Large scale MEMS robots cooperative map building based on realistic simulation of nano-wireless communications

Nicolas Boillot, Dominique Dhoutaut, Julien Bourgeois

Large scale MEMS robots cooperative map building based on realistic simulation of nano-wireless communications

Nicolas BOILLOT, Dominique DHOUTAUT and Julien BOURGEOIS[1,1,1]

Institut FEMTO-ST (UMR 6174) / Université de Franche-Comté (UFC) / Centre National de Recherche Scientifique (CNRS)
1 Cours Leprince-Ringuet - 25200 Montbéliard, FRANCE
Email : {nicolas.boillot, julien.bourgeois, dominique.dhoutaut}@femto-st.fr

Abstract

The Claytronics project has produced interesting hardware components like cylindric micro-robots called catoms and software models to enable the concept of programmable matter. One application is the use of several catoms linked together so that they can "walk". These walkers can explore an area and thanks to electromagnetic wireless nano-networks, they can communicate with each other sharing the map of the place to explore. In this paper, we study the different parameters influencing the transmission quality of the map to a sink which uses both traditional wireless and wireless nano-communication networks.

Keywords: Micro-robot, Claytronics, Nano-wireless, Map building, Nano-wireless simulation, Vouivre

1. Introduction

Modular robots already exist in various size and shapes, but miniaturization and Micro-Electro-Mechanical Systems (MEMS) technologies allow for even smaller robots and even larger collaborating ensembles. Communications are central to the behavior of such ensembles, both communications between the micro devices and the macro world, but also those between the micro robots themselves.

Within the Claytronics project [1, 2] a new type of modular MEMS robots has been designed for realizing programmable matter [3]. Each micro-robot is a cylinder called a catom (standing for Claytronics atom) that can stick to and move around its neighbors as seen on Figure 1. An individual unit has very few functionalities but an ensemble of catoms is able to act collectively.

Current catoms are limited to direct (by contact) communication using their surface features. This makes sense, as an isolated catom would not be able to do much anyway (it cannot even move without help from others) and fit very well the programmable matter paradigm. But, expanding communication range through wireless links can significantly broaden the scope of possible applications. In previous works, we investigated such possibilities, first adding standard CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) radio to the Claytronics simulation environment [4] and then investigating more specific communications means, through nano-wireless (nano-antennas, terahertz band and pulse based modulation fitting this environment), as seen in [5, 6]. Those previous works built the necessary foundations for more complex applications.

Section 2 presents the context and is subdivided as follow : Section 2.1 recalls more precisely what Claytronics is and how it is simulated. Section 2.2 introduces both the physical environment simulation software - DPRSim - and the wired and wireless communication infrastructure - Vouivre. Section 2.3 first shows and discuss as the specificities of nano-wireless communications and then presents in detail an implementation into DPRSim / Vouivre.

In section 3, a scalable application will be presented, where catoms scattered in small semi-autonomous groups, collaboratively explore an unknown physical environment and inform a "macro" user through a data sink. This allows a fast and detailed exploration of an unknown environment as each group of moving catoms is able to share the map of the environment while transmitting it to the sink. This kind of application could be used in many areas. In structural health monitoring (SHM), walkers would be able to detect very small damages in structures by comparing the difference between the original and the current state of the structure. In a human body, they would allow detailed monitoring of the shape a organ, for example, in the case of a cancer that would need 24/7 monitoring, it would help to understand the dynamic of the remission or development of the cancer. Multiple aspects will be considered, especially the necessity to use nano-forwarding to cope with the limited range of the nano-transceivers, and data aggregation, to cope with the considerable amount of data that can be exchanged when large numbers of robots are used.

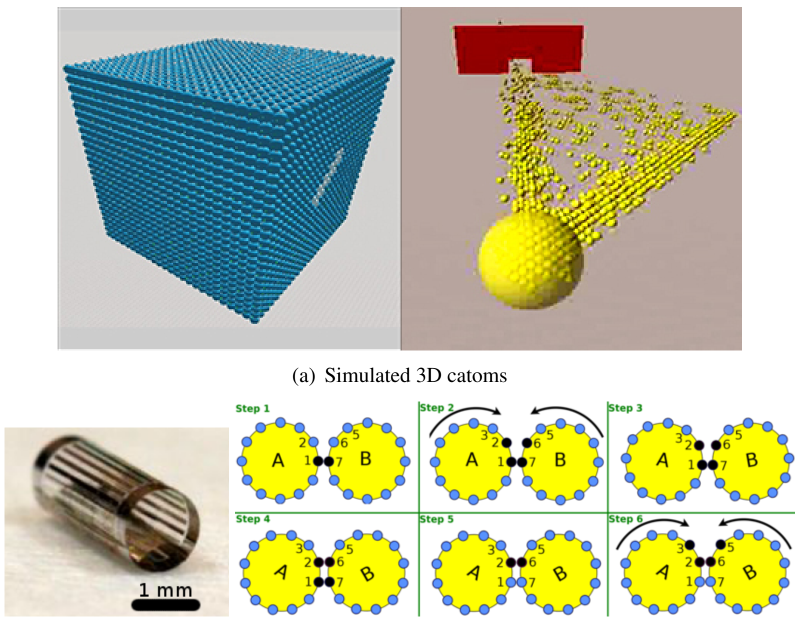
Finally, detailed simulations will be conducted and discussed in section 4 where the code running on each node will be executed individually. The network simulation will benefits from our previous works, where packet loses will depend on the specificities of the nano-wireless channel along with the number of concurrent transmitters.

2. Context, nano-wireless and simulation environnements

2.1. The Claytronics project environment

The Claytronics project has originally been created by Intel and Carnegie Mellon University, Université de Franche-Comté (UFC) / FEMTO-ST Institute joined the project later on. MEMS micro-robots called catoms (see simulated 3D ones in Figure 1.a and 2D prototype on Figure 1.b) can move around or assemble to their neighbors and collaboratively build basically any required shape. This is why they can truly be called “programmable matter”. They are covered with structures called “features”. Those features are used as mean of both dynamic attachment (by electromagnetic or electrostatic force [7]) and direct communication.

An isolated catom cannot move by itself, but, it can by coordinating the activation of its feature, it will turn around its neighbors(s). Figure 1.c displays 6 steps of a rolling motion of two catoms (the black dots shows currently activated, i.e., attracting, features). More complex movements are possible, but usually involve more catoms serving as anchors and preventing counter-reactions and movement of both catoms. Please note that depending on the distribution of the features on their surface, catoms may not form a regular lattice or be aligned. Even with an appropriate shape of the catoms, regularity is not easy to maintain. For the sake of simplicity, motion control algorithms would tend to seek stable and aligned formations, but this can prove difficult in the real world, because of the friction and the fluctuations in the precision of the movements. This phenomenon should not be neglected when later doing simulations ! In the rest of this paper we will considerate that binding 2 features of different catoms together is an atomic and orthogonal move. As basic as it may seems, moving a catom is already a complex problem, requiring multiple communications and coordination with one or more other catoms.



(a) Simulated 3D catoms

(b) Prototype

(c) Actuating features to move catoms

FIGURE 1: Claytronics catoms

Current catoms do not possess wireless transmission capabilities. The aim of this paper is to investigate the benefits of bringing nano-wireless to them through an application collaboratively exploring the physical environment. Because catoms only exist at the state of a few prototypes, and because nano-wireless itself is only in its initial stages of development, simulation is mandatory to conduct this investigation work.

2.2. *Simulation environment*

This work makes use of DPRSim as initially developed for the Claytronics project, complemented by our own Vouivre network simulation library used for the communications.

Dynamic Physical Rendering Simulator (DPRSim) [8, 9] has been developed by Intel since 2006 for the Claytronics project. This simulator is designed to support a potentially very large number (up to several millions) of Claytronics micro-robots. DPRSim is able to simulate physics via ODE (Open Dynamics Engine, a rigid body dynamic library) and to provide a 3D visualization environment via Drawstuff and OpenGL. In DPRSim, each catom is individually represented by an object instance, and the code running on a catom is also individually instantiated. To simulate codes running on catoms and physics, DPRSim uses a time-slicing operating mode, with an atomic duration step (called a "tick"), and processes the whole simulation step by step.

A tick is not a divisible duration and this has many repercussions (synchronism barrier, low simulation efficiency, etc.), especially on the messaging system.

We developed **Vouivre**, which has been first integrated in DPRSim but that can work now as a standalone network simulator. It allows an adaptable trade-off between complexity and realism of the network, and is able to simulate efficiently several simultaneous and heterogeneous timescales. It is based on discrete events simulation and can to handle the radio channel and its concurrent accesses.

In [4], we detailed these points and also presented an extended version of DPRSim interfaced with Vouivre, adding wireless communication capability to the simulated catoms. The radio wireless model exposed was based on a CSMA/CA implementation using a Friis propagation model with 2.4GHz centered frequency band.

In [5, 6] we then implemented nano-communications as a pulse-based mechanism and use a dedicated modeling of the environment to take into account the specificities of both nano-transceivers and terahertz frequency band. Evaluations were conducted on basic test scenarios to get a better grasp of the behavior of nano-communications.

In the current paper, we keep the nano-wireless radio model but we consider a more advanced scenario where wireless nano-communications between many collaborating micro-devices leads to innovative and efficient applications at the macro level.

2.3. *The nano-wireless model and its implementation*

2.3.1. *Specificities of electromagnetic nano-networks*

Nano-wireless communications make use of nano-meter scale antennas, and many aspects of communications are consequently affected. At that scale, classical modulation schemes are difficult to implement and pulse-based communications are preferred. Time-Spread On-Off Keying (TS-OOK) [10, 11] uses a simple coding where the presence of a pulse carries a "1" value and its absence carries a "0". Extremely short pulses (~100 femtoseconds long) spread their small energy - in the order of a picojoule - into an extremely wide channel, up to 10 THz large.

