BitSimulator, an electromagnetic nanonetworks simulator

Dominique Dhoutaut
dominique.dhoutaut@univ-fcomte.fr
FEMTO-ST Institute, Univ. Bourgogne Franche-Comté, CNRS
Montbéliard, France

Thierry Arrabal
thierry.arrabal@univ-fcomte.fr
FEMTO-ST Institute, Univ. Bourgogne Franche-Comté, CNRS
Montbéliard, France

Eugen Dedu
eugen.dedu@univ-fcomte.fr
FEMTO-ST Institute, Univ. Bourgogne Franche-Comté, CNRS
Montbéliard, France

ABSTRACT

Electromagnetic nanonetworks form an exciting new field of investigation. The frequency band and the channel access methods impose peculiar constraints on upper layer protocols. Given that research is impaired by the unavailability of ready to use hardware, simulation tools are needed to evaluate protocols and applications. This paper presents BitSimulator, a network simulator specifically targeting wireless nanonetworks. It is designed to accurately simulate collisions up to the bit level, allowing studies on coding, channel access, routing or congestion control. Being dedicated to this specific environment, it is highly optimized for speed and supports a large number of nodes.

CCS CONCEPTS

• **Networks → Network simulations**; • **Computing methodologies → Simulation tools**;

ACM Reference Format:
Dominique Dhoutaut, Thierry Arrabal, and Eugen Dedu. 2018. BitSimulator, an electromagnetic nanonetworks simulator. In *NANOCOM '18: NANOCOM '18: ACM The Fifth Annual International Conference on Nanoscale Computing and Communication, September 5–7, 2018, Reykjavik, Iceland.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3233188.3233205

1 INTRODUCTION

Electromagnetic nanonetworks represent a new field of research, where thousands or millions of tiny and simple devices communicate and interact. Many challenges exist in this field, especially as the nanometric scale imposes strong hardware restrictions. To cope with the physical and environmental specificities of nanonetworks (CPU, memory and energy are extremely limited), a refoundation of the whole network stack is required, from channel access and coding to routing and applications. On one hand, progress has already been made on electromagnetic channel modeling and access. In [5], Time Spread On-Off Keying (TS-OOK) has been proposed. It allows communications using extremely short electromagnetic pulses, as they can be generated by tiny antennas, and can be detected and processed with limited computation power. On the other hand, much has yet to be done at the higher network levels.

As complete implementations of nanonetworking devices are not yet available, work on network protocols, coding and applications has to be conducted either analytically or by simulation.

NANOCOM '18, September 5–7, 2018, Reykjavik, Iceland

ACM ISBN 978-1-4503-5711-1/18/09…$15.00
https://doi.org/10.1145/3233188.3233205

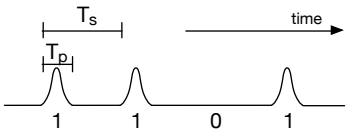


Figure 1: Modulating "0" and "1" in TS-OOK.

In this paper we present **BitSimulator**[1], a simulation software dedicated to electromagnetic nanonetworks. It has been developed for and is especially suited to help researchers experiment with and better understand wireless nanonetworks protocols.

2 CONTEXT AND DESIRABLE FEATURES

2.1 Electromagnetic nanonetworks specificities

Due to the tiny energy available and to the very small antennas, modulations from usual wireless networks, such as 802.11, cannot be used. A simple scheme has therefore been proposed, TS-OOK [5], which envisions extremely short radio pulses, as short as 100 fs (femtoseconds), guided by a very precise clock. Upon reception, a pulse is then simply interpreted as a binary "1" and its absence as a binary "0". Only a few values are required to communicate: The duration of a pulse T_p, a power reception threshold above which a "1" bit is considered received, and the symbol duration T_s (the time between two consecutive bits). The spreading ratio $\beta = T_s/T_p$ directly influences communication speed. The transmission of four consecutive bits is given as an example on Figure 1. Please note that the scale and the shape on this figure and the following ones is altered for readability reasons. Pulses are represented here as simple "peaks" and the delay between two consecutive pulses is expected to be much greater than shown (β ranging from a few hundreds to many thousands).

