the forwarding time

In the scenario, we look for the third sending node, after the source and the steering nodes, which is node 4074. To validate the forwarding time of this node, the relevant lines in the events log file are:

```
1 7911616 4074 0 1000 80 0 0 0 -1 -1 -1
1 15911616 4074 1 1000 80 0 0 0 -1 -1 -1
0 31674111616 4074 4074 1000 80 0 0 0 0 -1 -1 -1
```

The first column informs that the first two lines are for packet receiving (value 1), and the third one is for packet sending (value 0). The second column corresponds to the sending/receiving time. The pulse duration is 100 fs, hence we remove the last two digits, and T_s is 1000 more, hence we keep only the last three remaining digits; otherwise said, we keep the 3rd, 4th, and 5th bits from the right. For all three lines, we get the same time, 116, meaning that the second packet was received in the same T_p as the first one and that the third packet was sent in the same T_p too. Formally, this sending/receiving "normalised" time t can be computed as $\lfloor t/T_p \rfloor \% \beta = \lfloor t/100 \rfloor \% 1000$. This result, 116 on the three lines, validates the packet forwarding time.

4.2.3. Validation of the packet exchange in the network

As described in Section 3.2.3, each potential forwarder, after having received two DATA packets, does not send the DATA packet, but first sends an RTF packet, waits to receive the CTF packet, and only afterward forwards the DATA packet. The first packet exchanges in the original scenario, as shown by the simulator, are the following:

- RTF sent by node 4074
- CTF sent by node 1
- DATA sent by node 4074

[2] http://eugen.dedu.free.fr/bitsimulator/comnet24

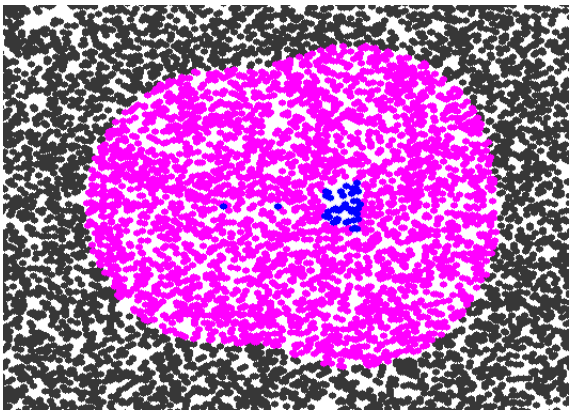


Figure 13: The locus of potential forwarders.

- RTF sent by node 5810
- CTF sent by node 4074
- DATA sent by node 5810
- RTF sent by node 40
- CTF sent by node 5810
- DATA sent by node 40

The above output can be divided into groups of 3 consecutive lines, and each group shows an RTF packet sent (e.g. by node 4074 for the first group, node 5810 for the second group, and node 40 for the third group), the corresponding CTF packet sent by the previous node (e.g. 1, 4074, and 5810 respectively), and finally the DATA packet sent (e.g. by node 4074, 5810, and 40 respectively). This output shows that the packet exchange works as expected.

4.2.4. Validation of the node collinearity check

For this section, we modify the implementation to make appear *all* the potential forwarders, and increase the number of nodes by 10 times to have a more complete (clearer) shape of the locus.

The simulation results for the first hop (after the steering node sends the DATA packet) are shown in Fig. 13. The blue nodes are potential forwarders (remember that forwarding nodes are in $((1/3)r, (2/3)r)$ range, cf. Section 3.2.4), and their placement (giving the forwarder area) corresponds to the illustration in Fig. 5. This similarity verifies the computation of the node collinearity of the simulator.

4.2.5. Validation of cascading rollbacks

As explained in Section 3.2.5, rollbacks can go back several levels. Figure 14 presents such an example, where the bottom nodes are close to the border. The node number represents the order in which nodes send their first DATA packet. In the following, only the packets needed for understanding are presented.