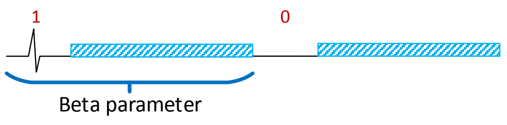


FIGURE 2: The β parameter in the TS-OOK model

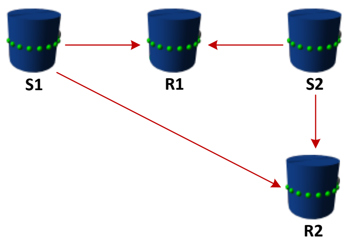
The total throughput is thus potentially extremely high, in the order of terabits/s. For a given communication, each symbol is separated from the next by an interval orders of magnitude longer than a pulse. If T_p is the duration of a pulse and T_s is the time between symbols, β (see fig 2) is the ratio T_s/T_p and a value of 1000 would not be uncommon. Because of this ratio, the achievable throughput for a given transmission is also orders of magnitude lower than the total theoretical throughput.

But a very interesting property of such a coding scheme is the ability to multiplex transmissions over time. Transmitters can indeed send their symbols during the inter-symbol time of the other transmitters. Collisions may occur, but their probability is directly dependent to the value of β and the number of concurrent transmissions. The multiplexing capability increases with the value of β.
Of course, higher level coding schemes are required to detect and eventually correct errors.

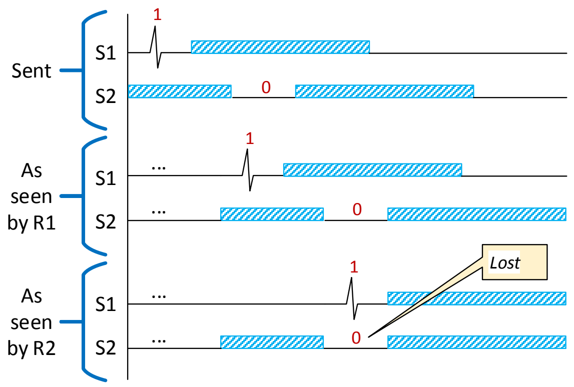
Such a wide channel behave peculiarly compared to more common wireless technologies. In particular various molecules in the environment will absorb parts of the spectrum while keeping others relatively untouched [12]. The energy absorbed by the environment is radiated again and essentially acts as noise on the channel. This results into equivocation. The noise on the channel rapidly increases with distance and transmission power, and rapidly surpass the signal power at the receiver.

This is happening even when no parallel transmission are occurring and limits the effective communication range. However communications ranging up to tens of centimeters seems possible

Because the pulses are very short (~100 femtoseconds), the propagation delay (~3 nanoseconds per meter) can be greater than the duration of a pulse. Then, collisions can easily occur at receiver side even if symbols were sent at different times. Figure 3 shows how symbols can overlap from a receiver point of view even if they were not sent at the same time. Sender S1 sends a “1” (as a pulse) and S2 later sends a “0” as an absence of pulse. Because receiver R1 is at the same distance from S1 and S2, it received those symbols one after the other and decodes them correctly. Receiver R2 is much closer from S2 than from S1. As the propagation delay from S1 is longer, the two symbols overlap and the “0” from S2 is masked by the “1” from S1. Also, if the nodes are able to move, the propagation delay will not remain constant over time for a given communication link.



(a) Spatial positions



(b) Chronogram

Figure 3: “0” symbols can be masked by “1” even if not sent at the same time

However, it is interesting to note that only in certain circumstances a symbol may be corrupted by a collision, thus alleviating a little this problem. Only when receiving a “0” can a node be affected by a collision. If a node is currently receiving a “1”, whatever it receives at that time from others nodes, it will still be interpreted as a “1”. When receiving a “0”, receiving other “0” will not cause a problem either. Only concurrents “1” will mask a “0” currently being received, as it would be the case for receiver R2 for the symbol sent by S2 on Figure 3.

2.3.2. *Discrete-event simulation of communications with propagation delay*

Vouivre [4] is a C++ network simulation library we developed as both an extension for DPRSim and as a standalone simulator. Whereas DPRSim makes use of a time-slicing approach (time is advancing step by step, and each duration of a step is constant), Vouivre is based on the discrete-event paradigm more commonly encountered in other network simulators like NS3 [13], OMNeT++ [14] or OPNET [15] to cite a few. Discrete-event simulators do not offer physics and real-world modeling like we are doing in conjunction with DPRSim : for example, mobile network nodes are only 2D objects with no support for collision.

One of the major problem when simulating nano-networks is to provide fast simulation. As we have seen a pulse duration in expressed in femtoseconds, so simulating few seconds of communications will require a huge number of events.

The timescale used to represent the events occurring in nano-wireless communications has to be extremely small. Nano-networks work in multiple THz wide bands and the duration of a TS-OOK pulse is below one picosecond.

This is not a problem for a discrete-event simulator such as Vouivre. But, the application code processing the received messages, is executing in DPRSim and is only called at each DPRSim tick. This was causing an usually long latency between the reception of a message and its processing. To prevent this it was necessary to reduce the duration of a tick. As a side-effect, as DPRSim also call ODE (its physic simulation engine) between each tick to update position and velocity, the simulation performances could have degraded a lot. To be able to correctly simulate TS-OOK nano-communications and still have reasonably fast simulation, we decided to use a 100 picoseconds tick duration as a trade-off. This duration adds a delay to the processing of each message but in most scenarios this can be considered as harmless.

We implemented new interfaces using TS-OOK over the same discrete-event model we used in [5]. The channel model can now be changed on the fly. In the following work, only the TS-OOK interface were used with logical “1” being the first derivative of a 100 femtosecond long Gaussian pulse and a duration between symbols of 5 picoseconds.

At the envisioned physical scale (nano-devices scattered over areas ranging from square centimeters to square meters) and because of the extremely small duration of the pulses, the signal propagation delay has to be taken into account. The propagation speed is around 3 nanoseconds per meter and the propagation delay can therefore be greater than the duration of a pulse which can cause collisions.

Moreover, catoms are mobile objects and the distances can vary over time.

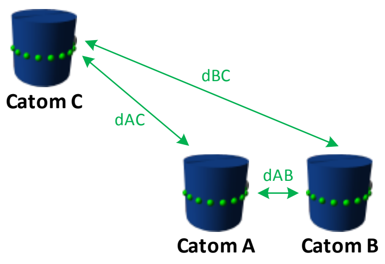


Figure 4: Propagation delay of electromagnetic signal depends on the distance between catoms.

Figure 5 shows events as they are generated when catoms represented on Figure 4 communicate. Catoms A and B transmit information and points of view of A, B and C are represented on Figure 5. TS-OOK uses temporal multiplexing of the radio channel such that multiple messages can be sent at the same time. As pulses are comparatively very short compared to the time between them, the probability of a collision is small. Moreover, as seen previously in Figure 3, only a “1” can mask a “0”. The “1” are said to be dominant and the “0” recessive. If the symbol being received for the current packet is a “1”, other concurrent “1” or “0” will not directly affect it. Only when receiving a “0”, if a “1” is transmitted at the same there will be a data corruption.

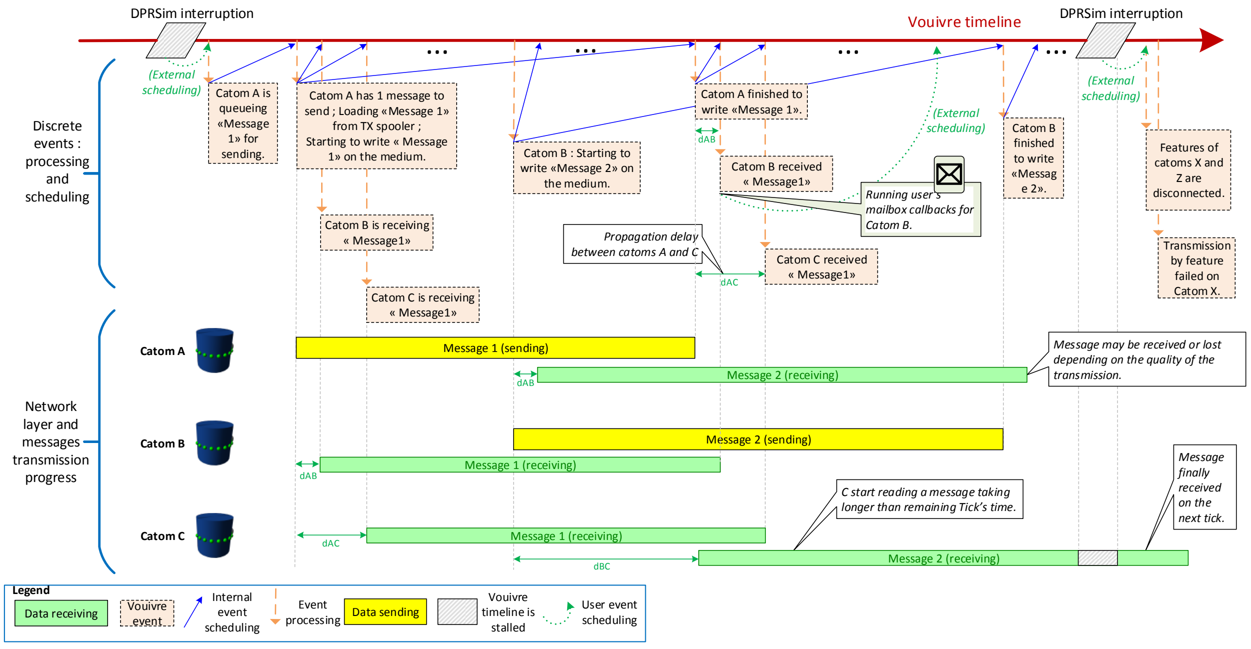

DPRSim interruption
Vouivre timeline
DPRSim interruption
(External scheduling)
Discrete events : processing and scheduling
Catom A is queueing «Message 1» for sending.
Catom A has 1 message to send ; Loading «Message 1» from TX spooler ; Starting to write « Message 1» on the medium.
Catom B is receiving « Message1»
Catom C is receiving « Message1»
Catom B : Starting to write «Message 2» on the medium.
Propagation delay between catoms A and C
Catom A finished to write «Message 1».
dAB
Catom B received « Message1»
dAC
Catom C received « Message1»
(External scheduling)
Running user's mailbox callbacks for Catom B.
Catom B finished to write «Message 2».
(External scheduling)
Features of catoms X and Z are disconnected.
Transmission by feature failed on Catom X.
Network layer and messages transmission progress
Catom A
Message 1 (sending)
dAB
Message 2 (receiving)
Message may be received or lost depending on the quality of the transmission.
Catom B
Message 2 (sending)
dAB
Message 1 (receiving)
C start reading a message taking longer than remaining Tick's time.
Message finally received on the next tick.
Catom C
dAC
Message 1 (receiving)
dBC
Message 2 (receiving)
Legend
Data receiving
Vouivre event
Internal event scheduling
Event processing
Data sending
Vouivre timeline is stalled
User event scheduling