When $T_s \gg T_p$, this modulation leads to frame multiplexing over time. This concept is represented on Figure 2, where pulses from multiple frames are interleaved. Indeed, even if able to send extremely short pulses, an individual node is not expected to send them very fast (mainly because of energy and computation constraints). Consequently, an individual frame cannot be sent at an extremely high speed. But in a dense environment, the aggregated throughput of many multiplexed frames can reach very high values. This multiplexing ability is very different from traditionnal wireless networks where frames are sent sequentially. Here, with large values of β, hundreds of frames could be over the air at the *same* time.

[1]The simulator is available under GPL licence, http://eugen.dedu.free.fr/bitsimulator.

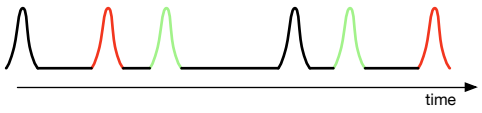


Figure 2: Temporally multiplexing frames in TS-OOK.

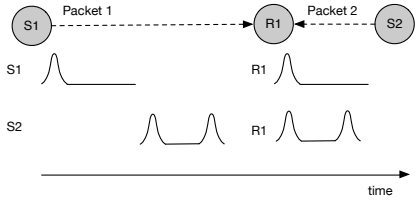


Figure 3: In TS-OOK, propagation delay influences packet arrival order and collisions.

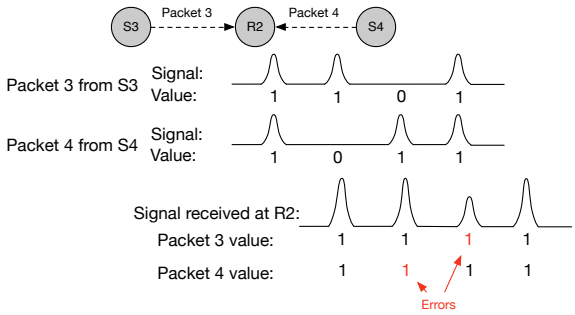


Figure 4: Incorrect reception with TS-OOK.

The extremely short duration of pulses brings another peculiarity: radio propagation delay is not negligible anymore, even over distances as short as a few millimeters. This delay can be much longer than the duration of a pulse and confuse the reception. Figure 3 shows a topology where S1 and S2 generate packets at different times, but they arrive at R1 at the same time. Especially in dense networks with many transmitters in range but located at various distances, this means receiving nodes will experiment differences in the order of the arriving bits. In particular, depending on the relative positions of the nodes, it will cause bits to overlap at some neighboring nodes and not at others.

Encoding bits as pulses also brings specificities to the effect of collisions. We consider a simplified model for pulses here. Indeed, two overlapping bits do not necessarily cause an error. No error occurs when the frame currently being tracked contains a "1" bit, and at the time of its reception a "1" bit from another frame arrives, since the power level over the channel is above the reception threshold anyway and the receiver considers that it received a "1". "0" bits neither do not generate errors, since they are silence. To conclude, collisions cause errors if a "0" was sent but a "1" arrives at the same time. This is illustrated on Figure 4, where frames concurrently transmitted by senders S3 and S4 collide at receiver R2: only "0" bits of both frames are altered.

2.2 Desirable simulator features

Researchers often use in-house simulation software developed as a side project. However, these tools often have flaws because of the small amount of time invested. In particular, these tools fail to find the correct balance between complexity, accuracy and requirements (CPU and memory). This has implications on simulation correctness. For our purposes, a simulator needs to have the following primary features.

Individual node and application instantiation. As stated in the introduction, one needs a tool to help design, simulate and validate network protocols and applications. Analytical work considering the network as a whole is often not practicable (when network stack or scenario is too complex) or not sufficiently detailed (to capture subtleties and special cases). An implementation of the whole network stack is usually needed, with an individual instantiation of each element. Each node and each piece of code running is treated separately.