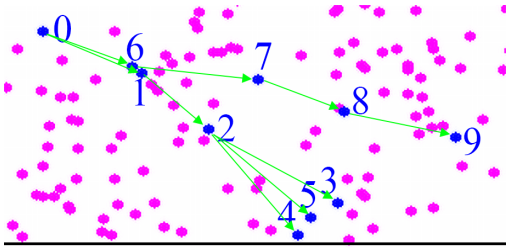


Figure 14: Illustration of cascading rollback.

First, the DATA packet is forwarded, in order, by nodes 0, 1, 2 and 3. Node 3, like nodes 4 and 5, is close to the bottom border, hence it stops propagation (i.e. when it sends the DATA packet, no node answers with RTF). Therefore, it sends a FORGET packet, and the previous node 2 sends a FORGET2 packet and resends the DATA packet. Another node answers, node 4, which stops again the propagation. After FORGET and FORGET2 packets, node 5 sends the DATA packet, and it stops again. After FORGET and FORGET2 packets, node 2 resends the DATA packet, and this time no node answers with RTF, because the three potential forwarders (3, 4 and 5) already answered. Node 2 then sends a FORGET packet, and the previous node 1 sends a FORGET2 packet, followed by the DATA packet. Thus, the rollback procedure goes back one additional level (cascading rollback), from node 2 back to node 1. Node 1 has no other potential forwarder (besides 2), hence it goes back once more, to node 0. This time node 6 is chosen, and the propagation resumes successfully with nodes 7, 8, and 9, in order.

This simulation proves also that a node is not chosen twice, otherwise, the rollback procedure would not go backward (from node 2 back to node 1 for example).

4.3. Comparison with related forwarding methods

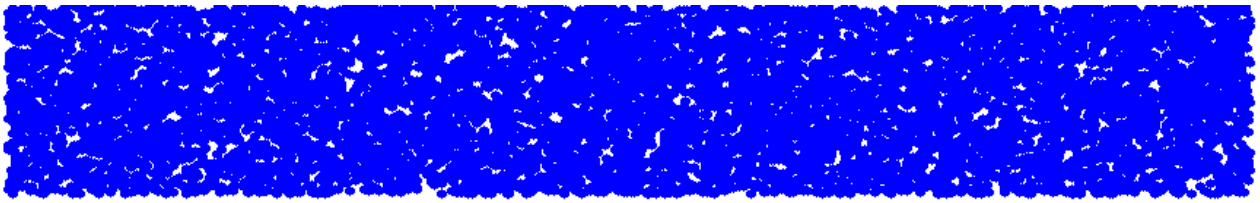
Similarly to the previous article [6], we compare ray tracing with several protocols. The ray tracing method does not use any coordinates. We therefore compare it with coordinate-free protocols and with coordinate-based protocols.

4.3.1. Comparison with coordinate-free methods

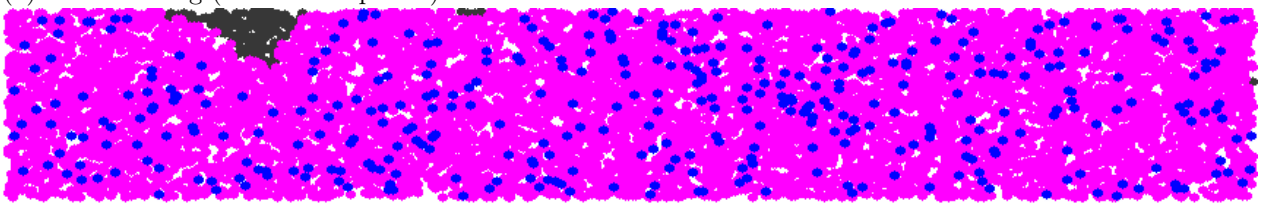
The only coordinate-free methods we found appropriate to nanonetworks are flooding protocols: pure flooding, probabilistic flooding (with probability of 4 %, the minimum value which still makes propagation reach the right border), and the highly optimized auto-adaptive backoff flooding, all of them described in Related work (Sect. 2). We compare the ray tracing method with them.