Figure 5: Discrete events scheduling and consequences on the network layer

2.3.3. *Simulation from receivers point of view to determine incoming message acceptance*

For each receiver, a table is maintained with all incoming messages. Each receiver gets a unique table as the received packets are affected by the propagation delay previously mentioned.

As previously explained, the radio channel used for nano-wireless is very peculiar. In particular, a transmitted signal is not only impacted by a continuous attenuation with the increasing distance, but also by the noise caused on the channel by itself and other concurrent transmissions. We have also seen that, especially when the number of concurrent transmissions is high (from thousands to hundred of thousands), reducing the weight of the coding brings better performances. Reducing the weight of the coding means increasing the proportion of "0" symbol (no pulse) against the '1' (pulse). Even if this way of coding increases the size of the message, its ability to correctly receive the message exceeds its drawbacks.

DPRSim is meant to simulate networks of sizes ranging from a few catoms to hundred of thousands of them. Determining if a packet has been received correctly or nor has to be done in a practical way. We need to be able to scale to large numbers, but still capturing the peculiar behavior of the nano-wireless radio channel.

In the programmable matter and Claytronics specific contexts, because of the application and control layers, there will be large variations in the communication needs over time. Sometimes almost no communications are required as the network reached a very stable state. Sometimes only a few catoms may want to transmit, and sometimes events or user commands may trigger large and immediate urge for communications. These requirements are for example different from those of nano-sensors networks, where the global network load is much more constant over time.

To avoid computing the propagation of each pulse we only compute changes in the number of concurrent transmissions affecting the reception of an incoming packet. Those changes define one to multiple periods for this incoming packet which are affected by a coefficient we get from a precomputed table.

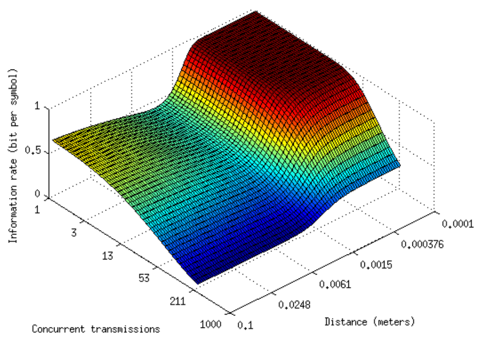


FIGURE 6: Single-user achievable information rate in bit/symbol as a function of the number of nano-devices and the transmission distance. Matlab model by Jornet and Akyildiz

The precomputed table has been generated from the statistical model of the network capacity done by Jornet and Akyildiz. This model takes into account the number of concurrent transmissions along with the distance from the sender and the generated dataset is represented on Figure 6. The path-loss and noise in the terahertz band are computed by using models introduced in [12, 11]. A standard medium with 10% of water vapor is considered. This model takes into account the number of concurrent transmissions along with the distance from the sender and the generated dataset is represented on Figure 6. As the number of catoms in the following experiments is kept relatively small (up to a few hundreds), we did not use a variable weight coding. Instead, we used an equal probability of "0" and "1" as it is mostly optimal for low numbers of concurrent transmission. An higher (thousands and above) numbers would have required an adaptive coding, with catoms sensing the activity (or being informed by some authority or centralized system) and choosing a more appropriate weight.

To speedup the simulation, we only compute changes in the number of concurrent transmissions affecting the reception of an incoming packet. Each change is named a pivot. Each couple of pivots determines a periods in which each incoming packet is affected by a coefficient. This coefficient is retrieved from a precomputed table generated from the statistical model of the network capacity done by Jornet and Akyildiz. This model takes into account the number of concurrent transmissions along with the distance from the sender. The path-loss and noise in the terahertz band are computed by using models introduced in [12, 11]. A standard medium with 10% of water vapor is considered. This model takes into account the number of concurrent transmissions along with the distance from the sender. We used an equal probability of "0" and "1" but we will see in the experiment that for high (thousands and above) numbers of communicating nodes a low-weight adaptive coding would enhance the network capacity.

Figure 7 shows the point of view - affected by the propagation delay - of a node simultaneously receiving three messages. The impact of concurrent transmissions on the various parts of the messages has to be computed as the average number of bits correctly received. During the first period (period A), only "message 1" is being received and got a factor of 1 from the channel capacity table. This table takes into account the distance and the equivocation effects of the message itself on the channel. Getting a factor of 1 is possible, but even alone on the channel, depending on the distance it could be less. The second period (period B) has two concurrent transmissions, which get different

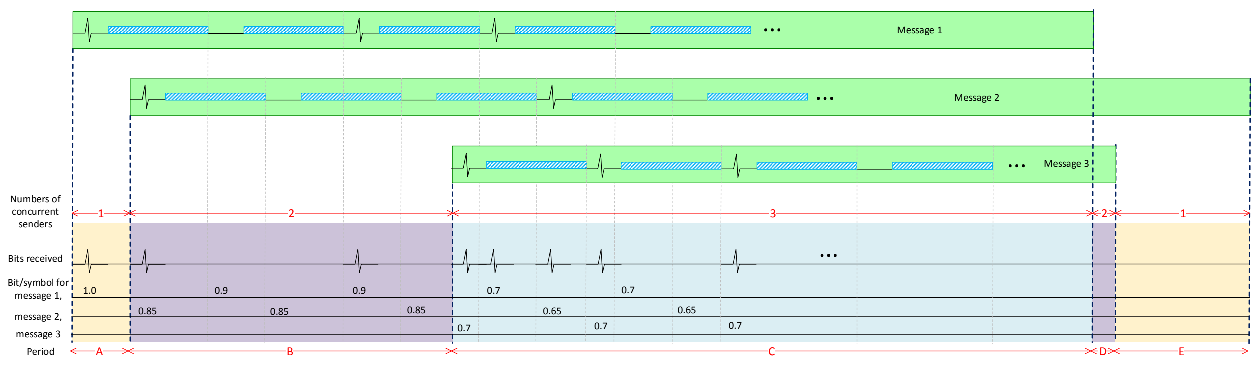


Figure 7: Calculating the number of bits received per message by using the information rate.

coefficients because the distance to the transmitter is not the same. The third period (period C) sees more transmitters and consequently even smaller factors. When the reception of a packet ends, a global factor for this packet is computed proportionally from the factors of its individual periods.

For each period (computed from a couple of consecutive pivots), the information rate in bit per symbol is extracted from the statistical MATLAB model of the network capacity done by Jornet and Akyildiz (cf. [11]). This bit information rate is then used to compute the SCORE of the message receiving which is the average of the information rate weighted by period lengths relatively to the whole message length.

2.3.4. *Simulating redundancy*

In order to reduce the number of errors, we use a MVT (Majority Vote Takers) redundancy algorithm which increases threefold the size of the messages. When the reception of a packet ends, a correction factor computed is applied. If a tolerance threshold is met, the message is dispatch to the mailbox of the catom. If this threshold is not reached, the packet is lost for this receiver. But as each catom will conduct this procedure within its own environment, the message could still be received elsewhere.

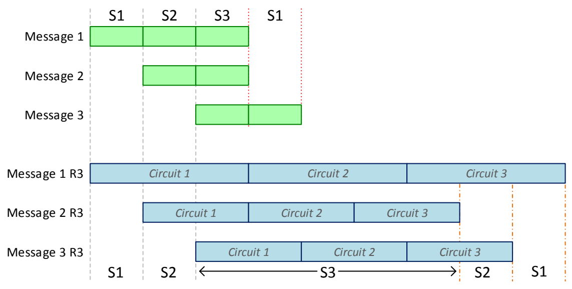


Figure 8: Modifying the scoring system to support N-modular redundancy : Redundancy circuits are shifting pivots which require a new scoring calculation

Using TMR (Triple Modular Redundancy, cf. [16]) in this context makes the assumption that during a communication the network condition will not vary dramatically. This assumption ensures the equiprobability of losing a bit during the three redundant transmissions of the message. Having different network conditions during one message transmission will be done in a future work.

Depending on the redundancy factor, pivots used to calculate scores for incoming messages are modified. As a consequence, the different messages are overlapping in a different way modifying the score calculation.

Furthermore, depending on the redundancy algorithm used and on the position of the redundancy circuits, the ability to correct message error can also be affected. Considering these two points, it is not possible to calculate the final score of the messages using the usual formula for N-modular redundancy described in [17]. Indeed, these formula are using an equiprobability for all redundancy circuits which is not the case in our situation.