Bit by bit transmission and error computation. As presented in the previous section, mechanisms affecting the bit error rate but also the distribution of errors heavily depend on the coding and the payload itself. Errors need to be correctly simulated, especially when working on coding schemes.

Radio propagation delay consideration. Small changes in the position or timing in the simulated nodes significantly affect bits effectively received and collisions. Channel access control protocols such as [10] use specific binary frame preambles and compute optimal inter-bits spacing. Those protocols significantly reduce the risk of collisions but cannot rule them out, especially in very high density scenarios. Correctly simulating frame's individual bits (cf. previous desirable feature), and the timing and scheduling of events (including the propagation delay) cannot be neglected at this scale.

(Numerous) frames multiplexing over the channel. This is a defining feature of wireless nanocommunications, where numerous frames (possibly hundreds or more) can be interleaved in the air. This implies the ability for nodes to decode multiple frames in parallel. This is technically possible, but the number of frames concurrently decoded has to be limited to take into account hardware or software constraints.

Scaling to large number of nodes. Applications of wireless nanocommunications include programmable matter and sensor networks, which can have numerous nodes (e.g. millions). The simulator needs to be scalable with respect to the number of nodes.

Simulation speed. To avoid particular cases and unpredictable variations, a scenario is often run multiple times with different random number generator (RNG) seeds, which greatly increases simulation time. An optimized software is therefore desirable. It should be noted that fine-grained parallelization is often useless due to the interdependence of the simulated events.

3 COMPARISON WITH OTHER SIMULATORS

3.1 Network Simulator and Nano-Sim

Network simulator has been around for almost thirty years. NS-2 and now NS-3 have been widely used in the literature. NS-3[2] is a very versatile tool and can simulate virtually any kind of network

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

(from satellites to Wi-Fi, Zigbee or wireless nanonetwork). NS-3 implements complete protocol stacks with many physical, link, routing, control and application layers.

NS-3 includes a nanonetwork layer, Nano-Sim [6], which targets pulse-based wireless nanonetworks and follows the TS-OOK model mentioned earlier. Nano-Sim allows packet multiplexing in the channel (many packets being sent in parallel, as long as their pulses are interleaved over time and do not overlap). However, the model implemented is too simple:

- Propagation delay is not taken into account: when a packet is sent, all nodes in range start receiving it at the same time.
- Collision detection considers that all packets use the same β value. It is an all-or-nothing computation, i.e. if a bit from a packet collide with a bit from another packet, then all the other bits collide too.
- Bit overlapping always leads to collision, no matter their value ("0" or "1").
- Packets are reordered without reason, and jitter does not vary, which is unrealistic [2].

Moreover, the strong versatility of NS-3 comes with a drawback in terms of CPU and memory overhead.

3.2 Vouivre

Vouivre [1] is a nanonetwork simulation library that can also be used in standalone mode. Because of molecular absorption, the channel tends to produce noise when excited, namely when sending "1" bits [3]. It allows to simulate extremely dense networks with millions of nodes. Using a statistical error computation algorithm from [4], it supports a large number of nodes (tens of thousands of neighbors) while remaining quite fast. The main drawback of this statistical approach is the inability to simulate in detail the effect of the packet payload.

3.3 Physical simulators

Very low level approaches use a very detailed physical model running in a generic simulator. In [7], COMSOL Multiphysics[3] is used to compute the behavior of a graphene-based nanoantenna. [9] uses AnyLogic[4] with raytracing.

These tools produce very accurate results, but have two major drawbacks for our purposes. They involve a complex setup phase where the environment has to be described in detail. Also, depending on the compromises (typically, if the actual payload is simulated or not), the computations can be very heavy and in practice can be used only in networks with a few nodes.