Fig. 15 shows for all the protocols compared the position of the sender for flooding and the directed routing for ray tracing during packet propagation. It can be seen that the ray tracing has the fewest number of senders and receptions.

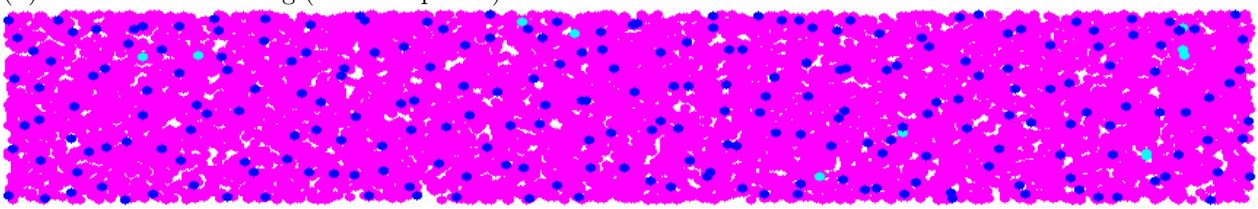
Table 2 presents the number of forwarding nodes and of *packet receptions* (a node receiving two packets is counted as two packet receptions) during the simulation. It can be seen that ray tracing drastically reduces



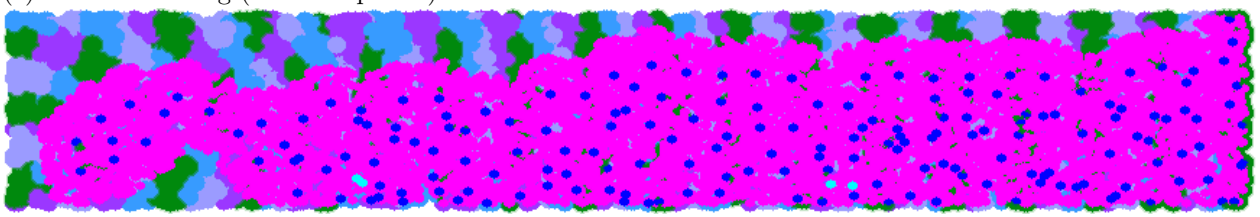
(a) Pure flooding (10 001 blue points)



(b) Probabilistic flooding (385 blue points)



(c) Backoff flooding (237 blue points)



(d) Counter-based SLR (183 blue points)

(e) Ray tracing (110 blue points)

Figure 15: VisualTracer capture of the compared methods, showing forwarders (blue nodes) and receivers (magenta nodes).

the number of forwarders (to 110) and receptions. It outperforms the highly optimized backoff flooding; the reason is that it has a quasistraight forwarding by avoiding top and bottom borders, whereas backoff flooding floods the network.

4.3.2. Comparison with coordinate-based methods

SLR is a destination-oriented routing protocol proposed for nanonetworks [27], which uses a setup phase where nodes assign coordinates. We note that it can be improved by combining it with backoff flooding

Table 2: Comparison of the ray tracing with related flooding and destination-oriented routing methods.

Method	Forwarding nodes	Packet receptions
Pure flooding	10001 (100 %)	1 702 740
Probabilistic flooding	385 (3.8 %)	66 272
Backoff flooding	237 (2.3 %)	38 095
Counter-based SLR	183 (1.8 %)	31 333
Ray tracing	110 (1.1 %)	4 692

(called "counter-based SLR" in the following). Both SLR and backoff flooding are described in Related work (Sect. 2). We compare the ray tracing method only with counter-based SLR, given that it is much improved compared to the original SLR (an example of results is given in the original ray tracing article [6]).

In the scenario, we manually select a node on the right of the network as the destination. Also, for a fairer comparison, we do not count the packet exchanges during the setup phase for the counter-based SLR method, because this phase is executed only once, at the network setup.