For each redundancy factor, it is necessary to determine the probability of success of the majority vote from all possible vote combinations. For instance, with a redundancy factor of 3 and considering P_{c1}, P_{c2}, P_{c3} as individual success vote probabilities for the 1st, 2nd and last redundancy circuit, the probability of success of the vote of the majority circuits is given by the following formula :

$$\begin{aligned} P_{r3} = &(1 - P_{c1}) * P_{c2} * P_{c3} \\ &+ P_{c1} * (1 - P_{c2}) * P_{c3} \\ &+ P_{c1} * P_{c2} * (1 - P_{c3}) \\ &+ P_{c1} * P_{c2} * P_{c3} \end{aligned}$$

We have generated the probability of success using the same way for all the modular redundancies we have used.

On such an unfavorable environment, the probability for a symbol to be lost or corrupted is usually great. Coding schemes adding resilience are necessary, but cannot get too complex because of limited processing power on nano-devices. To determine whether a message with corrupted symbols can be corrected or not, a Majority Vote Takers (MVT) algorithm [17] has been implemented.

The higher the redundancy factor is, the bigger the message size on the transmission medium will be (see Figure 8). As a consequence, it is necessary to take into account this phenomenon which shifts pivots for the score calculation of incoming messages. Figure 8 shows the point of view of a receiver (after taking into account the propagation delay) receiving simultaneously 3 messages without redundancy (upper part of the figure). The lower part of the figure highlights the same situation with a redundancy factor of 3.

2.3.5. *Behavior of the nano-wireless throughput*

To study the useful throughput related to the beta parameter, several simulations were conducted on statics nodes in [6]. Figure 9 shows thresholds in function of the number of concurrent senders. It is consequently necessary to dynamically adapt the beta parameter when the number of concurrent sender increases.

As explained in [6], the average number of received bits per message, for a given receiver, is calculated using a SCORE given to each concurrent incoming message (See Figure 7). To avoid threshold effects and to statistically smooth the results for all the simulations, a random message acceptance decision function is used. Thus, the score for an incoming message is processed using a probability. Messages having high scores have better chances to be received whereas messages having low scores are likely to be lost. The information rate in bits per symbol will change as explained in [11] depending on the number of nodes contained in the group of sender catoms. We configured this MATLAB model for an environment compounded of air with 10% of moisture. Three information rate matrices have been exported with different inter-symbol durations represented by the β-parameter with values 500, 1000 and 2500

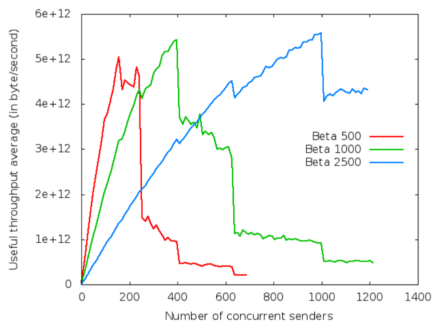


Figure 9: Average aggregated useful throughput after 200ns of simulation

3. Cooperative map building application : objectives and algorithms

To demonstrate the capabilities and usefulness of wireless nano-communications, we present a cooperative map building scenario. In this section we will first introduce the approach, discuss the multi-levels architecture of the application, and finally present the selected control and communication algorithms.

3.1. *A low-level motion : From catoms to walkers*

Coordinating motion of catoms or ensembles of catoms is a complex problem that requires an iterative reduction into smaller and simpler ones. We choose to simplify the main problem into two main levels :

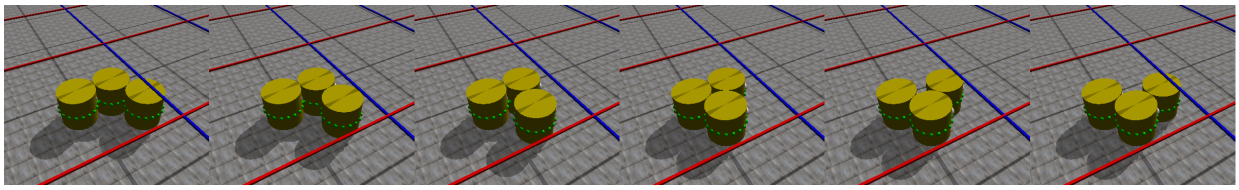


FIGURE 10: Software driver implemented in DPRSim to control successive motions of each catom in a walker.

• Individual catom management : A seen in Section 2.1, an isolated cannot move by itself. A small group (usually at least 3 catoms) is required. Algorithms from this level are responsible of moving a catom by managing the necessary communications and physical interactions with its neighbors.
• Group coordination is required to coordinate a set of catoms. At the very least, the presence of others is required for a catom to move. Catoms can communicate together through theirs features. By multi-hop routing, messages can be relayed and we considered that each catom can discover the group to which it belongs. Each group of contiguous catoms forms a walker. By direct wired communications, a walker can share energy, distributed CPU and memory. As a consequence a big walker can store more map informations than a small walker.

In DPRSim, we implemented a software driver in charge of controlling the position of each catom in a walker. Moving a walker means moving one of it component to the next feature sequence. The walking algorithm can be viewed on Figure 10 where a first catom changes its position (steps 1 to 4) and then another one moves (steps 5 and 8), effectively progressively moving the whole group or "walker". A leader is voted in each walker. To minimize the stick-slip phenomena (sudden jerk forward due to friction), inertial deviations and counter-reaction, the leader will synchronize the movement of catoms. This leader coordinates all motions inside the walker by choosing which catom have to move and by locking others. A such semaphore is necessary because two catoms cannot move simultaneously in a walker of 3 catoms.

Walkers may have different movements strategy related to their size. Walkers would form a chain or "snake" instead of a pyramid. To simplify the control, each Walker consists of 3 catoms and only one catom at a time is allowed to move.

The decision to move a catom is taken at the group level, but has been implemented locally in the moving catom and in all the ones it will touch on its way to the designated place. After each feature motion step, the moving catom will determine which is the next of its feature to actuate and which is the target features to reach among features of the others catoms of the walker. By this way, catoms are rolling feature after feature to the intra-walker destination position (see Figure 10). Selecting a wrong stopover feature during this journey (or having a wrong knowledge of the walker configuration) will cause the dislocation of the walker (see Figure 11).

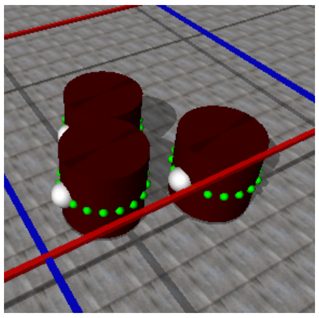


FIGURE 11: A walker dislocation due to a wrong next feature hop choice

3.2. *The walker behavior in a cluttered environment*

3.2.1. *Obstacle detection*

To detect obstacles, several technical solutions can be considered. A proximity sensors could be an ultra-sound sensor, it could also be a kind of radar using radio waves. It could measure infrared signal attenuation from a transmitter to determine its relative position. Last but not least, it could directly rely on the "features" normally used to move and sense their state (electrical current and charge). In this work, we consider a generic proximity sensor and the collision information is extracted from the DPRSim simulator itself to be transparently made available to the code running on the catoms.

Obstacles present in the environment are usually over-sized related to the walkers and will restrain their freedom of movement. Whatever the technical solution for the sensing, a physical collision may occur while a catom is moving from one position to another inside the catom (we call it "intra-walker positioning"). This catom has then to roll back to its previous "stable" position. This "stable" position can be found within the position history and is used to select the next branch in the backtrack tree. The "triangle" formation is easy to model and network relaying within the walker is not necessary because each catom has all other catoms of the walker as neighbor.

By "fumbling" over the obstacles and moving around, a local map is progressively built. When a collision with an unintelligent / non communicative object has been confirmed, the status of the cell is locally stored in the internal map of the walker leader. Collisions with other walkers are recognized as such and therefore not broadcast. It is nevertheless interesting to note that by regular sending of beacons containing their position, walkers are able to announce their "territories" and consequently anticipate collisions. Also, temporary contact between different walkers may also be beneficial for energy transfer or high bandwidth data exchange via wired contact interfaces.

Upon a collision with an unintelligent object and the consequent rolling back to a triangle shape by the concerned catom, another catom of the walker is chosen by the leader to try another intra-walker-motion.

3.2.2. *Intra-walker backtrack : storing catoms movements at the walker scale*

Sometimes, depending one the environment, when obstacles are small, several or all catoms composing the walker may successively collide with various obstacles. When a catom fails to perform an intra-walker move in a specified direction, the failure is stored and reported to the leader. This catom may then attempt to process an intra-walker rolling in the opposite direction but, in this case, the walker would temporarily go far away from his destination. The leader can then preferentially select intra-walker movements operated by other catoms. Independently of the movement strategy adopted by the walker (integral rolling of a determined catom of the walker up to reach again a "triangle" formation or otherwise moving each catom of the walker one after another to control the walker with a "snake" behavior), the walker may always be stuck in a pit or local optimum.

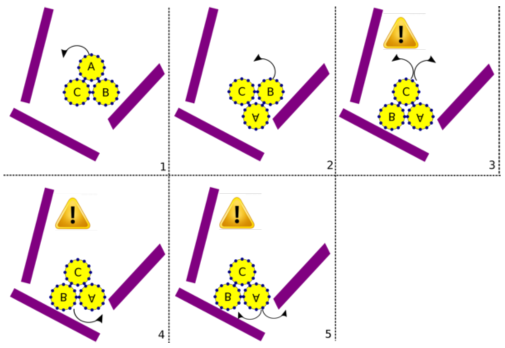


Figure 12: A walker arriving in cavity trap (steps 1 and 2). It is then unable to make new moves (steps 3, 4 and 5 results in collision). The only way out is to roll back the catom B.