4 BITSIMULATOR DESIGN

Because of the limitations of the available tools, we choose to develop a dedicated simulator. It aims a good tradeoff between level of detail, accuracy and execution speed. By targeting only one type of network, we can simplify the design and get rid of a lot overhead (found for example in NS-3). Almost all the code is nano-wireless specific, and this makes developing analysis and visualization tools equally easier.

[3]https://www.comsol.com
[4]https://www.anylogic.com

This section gives more insight into the design choices of BitSimulator.

4.1 Core design

Like many network simulators handling applications and routing protocols, BitSimulator is event-driven. At its core lies a **discrete event model** where actions are scheduled into a time ordered list. Events in this queue are processed in order, and may in turn trigger the insertion of new, subsequent, events. The simulation ends when there is no more event in the list or a predetermined simulated time is reached.

The simulator has to cope with both very short durations (such as the duration of a pulse or the radio propagation delay over a few millimeters) and relatively longer durations (such as the time between packets generated by applications). This is done by internally storing the time as a 64 bits integer number of femtoseconds. This very high time resolution allows to process both 100 fs long pulses and long simulated times. Nodes exist in a simulated 2D or 3D world and may change their position and orientation over time. To be consistent with the very high time resolution, resolution of the positions has to be equally high, so position and distances are internally expressed in nanometers.

Reproducibility of the results is ensured by the use of several seeded RNG. This allows for example two simulations which use the same random placement of nodes, but different random parameters of the simulated protocols.

Scenarios are defined in XML files, but many parameters can be defined or overridden by command line options. This allows to start simulations in an executable script file.

The scalability in the number of simulated nodes is a main design goal. It is attained by a careful balance between versatility and simplicity of the code. A relatively small number of C++ classes are provided and encapsulate all main features, as explained below.

4.2 Network features

To keep the simulator simple and fast while allowing researcher to control application and network protocols, an infrastructure with three main networks layers is provided.

Physical and channel access control layer. It deals with radio propagation and computation of reception errors. Simulated devices are equipped with a unique nanowireless transceiver, whose range and orientation are configurable. This layer implements by default the TS-OOK model with 100 fs pulses and a per frame configurable β parameter. It can also be easily altered to implement any other pulse based model.

Because multiple frames can be temporally multiplexed over the channel, nodes have to track the one (or possibly the ones) they are interested in. Hardware or software on devices often limit the number of frames that can be tracked simultaneously. This value is configurable in the simulation through the `maxCurrentReceptions` parameter.

This layer is primarily implemented in the `Node` C++ class (see Figure 5), with interactions and supporting placement data structures implemented into the `World` class.

Routing. Due to the very limited available energy, communication range of nanodevices is expected to be very short. Multi-hop, ad

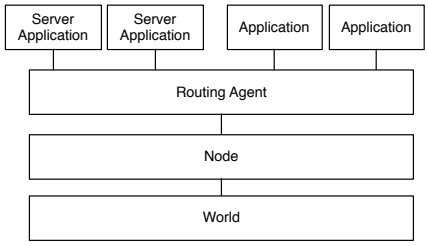


Figure 5: Interactions among main C++ classes.

hoc style, networks are expected to be common in nanonetworks. To this end, the routing layer implements three options: no routing, flooding and SLR [8]. Custom ones can easily been added.

Application. BitSimulator allows two types of applications: `Application`, which can only send packets, and `ServerApplications`, which can send and receive packets. A `Node` can host any number of both. `Server` applications bound to a logical port that allows demultiplexing and distribution of incoming packets to the correct `ServerApplication`. One just has to derive those classes into his own, and attach their instances to the nodes.

Data flowing through the network are modeled by the `Packet` class. Packets contain a binary payload (that can be defined by an application, defined statically or defined randomly), along with various metadata, which help to visualize and understand protocols involved. They include source, destination, packet and flow identifiers, along with a few others. They can be easily extended to match further requirements.