The propagation is shown in Fig. 15 (showing the SLR zones too), and the results are provided in Table 2. Ray tracing (110 forwarders) outperforms counter-based SLR (183 forwarders). The reason is that ray tracing is more linear than counter-based SLR. Thus, even if the latter is quite optimized, ray tracing reduces even more the number of forwarders and receptions.

To conclude, ray tracing outperforms both related coordinate-free and coordinate-based methods. Additionally, the latter methods need an additional step at network deployment (to create the coordinate system), which might lead to issues in nonconvex or very long environments such as blood vessels. Instead, ray tracing does not need any coordinate system, but only a direction (subject to constraints described in the following sections).

4.4. Inspection

4.4.1. Usefulness of $(1/3)r$ limitation

When removing the limitation of the next node being at a distance greater than $(1/3)r$ (presented in Section 3.2.4), two consecutive forwarding nodes may be very close to each other as shown in Fig. 16. This has two consequences.

First, two close nodes make a wider forwarding angle (cf. Section 3.3.1), as shown in Fig. 16, increasing the risk of inversion (propagation going towards left). Figure 17 shows such an example on the base scenario.

Second, two close nodes reduce the forwarding distance requiring a higher number of forwarders for the propagation (cf. Section 3.3.1). For example, in the same ten scenarios (differing by the random seed for backoffs), the number of forwarders is 126 with the limitation and 164 without the limitation.

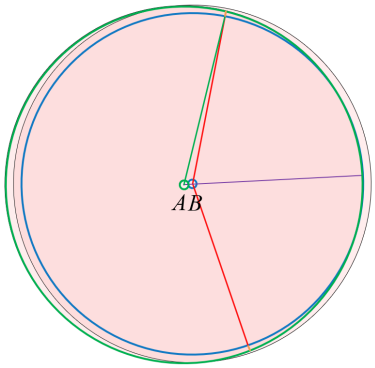


Figure 16: Forwarding angle with very close forwarding nodes.

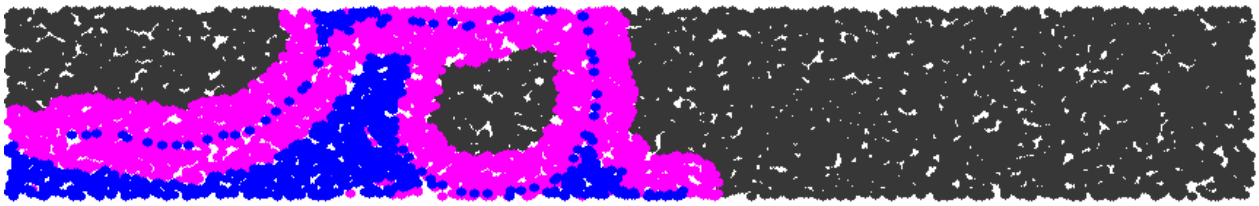


Figure 17: Removing the $(1/3)r$ limitation may lead to direction changing in some instances.

4.4.2. Heterogeneous networks

In this section, the scenario is changed as follows. Instead of placing 10 000 nodes, the environment is divided into three equal vertical zones: in the left zone 10 000 nodes are randomly placed using a uniform distribution, in the middle zone 2500 nodes using a uniform distribution (accounting for a less dense zone), and in the right zone 7500 nodes using a normal (Gaussian) distribution. As shown in Fig. 18 (giving one example), the propagation goes through the middle zone, where it generates some failed attempts to propagate, enters the right zone, and eventually reaches the right border.

This representative example shows that the network heterogeneity does not have any influence on the results of the ray tracing algorithm.

4.4.3. Influence of environment size and transmission range

The results of the ray tracing algorithm are sensible to the environment size. On the one hand, if network space is too narrow, e.g. less than around $2r$ (i.e. 2 hops) in our scenario, then backoff flooding (explained in

Figure 18: Ray tracing algorithm in a heterogeneous network.