Figure 12 shows a walker arriving from the top and traveling to the bottom. The walker gradually traps itself in a cavity designed by three obstacles. Step by step, motion possibilities of the walker are less and less numerous. From step 3, the catom C is unable to move in two directions. In step 4, the catom B tries then fails an intra-walker rolling to the right. In step 5, the catom A realizes that it is also trapped. At that point no more new movements are possible without triggering a collision (candidate movements in steps 3, 4 and 5). The only way out, managed by the leader, is the roll back over previous successful moves. We denote that another motion strategy like the "weaving" does not avoid the problem. The granularity of intra-walker moves would be however lower. As result, walkers would be still bogged down but in finer obstacles. To prevent the walker to be trapped locally in such cavities, it is essential to have a intra-walker backtrack mechanism. To avoid such situations, a intra-walker backtrack mechanism has been implemented in DPRSim. This mechanism is used only for local motions. For map exploration, another algorithm will be presented in next section.

3.3. *The walker exploration algorithm*

To better coordinate the distributed mapping of the environment, and to provide a homogeneous measurements, our application use an overlaying grid. It is used to discretize the space and define sub-areas to explore. Although the resolution is not defined in classical DPI unit, this dimension is however related to the diameter of catoms but also to other physical dimensions used by the simulators (i.e., Watt, meters or bits/second used by Vouivre or related to Newton, Kilogram or g used by ODE). This grid helps to prevent walkers from staying in the same general area and also prevent them to cross other areas without exploring them much.

The efficiency of the exploration strategy can still be improved by trying to go around an obstacle once it has been encountered. This translates into the leader preferring movements exploring unknown space adjacent to already encountered obstacles. Complex environments with multiple large-scaled obstacles, pose again the problem of local optimums. This is where the global grid described earlier becomes handy, as it enables a second level of backtrack. These two levels of backtrack (intra-walker backtrack and spacial grid backtrack) do not have the same function. The grid backtrack is used to correctly explore the whole map discretized in cells (think of navigating over a large area), whereas the intra-walker backtrack is used to get the walker out of a jammed position (handle the details of the road). The last image in Figure 20 shows a walker using the backtrack mechanism related to grid to leave an already scanned area in a complex environment.

3.3.1. The autonomous walker motion algorithm with backtrack

A walker can move autonomously in the environment. To face large obstacles with complex shapes such as labyrinths, it is necessary that walkers have their own individual movement intelligence. Thus each walker built its own map of the environment and walkers are able to move without being trapped into a local optimum.

Indeed, the geographical positions of other walkers and the map structure itself mean that walkers may, temporarily or permanently, be unable to communicate with their partners. Partitioned networks may temporarily be formed.

At the beginning of the simulation, walkers are located at random positions on the map. We can consider that are dropped into a totally unknown area. Some walkers may be jailed inside hollow obstacles from which it will be impossible to get out (see Figure 14).

The initial connectivity may be bad, but through movements and communications they will try to achieve and maintain a tree network topology. However, even under this assumption, the map can change with time and new obstacles may appear.

Similarly, it is possible that walkers are not numerous enough to scan the space and must break the network connectivity to go to explore.

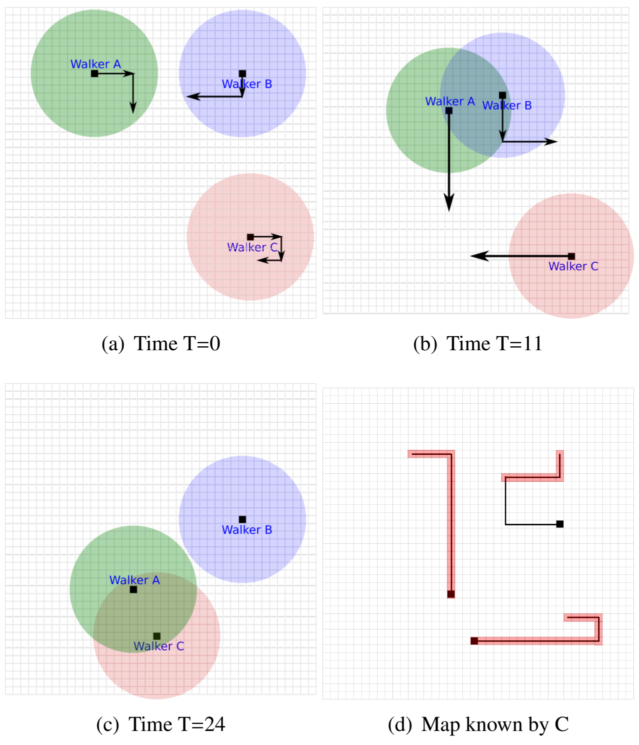


FIGURE 13: Map knowledge can be carried out or broadcasted

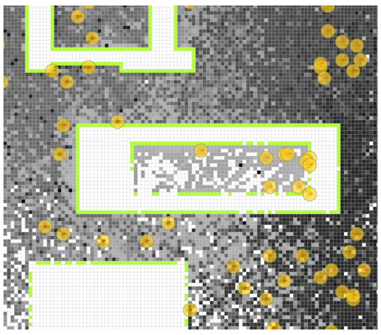


FIGURE 14: Some walkers jailed inside a hollow object. Maintain a network coverage is not always possible.

It is not possible to guarantee an overall network coverage in an unknown space. As a consequence assigning exploration areas to walkers may be counter-productive and fail to assure a reliable network coverage but fail also to assure that walkers will be able to move to areas.

In some situations, the physical carriage of data (see Figure 13) may be the only one way to share data. If the radio coverage is unavailable, we can imagine, for example, a walker which would move alternatively from a side of a jail to another to relay by carriage some informations between 2 others walkers. By the way, map knowledges can virtually travel through radio waves but also physically through the movement of robots.

Figure 13 shows an example of these two mapping informations travel opportunities. All walkers are out of radio communication range and start to move in indicated directions. Walker A and Walker B are able to communicate together by radio. They exchange their map knowledges. Walker A and Walker C are able to communicate together by radio. Walker A carried the chunk of map (known at T=11 by the walker B) and share it now with Walker C. The black line shows the complete overview map. The red background is the the map known both by Walker A and C. Map knowledges of walkers grow with time. The temporal windows for wireless data transfer is small due to radio coverage and walker moves. With short punctual radio link, a walker can carry the shared map of another walker.

The main proposed algorithm allows to find a path to the next undiscovered cell which is the closest from the current walker position. It is a breadth-first traversal which favors a global direction.

Adjacent cells to the current position which contain no obstacles are firstly listed. These cells may be undiscovered or may have been previously explored. In the case which only one of these cells is undiscovered, this cell will be the next walker destination.

In the case which several adjacent cells are undiscovered, it is necessary to choose and to define a priority in the choice of the immediate destination. So the list of next hop destination is sorted by exploration priority order. This local priority avoids the global convergence of all walkers to the same destination. Indeed, if the priority is constant, the walkers will gradually converge globally to one of eight possible directions (north, south, east, west or combination as north-east, etc...). These priorities can be defined to be constant (cf.), to be random (cf.) or by others kind of algorithms. With the random priority, the total path length of walkers is very high and it is often necessary to go back because some cells have been let during the travel.

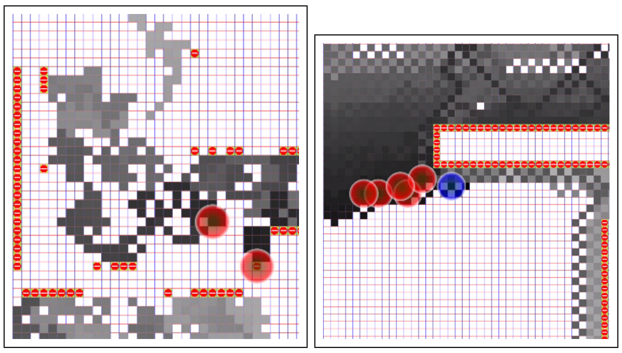
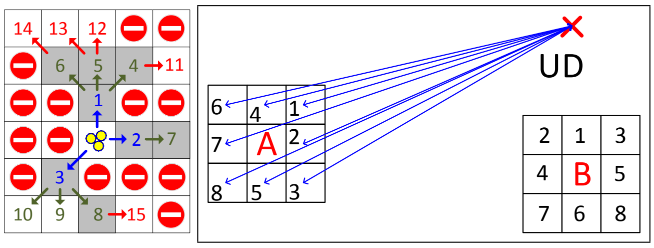

(a) Random adjacent cell priority. (b) Constant adjacent cell priority.

Figure 15: Zoom of map exploration related to adjacent cell priority



(a) Local map exploration algorithm (b) The utopian destination parameter acts as a compass and gives a direction.

Figure 16: Local and global map exploration algorithms

To avoid both these problems we implemented an algorithm which change the adjacent cells choice priority for each walker. We choose for each walker a global destination at the scale of the map. This destination may be unreachable. So we call it utopian destination. The walker will try to go to this utopian destination without necessarily being able to reach it. For each walker, the utopian destination parameter acts as a compass and gives a distinct global direction by defining a rotation center to select adjacent cells. Figure 16(b) shows this algorithm. Point A and B are two possible cells in which a walker can be. UD is the utopian destination for this walker. We define the exploration priority (numbered 1 to 8) of adjacent cells related to the distance between UD and these cells.

Note that if this utopian destination change over time, makes it possible to control the exploration of successive areas by the walkers. A hierarchy could be implemented to allow to a local walker to distribute exploration areas to others walkers.