Upon correct reception, `Packets` are handed to `ServerApplication` instances running on `Nodes`. It is possible to set the maximum number of erroneous bits for which the packet is still considered correct. Packets, even damaged, can be passed to the upper layer, allowing to implement a coding or redundancy scheme.

5 ANALYSIS AND VISUALIZATION TOOLS

VisualTracer is an efficient tool to display trace files generated by the simulator. It currently supports two display modes: global and individual. The global one shows a map and allows interactive highlighting of actives nodes (transmitting, receiving, experiencing a collision, ...). The user can choose a time interval to display and can interactively navigate over the time axis.

On Figure 6, a scenario with 30 000 nodes (all in the same communication range) and 4 senders was used. All senders started their transmissions almost at the same time. Figure 6 shows screenshots of the map part of the interface at 4 different times. We can see nodes finishing to receive a packet from image to image. Even if not really visible on these particular pictures because of display parameters, some nodes experience collisions, following the model presented earlier.

The full interface of VisualTracer in global mode is shown on figure 7. The same scenario as in Figure 6 is represented, but the time interval to display is much larger in Subfigure 7a. In this subfigure, nodes getting corrupted packets are easily identified (the dotted lines). On the right part of the screenshot, graphs show the activity over the selected time interval. Here we can see the 4 packets being sent and the neighbors gradually receiving them depending on their

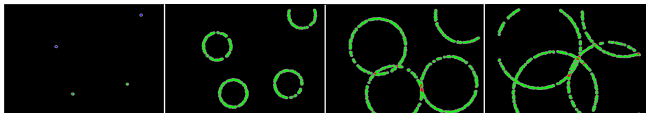
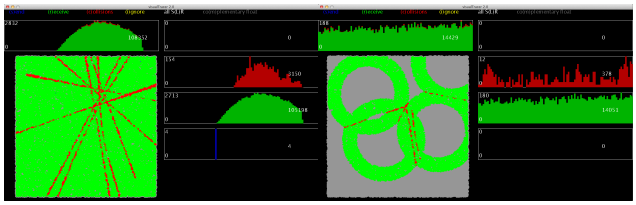


Figure 6: 30 000 neighbors receiving a few packets over time (4 very small time steps).



(a) Large time interval. **(b) Medium time interval.**

Figure 7: Propagation delay causing deferred reception as seen in VisualTracer.

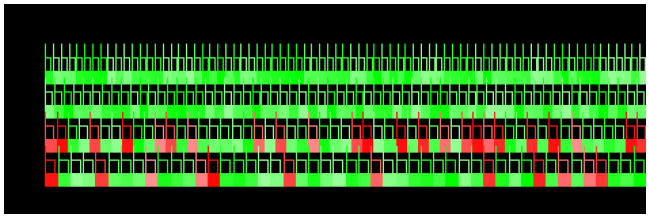


Figure 8: Multiple collisions between 2 flows in a 4 flows scenario.

distance to the sources. Subfigure 7b represents the same scenario again, but with a smaller time interval, helping to better understand where and why collisions occur.

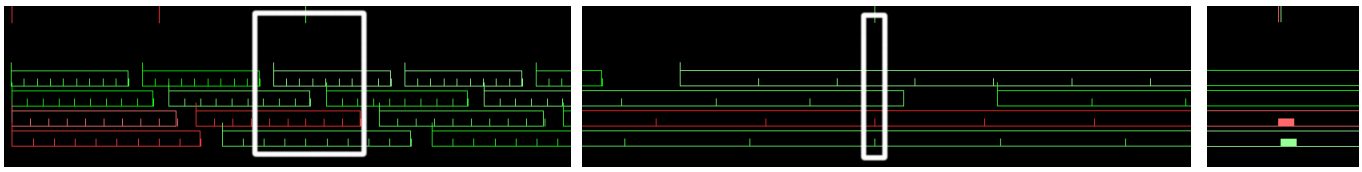
The second visualisation mode is individual, it represents the point of view of a given node as a chronogram. It displays bits and packets over a timeline and is especially useful to detect repetitions and bursts in the collisions. Figure 8 shows collisions occurring in a 4 flows scenario. In this case only 2 of the 4 flows are affected by collisions. Figure 9 gives an example of various (interactive) zoom levels in another scenario with 4 active transmitters sending 10 bits packets and a few collisions occurring.