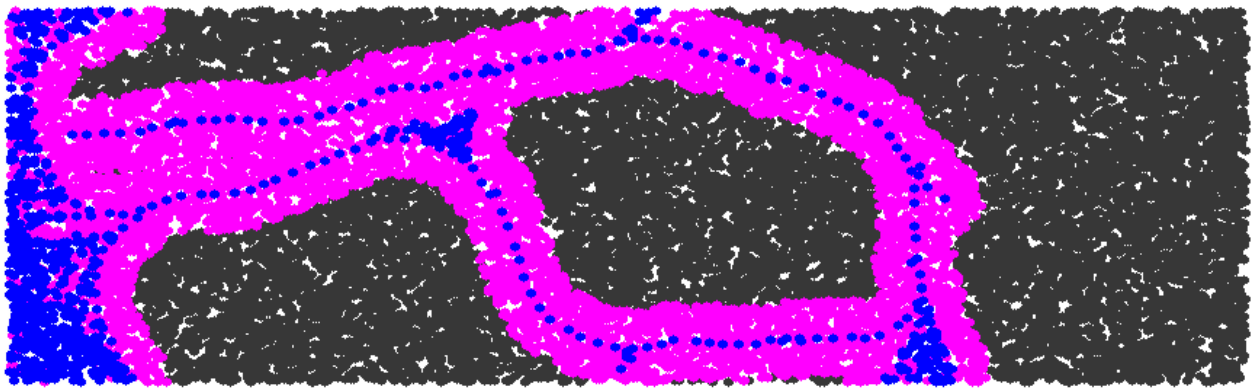


Figure 19: If the strip is too wide, the propagation can turn backward in some instances.

Section 4.3), where all the nodes receive the packet, has better results and is simpler to implement. On the other hand, if network space is too wide, e.g. more than around $10r$ (i.e. 10 hops) in our scenario, backoff flooding has inferior results (since it is too verbose as all the nodes need to receive the packet), but ray tracing faces a major problem with the direction. In the following, we discuss this problem.

We remember that the goal is that the propagation from node to node reaches the right border (cf. Section 4). However, the direction taken by the propagation depends on each hop on the position of the next forwarder. This forwarder is chosen *randomly*, as it is the first node to send the RTF packet, which is the node having chosen the smallest backoff. For example, Fig. 2 shows that the propagation goes sometimes up, sometimes down. In a wide environment, after some hops, the propagation can even change direction and go back (return) towards the left border. Such an example, with a double width (30 mm, i.e. $10r$) and a double number of nodes (20 000) compared to the base scenario, is shown in Fig. 19. Note that this issue does not happen frequently in this scenario, but due to the randomness of selecting the next forwarder, the chance of going back (return) increases in wider environments. This can be reduced by reducing the forwarding angle, hence the use of the $(1/3)r$ limitation. However, a very high reduction of the angle also reduces the forwarding area, potentially leading to the activation of the ROLLBACK procedure.

The direction inversion does not depend on strip width only, but also on the transmission range. Indeed, a higher range means further next forwarders and consequently a lower forwarding angle (where potential forwarders are), and a smaller probability of finding a forwarder, and an increasing need to rollback. A smaller radius translates into a higher zone, an increased probability of propagation, but a higher probability of direction inversion towards the left border.

5. Application

The proposed ray tracing routing has some features useful in applications. It is a *coordinate-free* protocol and does not need a setup phase, unlike coordinate-based protocols such as SLR [27] which require a setup phase before being able to route the data. Coordinate-free protocols can be used in dynamic environments