As soon as all adjacent cells have been explored, we need to look at the next - larger - periphery to find a path to an undiscovered cell. Figure 16(a) shows an example of the main algorithm. Gray cells have been previously discovered, white cells are undiscovered and others cells contain obstacles. In this example, cells 1,2 and 3 (numbers written in blue) are already discovered. As a consequence of the position of the utopian destination, we will look at the next cells in this order : 4, 5, 6, 7, 8, 9 and 10 (cell numbers written in green). This is the next periphery level. The cell 9 is undiscovered (no gray background) and will consequently be chosen. Please note, that this algorithm does not provide the optimal path in all situations. Indeed, supposing that cells 9 and 11 would contain an obstacle, the cell 12 would be consequently chosen. The most direct path would be to cross through the cell 5 instead of the cell 4. This algorithm can be improved to obtain more direct paths. Each time a new periphery is determined, it would also be necessary to reorder all paths priorities in function of the utopian destination. Figure 17(b) shows a lone walker (after 140 sec.). The top of the figure has not been explored because this walker want to reach his utopian destination located in the south of the map. However the walker need to turn around obstacles and to go through a labyrinth before being able to reach his utopian destination.

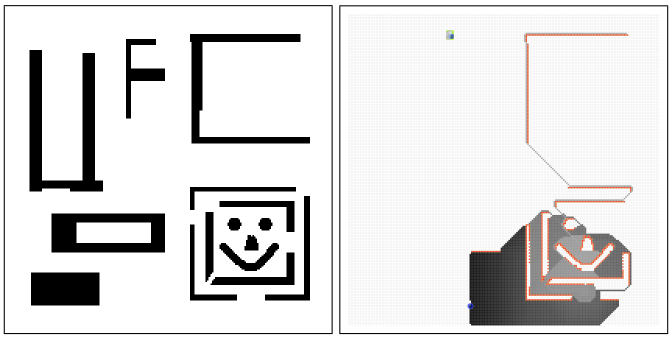

(a) Environment used in large scale simulations : Matrix of 40k cells (eg., 200*200) (b) A walker going through a labyrinth.

Figure 17: Original map on the left and discovered map on the right

Figure 17(a) shows the map used in large scale simulations. Please note that all maps proposed in this paper let cells appear in different gray level. This indicate the path history of walkers. Black cells have been recently discovered and light gray is used for old discovered cells. A wrong way logo is used to represent an obstacle. The walker size is actually smaller than the cell size but to see walker positions on big scaled map, some circles have been drawn in some maps. Blue circle indicate which is the walker who is drawing the map. Green walkers are the 8 last senders of the latest received messages. The last known positions of a others walker is yellow or red depending on simulation. In some maps, data sinks have been added. A small server icon has been used for it.

Please note that in highly detailed simulations (cf. 4.1) realized with DPRSim, the utopian destination parameter has been manually set for all walkers to the center of the map. In high scaled simulation (cf. 4.2), this parameter is randomly chosen for each walker at the beginning of simulations.

3.4. *The network policy algorithm*

As explained previously (3.3), the individual walker exploration algorithm has an intrinsic effect over the network topology. Analogously, setting up a network topology will immobilize walkers which has a cost on the map exploration. Moreover, this is not always necessary. Under some conditions, migrations of walkers will be performed. Some will have to turn around obstacles and topology will change. It can change at different scales : locally, globally or even to dynamically adapt related of data stream. Conversely, a good or bad map sharing will heavily influence walkers in their explorations directions choice. All these factors are strongly nested, and choices to be taken by walkers are themselves influenced by the partial knowledge of walkers them-selves. Indeed, the walkers do not know which map data other walkers have. The map dissemination policy over the network is consequently an important factor. Each walker contains 3 distinct maps in memory (see middle of Figure 18) :

— the map exclusively received by nano-wireless messages
— the map self-discovered by the walker
— the map whole map which is a merge of the 2 previous map

When a cell is visited by a walker, the self-discovered map and the whole map are updated. Similarly, when a message is received, the network map and the whole map are updated.

Each map information exchanged by radio messages contains 4 informations : the date of the visit, the position of the cell visited, the identifier of the walker which visited the cell and the status of the cell. Each map information is 17 bytes length. For each message, some simple headers (12 bytes) are added : the number of map information contained in the message and the identifier of the walker sending the message.

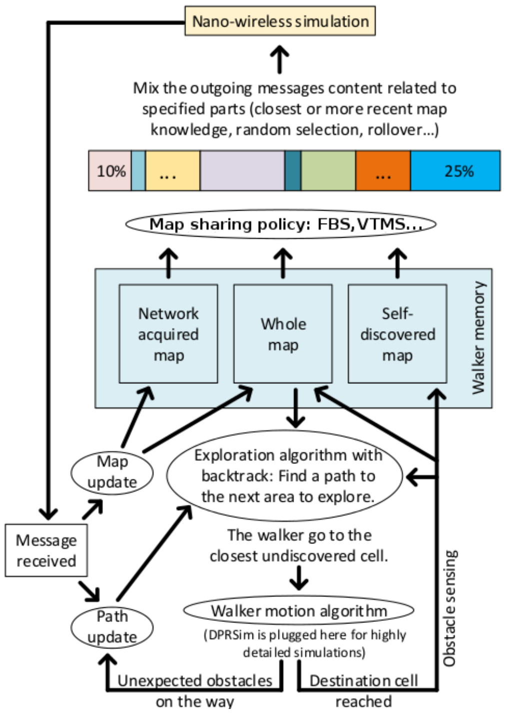

Nano-wireless simulation
Mix the outgoing messages content related to specified parts (closest or more recent map knowledge, random selection, rollover...)
10%
...
...
25%
Map sharing policy: FBS,VTMS...
Network acquired map
Whole map
Self-discovered map
Walker memory
Map update
Exploration algorithm with backtrack: Find a path to the next area to explore.
Message received
The walker go to the closest undiscovered cell.
Obstacle sensing
Path update
Walker motion algorithm
(DPRSim is plugged here for highly detailed simulations)
Unexpected obstacles on the way
Destination cell reached

FIGURE 18: Walker behavior

Please note that the message length will be increased before being sent on the radio medium due to redundancy used 2.3.4. This redundancy will be applied on the network layer. This redundancy will increase the communication range as explained in section 2.3.3. At the application layer, another type of redundancy is used depending of the network policy.

Figure 18 shows the individual walker behavior in the cooperative map building application. The bottom part of the Figure shows the "Walker motion algorithm". This is the optional plug-in connection with DPRSim for highly detailed simulations (cf. 4.1), or at the contrary, some simplifications presented in 4.2 for large scaled simulations. When a walker receives a message containing new map information about his destination, he updates his current path. Another destination will be chosen related to previously explained algorithms. In order that all walkers and the sink acquire the map, it is necessary to relay map informations. Cartographic knowledge are disseminated by multi-hop. However systematically relay all messages would be too expensive in communication cost. The walkers travel on the map. As a consequence they need be able to get a recent knowledge of the immediate environment. It is necessary that despite walkers movements, the knowledge acquired locally in an area persists locally in this area after the departure of the walker : A walker go out but his knowledge stay on the place. Thus walkers acquiring local environment knowledge through network when they arrive in an area. Then they improve this knowledge by themselves. Finally, when leaving the area, they leave it as a legacy to closers walkers which are incoming in this area. On the other hand, walkers can physically carrier some map knowledge in their memory. So they can redistribute it later to new walkers meet during their journey. To take advantage of these possibilities, we defined four algorithms which are mixed together related to fixed proportions. When the walker sends messages (see top of Figure 18), it can promote the local map share, the overall map informations relaying, the local map information relaying, the dissemination of its own acquisitions, etc.

— The first of these algorithms is the selection of visits by date. Only the most recent visits are added to sent messages.
— The second of these algorithms is the selection of proximity maps knowledge. Only informations concerning the closest to the current cell position of the walker are sent.
— The random selection is used to send some map knowledges of a walker which are randomly chosen. It may be long away or old knowledges but it allows a overall and uniform dissemination of the map.
— The last implemented algorithm is the rollover. After each message sending, a looping counter is incremented on all informations contained inside a map . Thus all the data will be periodically resend. This promotes global sharing of all map knowledge.

Note that all these algorithms can be applied equally to one of the three walker maps. Applied to self-discovered map, it help to spread of self-acquired-knowledge. Applied to the network acquired map, it promotes knowledges of others. Message content is an important factor but other parameters must be taken into account in maps exchange policies. First, the regularity and frequency of sharing. Depending on the walkers mobility , providing informations more or less frequently will deplete the local area knowledge persistence. Indeed, if walkers often change area, knowledge will flee with them. Conversely, a too high a frequency sharing provide an excessive information redundancy.

We developed two methods triggering the information sharing :
— Frequency Based Sharing (FBS) : Messages with a fixed length are regularly sent.
— Variation Triggered Map Sharing (VTMS) : Messages of variable length are sent when the map updates reach a threshold. For instance, after 20 new cells discovered or after 80 updates, a message need to be sent. (this parameter have been used for simulations)

4. Simulations and results analysis

The multi-scale algorithms presented in section 3 have been implemented. Results of various policies are presented and discussed in this section.

4.1. Highly detailed simulations in DPRSim

Simulations presented in this section have been realized with DPRSim. This allows to simulate step-by-step, feature after feature, each movement processed by catoms within walkers. These simulations demonstrate the proper functioning of local algorithms for obstacle detection and for local backtrack (at the level of a cell). By this mean, walkers are able to move from a cell to another, are able to detect small obstacles existing inside cells and are also able to turn around these small obstacles to reach the next cell.

20 walkers have been used to scan map of 20*20 cells. Two distinct environments were simulated. Figure 19 shows the first one and the map build through nano-wireless communications. Figure 20 shows a cluttered environment with the associated map.