6 CASE STUDIES

A simulation tool is meant to be used to search for new protocols and applications. In this section we briefly present some of the simulations we have already conducted and we highlight the benefits from using BitSimulator.

6.1 Collision accuracy

We developed a scenario with 5000 nodes in communication range. Among them, a few tens independently broadcast constant bit rate flows of 8000 bits packets. Packet generation rate is chosen so that there is just enough time to completely send a packet with $\beta = 1000$ before generating the next packet. If a smaller value of β is used,



(a) Concurrent 10 bits packets. (b) Zooming on colliding packets. (c) Bits coll.

Figure 9: Chronogram mode in VisualTracer.

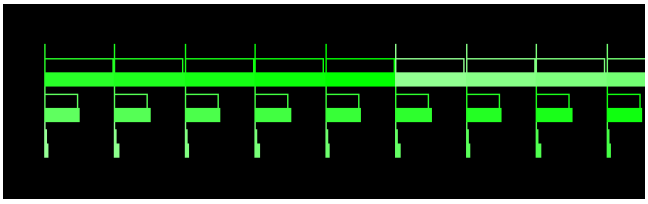


Figure 10: Time spacing of constant bit rate packets for $\beta = 1000$ (up), $\beta = 500$ (middle) and $\beta = 50$ (bottom).

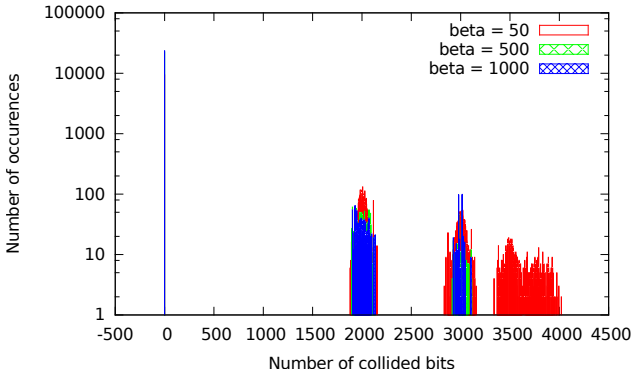


Figure 11: The number of collided bits is either around 2000, when two packets collide, or around 3000, when three packets collide, or around 3750, when four packets collide.

this means a packet would just take less time to be sent, and that the node has to wait for a new packet to be generated. However, all nodes are configured to use the same β value (50, 500 or 1000 respectively in the three runs described below). Figure 10 shows graphically how those flows occupy the channel over time.

To prevent collisions occuring because all transmitters start at the same time, an additional random backoff is used, with a maximum duration in the order of T_s (i.e. the duration of a pulse multiplied by β). The payload of those packets is randomly chosen and all are unique.

This scenario can be considered an all-or-nothing for packets collisions. Between two concurrently received packets, bits are either all aligned (because of the respective backoffs of those packets and their propagation time), or none of them. But even if all bits of a packet are aligned (i.e. received at the same time) with those of another flow, it does not mean that all of them will be altered (cf. Figure 4). In fact, if two 8000 bits packets collide, only "0" bits received are overridden by "1" bits from the other. For random payload, this occurs about 1/4 of the time. This is exactly what is shown on Figure 11, which counts packets having a given number of altered bits. Most packets observed in the scenario have around 2000 erroneous bits over their 8000 total.

However, with smaller values of β, additional groups of packets tend to be more common, having around 3000 and 3750 errors (and higher spread). This is explained by a higher contention and collision not with other packets, but 2 or even 3 others at the same time. The observed values can be analytically computed, but we do not present them here because of space constraints.