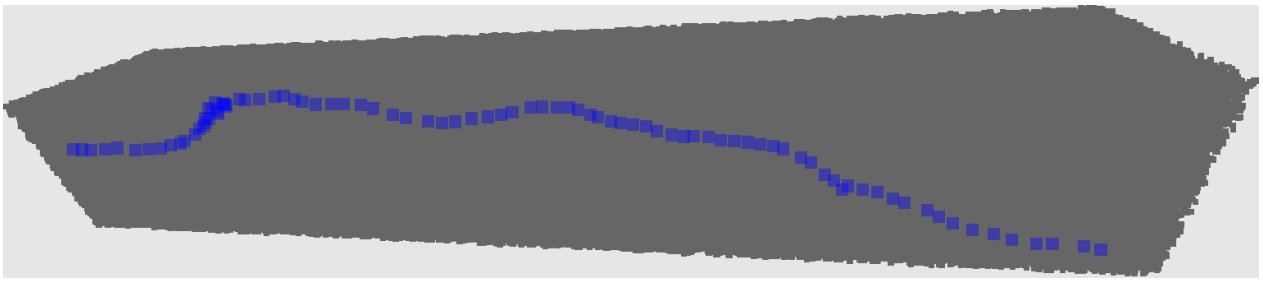


Figure 20: Ray tracing algorithm in 3D (senders are in blue, the other nodes in grey).

with mobile nodes such as the human cardiovascular system because each node tries to forward its data packet to the destination without knowing the data path. Ray tracing is also a *stateless* protocol (except for keeping the id of already seen packets in nodes); note that the rollback procedure is stateless too if nodes piggyback the data payload in FORGET and FORGET2 packets. In comparison, SLR is stateful, since it requires nodes to always keep coordinates acquired at the network setup.

Since the blood vessels are 3D environments, we are interested to see if the ray tracing algorithm works in 3D environments too. We note that, compared to 2D, in 3D the forwarding region enlarges in the third dimension, and the forwarding angle makes a cone shape (more precisely a difference of two cones, i.e. a truncated cone) forwarding region instead of the 2D triangle (more precisely a difference of two triangles, i.e. an isosceles trapezoid) that has been shown in the previous sections.

The result is given in Fig. 20, and shows that the ray tracing works in 3D environments too albeit requiring much more nodes than in 2D. For example, in this figure, the node density is 44 nodes/mm^3 which is much less than the average red blood cell counts of an adult human around 5.1×10^6 cells/mm^3 [28]. Therefore, the considered high node density is expected for cellular level monitoring and detection.

In Fig. 21, we envision an application for ray tracing routing in the human cardiovascular system. Consider node A has sensed an event (e.g., a cancerous cell) that needs to be notified to the gateway and further to the outside world. Then, node A randomly selects a steering node B and starts a multi-hop collinearity ray tracing toward one end of the blood vessel. As shown in the figure, there is a gateway placed at one end of the blood vessel with more resources, and especially with a high receiving sensitivity, which can capture the packet sent by the nodes in its vicinity (i.e., the strip width). Upon reception of a packet by node F in its vicinity, it sends a high-power packet, received by all the nodes in its vicinity, informing them to stop further propagation, as the gateway has received the packet. The gateway will transfer the received packet to a macro world wireless protocol (such as Wi-Fi or Bluetooth) to take the appropriate action. However, since node A does not know the location of the gateway and selects the steering node B randomly, the propagation might go in the opposite direction to the gateway. To solve this issue, a reverse direction forwarding from node A needs also to be started. This is done by using the initial steering node B as the source node and the primary source node A as the steering node of the reverse flow. Therefore, two data flows will be initiated from the primary source node A in opposite directions and only one of them will reach the gateway. In order