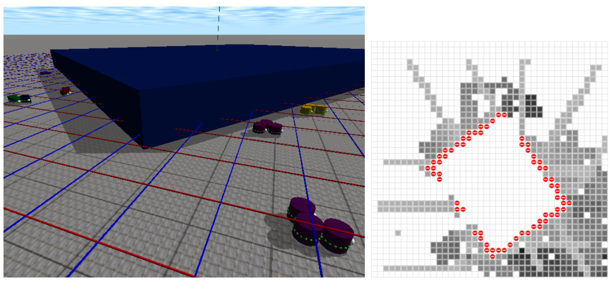
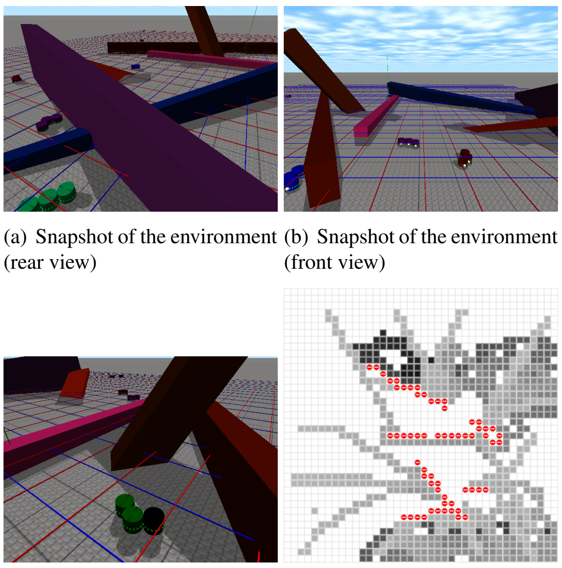

(a) Several walkers scanning a simple environment (b) Ground floor map overview (tick 9960)

FIGURE 19: Exploration of an environment composed of a giant regular square box in DPRSim


(a) Snapshot of the environment (rear view) (b) Snapshot of the environment (front view)

(c) Example of a walker escaping a cavity trap with backtrack (d) Map overview at tick 14940

FIGURE 20: Walkers scanning a complex environment

However interfacing a time-slicing based simulator (DPRSim) with our discrete event simulator remains costly due to synchronism implied. The low number of walkers, the small dimensions of the map and the large number of motions simulated does not highlight significant losses network messages.

4.2. Large-scale simulations and study of the network policy

In order to simulate a large number of walkers in larger environments, we set the speed of movement of walkers at 0.05 m/s and disable the detailed simulations plugged with DPRSim. The map exploration algorithm is initiated using discrete events and not anymore from DPRSim ticks. Cell incoming or cells outgoing events of walkers are no longer generated as effects of the motions of catoms within walkers but simply based on travel distances that walker have to move to go from one cell to another. When a walker arrives in a cell containing an obstacle, he will have to go back to the previous cell (cell which was free). In this case, we believe that due to the presence of the obstacle, the walker has only covered 75% of the distance originally planned. To increase simulations performance, we simulate only walkers and considering that only walker leaders are able to use nano-wireless communications.

Simulation identifier	1	2	3	4	5	6	7	8
Map sharing behavior	FBS	FBS	FBS	FBS	FBS	FBS	VTMS	VTMS
Message length (in number of cell informations)	40	80	35	35	500	100	Variable	Variable
Message sending frequency	175ms	350ms	35ms	35ms	100ms	5ms		
Part of closest self-discovered map of the walker	10%	3%	10%	25%	2%	5%		+10%
More recent self-discovered map of the walker	30%	20%	15%	25%	35%	10%		
Part of randomly choose self-discovered map	7%	8%	10%	10%	5%			
Part of rollover in self-discovered					10%	15%	+5%	+5%
Part of closest network map acquired by the walker	15%	10%	5%	10%	10%	10%		+20%
Part of recentest network map acquired by the walker	25%	30%	30%	20%	18%	20%		
Part randomly choose in the network map acquired	13%	20%	30%	10%	5%			
Part of rollover in the network map acquired		9%			15%	40%	+10%	+15%
Part of recentest whole map acquired by the walker							100%	100%

Table 1: Parameters used in large scale simulations. (FBS : Frequency based sharing ; VTMS : Variation Triggered Map Sharing.

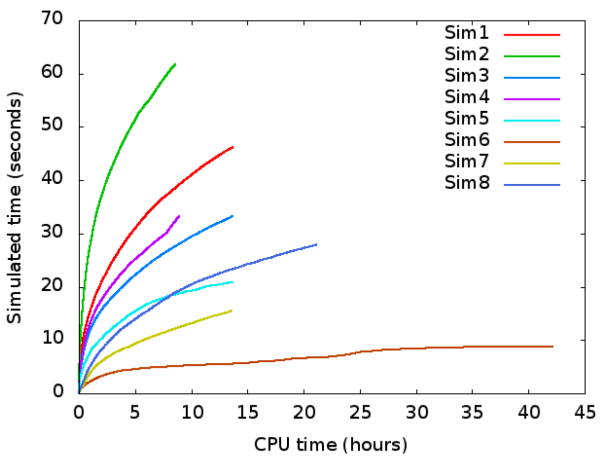


Figure 21: CPU time related to simulated time for 500 nodes. (1 CPU core 2.4Ghz / 6 GB RAM)

As mentioned in the previous sections and by Figure 9, it is possible to simulate radio communications for more than 1200 nodes. Simulations presented here have been however realized with 500 walkers in an environment of 40 000 cells (200 * 200) described by Figure 17(a). We have performed 8 simulations by changing map sharing policies. The values used for the various parameters are available in Table 4.2.

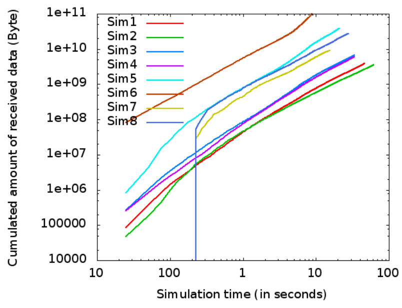
Figure 21 shows the computation duration required related to simulated time. The discrete events simulator uses only one CPU core.

Simulation 1 and 2 have the same number of map knowledge to share (228.5 map informations per second per walker). The simulation 1 sends smaller messages but more frequently than the simulation 2. Figure 22(b) shows the total cumulated amount (in bytes) of raw data sent by all the walkers during the simulation. We note however that the curves are distinct. This is mainly due to messages headers.

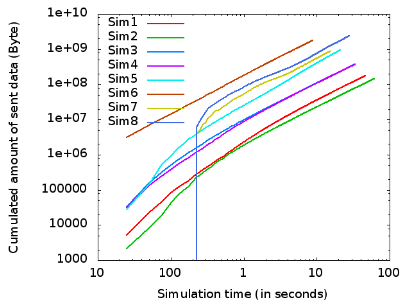
However, a message being sent cannot contain duplicated on map informations. So in the case which visits provided by various algorithms explained in 3.4 are the same, the message size can be slightly reduced related to the nominally fixed length. Indeed, sometimes the most recent visits are also the closest visits. The total (cumulated) number of messages sent and received by all the walkers during simulations are shown by Figures 22(c) and 22(d). They are constants for FBS (Frequency based sharing) simulations (1, 2, 3, 4, 5 and 6). For simulations 7 and 8, there are no messages sent at the beginning of simulation due to the trigger which is not reached. Figure 22(a) shows the total (cumulated) of raw data received by all the walkers during simulations. We note that after 10 seconds, in some simulations walkers received more than 50 GB of data.

It is noted that less data are sent in the simulation 5 than in simulation 8 but more data are received in the 5 simulation. This is due to the time-spreading of the messages in simulation 5 whereas the triggering mode of the simulation 8 creates a synchronism in the message sending. Thus the maximum number of concurrent senders rises up to 16 in the simulation 8 which causes some messages losses. The communication range is consequently reduced.

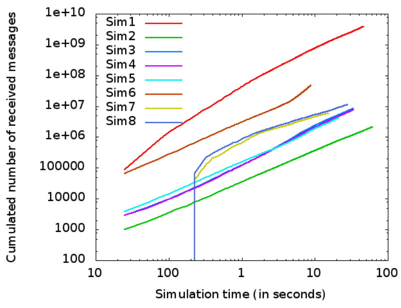
The aggregated throughput snapshot for data incoming and outgoing is indicated for simulations 1 and 5 and 7 by Figures 22(e), 22(f), 22(g), 22(h). Simulation 5 sends large messages.



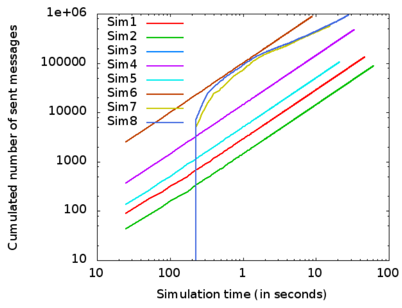
(a) Received data amount



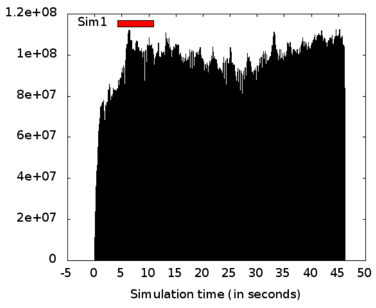
(b) Sent data amount



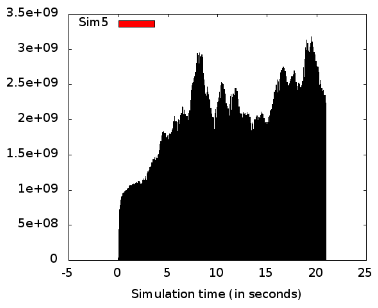
(c) Number of received messages



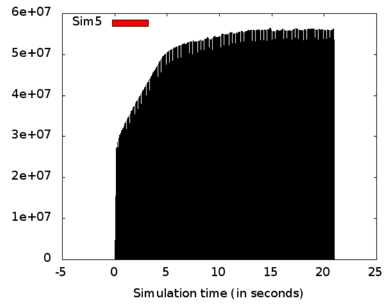
(d) Number of sent messages



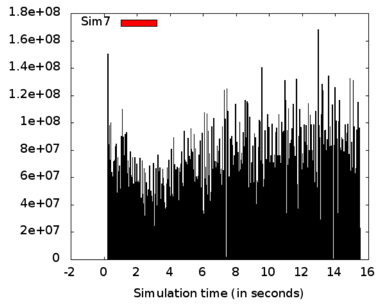
(e) Incoming aggregated throughput (sim1)



(f) Incoming aggregated throughput (sim5)



(g) Outgoing aggregated throughput (sim5)



(h) Outgoing aggregated throughput (sim7)

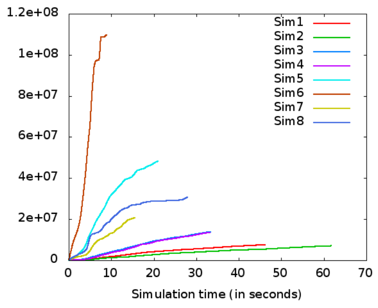
FIGURE 22: Comparison of cumulated data amount and cumulated number of messages during simulations

At the beginning of the simulation, the nominal size of the message cannot be reached. For this reason the outgoing aggregated throughput increases gradually before to get stable. For the simulation 1, the incoming and outgoing aggregated throughput are very stable during the whole simulation.