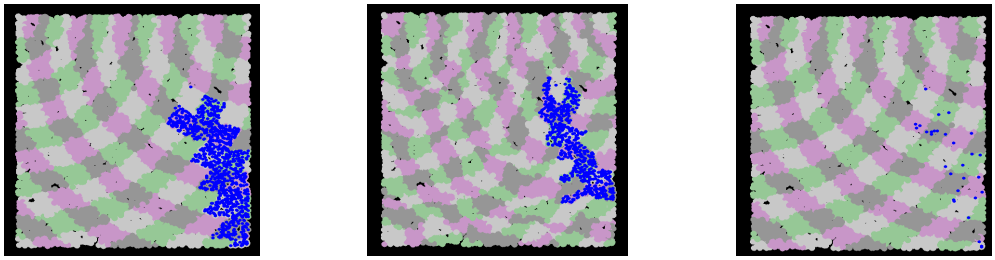
6.2 Multi-hop broadcasting and reachability

Given that nodes have a small individual communication range, multi-hop nanonetworks are expected to be common. Information and control packets will have to be propagated through parts or even to the whole network. Many strategies exist already, and more will be proposed; but all have to be evaluated in the context of wireless nanonetworks.

BitSimulator allows to analyze such protocols, even in the case of dense or large networks (hundreds of thousands of nodes, thousands or more direct neighbors for each node). Moreover, through the interactivity of VisualTracer, it becomes easy to pinpoint behaviors and threshold values below which an evaluated protocol starts to crumble. Figure 13b shows the nodes reached by a multi-hop propagation. Parts of the network obviously were not reached (black holes, especially top-right area). Such as map (that can be interactively animated in VisualTracer) is a useful complement to traditional metrics. For example reachability percentage or copies received (shown on Figure 13a) do not capture distribution over time and space.

6.3 Routing and optimization

Broadcasting and propagating information over the whole multi-hop network is not always desirable. Routing protocols are consequently required. BitSimulator implements the SLR routing protocol [8]. In SLR, nodes have a position expressed in number of hops to a few chosen and immobile nodes called *anchors*. In the original SLR protocol, positions of all nodes are computed by flooding beacons from the anchors. Multiple nodes can share the same coordinates and form *zones*. Routing of data packets then works with nodes determining if they are on the path of an incoming unicast packet, and if so, simply forwarding it. Figure 12a gives an

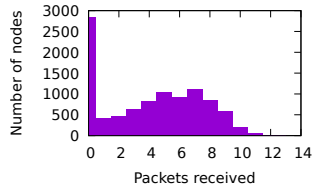


(a) Default SLR zones and an example of unicast route.

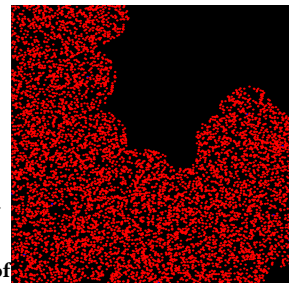
(b) SLR zones when few nodes participate in the initial beacon flooding.

(c) Optimizing the number of forwarders per zone.

Figure 12: Simulation with the SLR routing protocol.



(a) Distribution of the number of copies received.



(b) Incomplete propagation area.

Figure 13: Reachability in a multi-hop network.

example of zones produced by SLR, and shows all the nodes that consider themselves as forwarders along a unicast route.

As the initial flooding used to propagate the beacons can be very costly, strategies to reduce its impact can be of interest. Figure 12b shows the impact of such a strategy. In this case, we started by running DeDen, a neighboring cardinality estimation protocol. By knowing the local density of the network, we can then strongly limit the number of nodes participating in the propagation of the beacons, while preserving the connectivity of the network.. Retransmitting nodes are however chosen randomly, and, as seen on the figure, this has a visible impact on the regularity and shape of the zones.

In the original SLR protocol, all nodes of a zone that consider themselves on the path of a data packet will forward it. This can be costly in dense environments. Figure 12c shows the effect of a strategy that strongly reduces the number of retransmits per zone. To this end, nodes use an independently computed estimation of their number of neighbors.