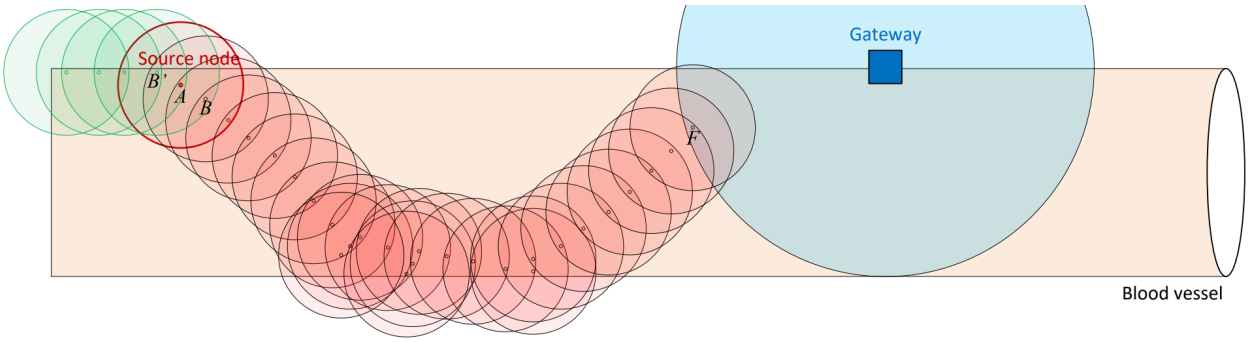


Figure 21: An application of the ray tracing routing in the human cardiovascular system.

to stop the other flow, which does not reach the gateway anyway, a TTL-like max hop count limit can be used, stopping it after enough retransmissions.

6. Conclusions and perspectives

In a previous paper, we introduced the ray tracing routing protocol, aiming to send packets in a quasistraight line. This protocol was designed for dense nanonetworks, where there are numerous nodes with scarce resources. It turned out to be more efficient than related protocols, especially in terms of number of forwarders used. Despite its efficiency, after a thorough analysis, several issues with the protocol have been noticed, such as double propagation, propagation die-out, and large forwarding angle.

In the current paper, we have proposed a major enhancement of the ray tracing protocol. We have included new control packets: RTF and CTF to fix the double propagation issue, FORGET and FORGET2 to fix the die-out, and a distance limitation to reduce the forwarding angle.

Extensive simulations, analyzed through both statistics and visual inspection, have shown not only that packet reception times in TS-OOK can successfully be used for linear routing, but also that in the long and narrow nanonetworks, the enhanced ray tracing protocol proposed outperforms in terms of the number of forwarders and receptions all the related methods without relying on a coordinate system.

We have envisioned an application of ray tracing in the human cardiovascular system, where the information about the events sensed by the nanonodes will eventually reach, through blood vessels and using three-dimensional ray tracing, a gateway connected to the external world.

Perspective work includes node mobility, where the ray tracing is expected to work (since it does not have a setup phase and the forwarding decisions are made by the last three nodes in the forwarding chain), and investigation in lossy networks (through a shadowing propagation model for example).

CRediT authorship contribution statement

Eugen Dedu: Conception and design of study, Methodology, Software, Acquisition of data, Analysis and/or interpretation of data, Writing – original draft, Writing – review & editing. **Masoud Asghari**:

Conception and design of study, Methodology, Analysis and/or interpretation of data, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

As specified in the paper, all the results are fully reproducible, and all the related information is provided on the Web page indicated (Section 4.2).

Acknowledgments

This work has been funded by Pays de Montbéliard Agglomération (France).

References

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

[2] F. Lemic, S. Abadal, W. Tavernier, P. Stroobant, D. Colle, E. Alarcón, J. Marquez-Barja, and J. Famaey, "Survey on terahertz nanocommunication and networking: A top-down perspective," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 6, pp. 1506–1543, 2021.

[3] 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.

[4] M. Asghari, "Intrabody hybrid perpetual nanonetworks based on simultaneous wired and wireless nanocommunications," *Nano Communication Networks*, vol. 32, p. 100406, 2022.

[5] M. Asghari, "Integration of semi-permanent wired clusters into intrabody wireless perpetual nanonetworks," *Telecommunication Systems*, vol. 84, no. 3, pp. 285–301, 2023.

[6] E. Dedu and M. Asghari, "Time-based ray tracing forwarding in dense nanonetworks," in *37th International Conference on Advanced Information Networking and Applications (AINA)*. Juiz de Fora, Brazil: Springer, Mar. 2023, pp. 497–508.

[7] 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, 2015.

[8] T. Arrabal, D. Dhoutaut, and E. Dedu, "Efficient density estimation algorithm for ultra dense wireless networks," in *Int. Conf. on Computer Comm. and Networks (ICCCN)*, 2018, pp. 1–9.

[9] N. Abu-Ghazaleh, K.-D. Kang, and K. Liu, "Towards resilient geographic routing in wsns," in *Int. Workshop on Quality of service&security in wireless and mobile networks*, 2005, pp. 71–78.

[10] N. Bulusu, J. Heidemann, and D. Estrin, "Gps-less low-cost outdoor localization for very small devices," *IEEE Personal Communications*, vol. 7, no. 5, pp. 28–34, 2000.

[11] Z. Hossain, Q. Xia, and J. M. Jornet, "Terasim: An ns-3 extension to simulate terahertz-band communication networks," *Nano Communication Networks*, vol. 17, pp. 36–44, 2018.

[12] 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, Oct. 2011.

[13] S. R. Neupane, "Routing in resource constrained sensor nanonetworks," Master's thesis, Tampere University of Technology, Finland, 2014.

[14] C. Liaskos, A. Tsioliaridou, S. Ioannidis, N. Kantartzis, and A. Pitsillides, "A deployable routing system for nanonetworks," in *IEEE Int. Conf. on Communications (ICC)*, 2016, pp. 1–6.

[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, 2017.

[16] C.-F. Chou, J.-J. Su, and C.-Y. Chen, "Straight line routing for wireless sensor networks," in *10th IEEE Symposium on Computers and Communications (ISCC'05)*. IEEE, 2005, pp. 110–115.

[17] H.-H. Liu, J.-J. Su, and C.-F. Chou, "On energy-efficient straight-line routing protocol for wireless sensor networks," *IEEE Systems Journal*, vol. 11, no. 4, pp. 2374–2382, 2017.

[18] M.-T. Sun, K. Sakai, B. R. Hamilton, W.-S. Ku, and X. Ma, "G-star: Geometric stateless routing for 3-d wireless sensor networks," *Ad Hoc Networks*, vol. 9, no. 3, pp. 341–354, 2011.

[19] F. Hoteit, E. Dedu, W. K. Seah, and D. Dhoutaut, "Ring-based forwarder selection to improve packet delivery in ultra-dense networks," in *IEEE Wireless Communications and Networking Conference (WCNC)*, Austin, TX, USA, Apr. 2022, pp. 2709–2714.

[20] F. Hoteit, D. Dhoutaut, W. K. Seah, and E. Dedu, "Expanded ring-based forwarder selection to improve packet delivery in ultra-dense nanonetworks," in *18th International Wireless Communications and Mobile Computing Conference (IWCMC)*, Dubrovnik, Croatia, May-Jun. 2022, pp. 1406–1410.

[21] F. Hoteit, E. Dedu, D. Dhoutaut, and W. K. Seah, "3d dynamic ring-based forwarder selection to improve packet delivery in ultra-dense nanonetworks," *Nano Communication Networks*, vol. 39, p. 100492, 2024.

[22] J. M. Jornet and I. F. Akyildiz, "Information capacity of pulse-based wireless nanosensor networks," in *2011 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*. IEEE, 2011, pp. 80–88.

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

[24] C. Vlachopoulos, M. O'Rourke, and W. W. Nichols, *McDonald's blood flow in arteries: theoretical, experimental and clinical principles*. CRC press, 2011.

[25] 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.

[26] E. Sahin, O. Dagdeviren, and M. A. Akkas, "An evaluation of internet of nano-things simulators," in *6th International Conference on Computer Science and Engineering (UBMK)*, Ankara, Turkey, Sep. 2021, pp. 670–675.

[27] 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.

[28] L. R. Dixon, "The complete blood count: physiologic basis and clinical usage," *The Journal of perinatal & neonatal nursing*, vol. 11, no. 3, pp. 1–18, 1997.