It should be noted that an easy to use infrastructure is provided to implement new routing protocols. One only has to derive the `RoutingAgent` class and attach it to the `Nodes`.

7 CONCLUSION

This paper presents BitSimulator, an open source network simulator targeting electromagnetic nanonetworks. It takes into account the specificites of the channel and channel access, especially by allowing many parallel communications (packets being sent simultaneously). It allows low level studies by effectively computing collisions at the bit level. As such, it enables work on the coding itself, and is also very useful for designing new channel access or channel sharing schemes. It is highly optimized for speed and supports numerous nodes (hundreds of thousands nodes, each having tens of thousands neighbors have been tested). BitSimulator is accompanied by VisualTracer, an interactive visualization and analysis tool that greatly helps in dense and complex scenarios.

Future work includes the enhancement of the physical model by taking into account the noise generated by an excited medium, along with more computation speed improvements.

ACKNOWLEDGMENTS

This work has been funded by Pays de Montbéliard Agglomération.

REFERENCES

[1] Nicolas Boillot, Dominique Dhoutaut, and Julien Bourgeois. 2015. Going for large scale with nano-wireless simulations. In *2nd ACM International Conference on Nanoscale Computing and Communication (NanoCom)*. ACM, Boston, MA, USA, 1–2.

[2] Eugen Dedu, Julien Bourgeois, and Muhammad Agus Zainuddin. 2014. A first study on video transmission over a nanowireless network. In *ACM International Conference on Nanoscale Computing and Communication (1)*. ACM, Atlanta, Georgia, USA, 1–6.

[3] J.M. Jornet and I.F. Akyildiz. 2011. Low-Weight Channel Coding for Interference Mitigation in Electromagnetic Nanonetworks in the Terahertz Band. In *Communications (ICC), 2011 IEEE International Conference on*. IEE, Kyoto, Japan, 1–6.

[4] Josep Miquel Jornet and Ian F Akyildiz. 2011. 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)*. IEEE, Salt Lake City, UT, USA, 80–88.

[5] Josep Miquel Jornet and Ian F. Akyildiz. 2014. Femtosecond-Long Pulse-Based Modulation for Terahertz Band Communication in Nanonetworks. *IEEE Transactions on Communications* 62, 5 (May 2014), 1742–1753.

[6] Giuseppe Piro, Luigi Alfredo Grieco, Gennaro Boggia, and Pietro Camarda. 2013. Nano-Sim: simulating electromagnetic-based nanonetworks in the network simulator 3. In *6th International ICST Conference on Simulation Tools and Techniques (SimuTools)*. ACM, Cannes, France, 203–210.

[7] Prateek K. Singh, Gregory Aizin, Ngwe Thawdar, Michael Medley, and Miquel Jornet. 2016. Graphene-based Plasmonic Phase Modulator for Graphene-based Plasmonic Phase Modulator for Terahertz-band Communication. In *Proc. 10th European Conference on Antennas and Propagation (EuCAP)*. IEEE, Davos, Switzerland, 1–6.

[8] Ageliki Tsioliaridou, Christos Liaskos, Eugen Dedu, and Sotiris Ioannidis. 2017. Packet routing in 3D nanonetworks: A lightweight, linear-path scheme. *Nano Communication Networks* 12 (June 2017), 63–71.

[9] A. Tsioliaridou, C. Liaskos, S. Ioannidis, and A. Pitsillides. 2016. Lightweight, self-tuning data dissemination for dense nanonetworks. *Nano Communication Networks (Special Issue on EM Nanonetworks)* 8 (2016), 2–15.

[10] Hang Yu, Bryan Ng, and Winston K.G. Seah. 2017. Pulse Arrival Scheduling for Nanonetworks Under Limited IoT Access Bandwidth. In *42nd IEEE Conference on Local Computer Networks (LCN)*. IEEE, Singapore, Singapore, 18–26.