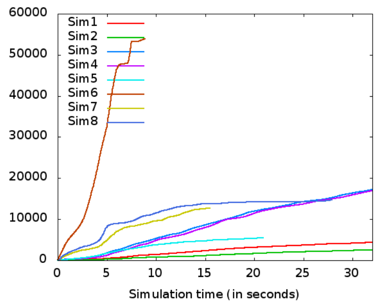
Regarding simulations 7 and 8, the raw throughput is very irregular due to avalanche effect. The arrival of new messages will result in new knowledge which triggers sending new messages. This can be seen on all throughput relating to these simulations :

— incoming and outgoing aggregated throughput snapshot of raw data (see 22(h))
— incoming throughput snapshot of raw data on the sink (see 24(c))
— and also incoming and outgoing throughput snapshot of a majority of walker like the walker 998 for example (see 24(f) and 24(e)).

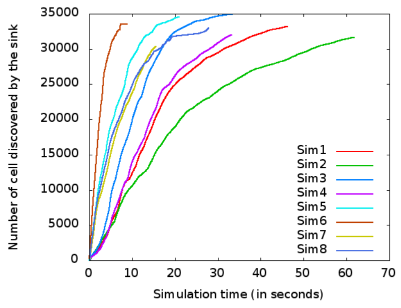
Figure 23(c) shows the cumulative number of new cells discovered by the sink during simulations. In the case which cells are already known by the sink, dates of visits and latests known position of the walkers is updated.



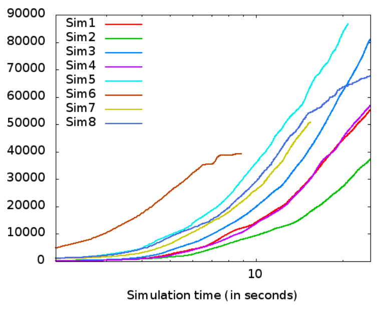
(a) Cumulated amount of data received



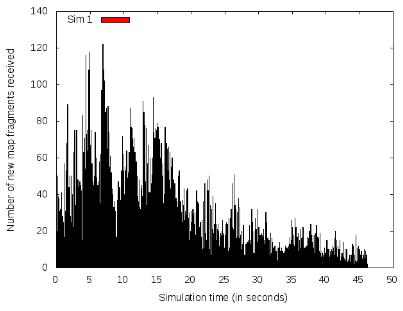
(b) Number of messages received



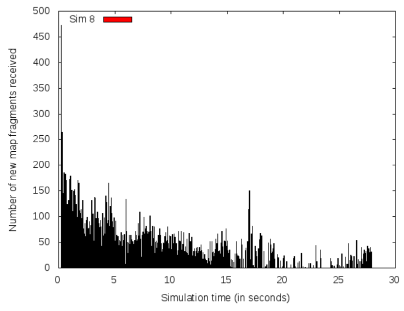
(c) Acquisition of new cells



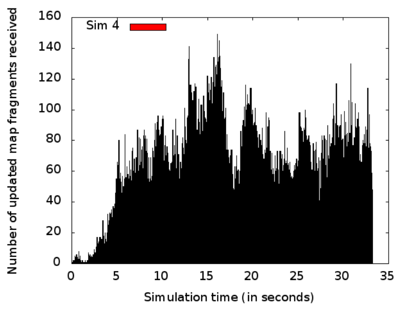
(d) Updates of already known cells



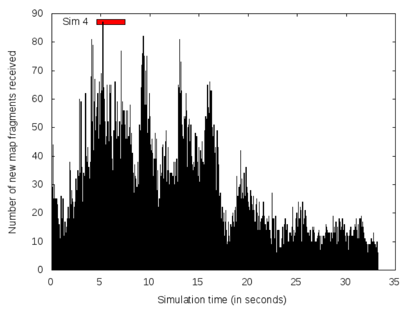
(e) Sim1 / Acquisition of new cells



(f) Sim8 / Acquisition of new cells



(g) Sim4 / Updates of already known cells

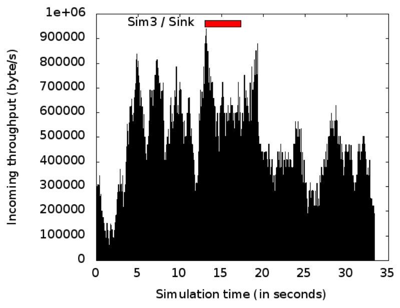


(h) Sim4 / Acquisition of new cells

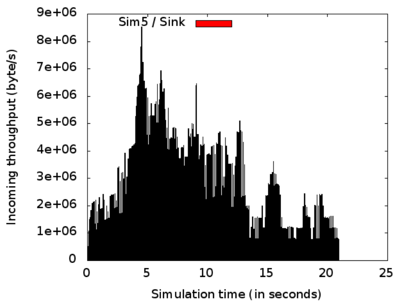
FIGURE 23: Sink point of view

The cumulated number of updates during simulations is represented by Figure 23(d). Others sub-figures of 23 bring a detailed view of simulations 1,8 and 4.

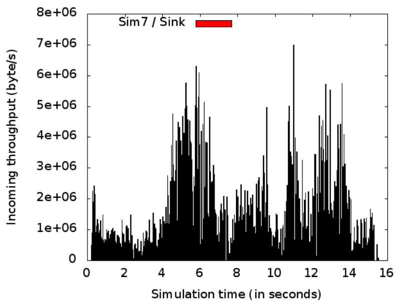
Simulations 4 and 5 send the same messages number at the same frequency. It is noted however that in the simulation 3, the sink acquire rapidly the whole map knowledge. Simulation 4 promotes the spread of self-acquired knowledge whereas simulating 3 promotes relaying of network map knowledge acquired. In case of high walkers mobility, simulation 4 would be more beneficial. In the map structure used for these simulations, this is not the case.



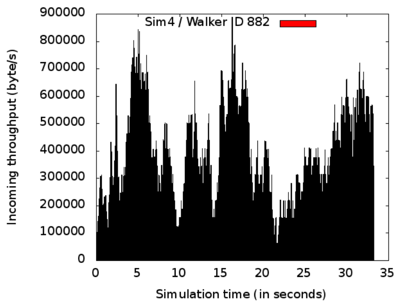
(a) Sim3/Sink/Incoming



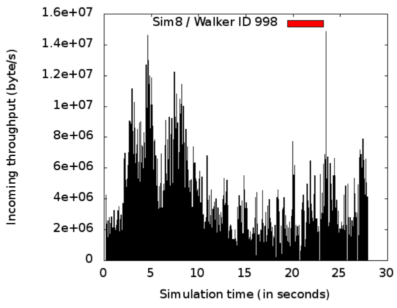
(b) Sim5/Sink/Incoming



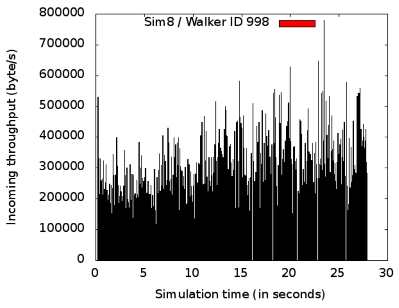
(c) Sim7/Sink/Incoming



(d) Sim4/Walker 882/Incoming



(e) Sim8/Walker 998/Incoming



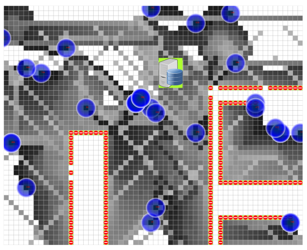
(f) Sim8/Walker 998/Outgoing

FIGURE 24: Incoming and outgoing throughput of the sink during simulations

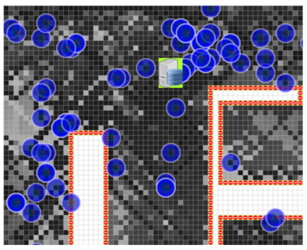
When a walker receives a message which contains map informations indicating that the current destination of walker has been explored, the walker must choose another destination. The number of path recomputed due to new incoming informations arrived by network is shown in Figure 25(a). When this number is high, it shows a good spread of the map. When it is low, it indicates that a lot of walkers are going to explore cells which have been already explored by other walkers.

Analogously, Figure 25(b) indicates in the point of view of the oracle, the number of new cells discovered related to simulated time. The oracle has an instantaneous overall view of the whole map. Thus, if many cases are discovered quickly, it also confirms performances of the network policies in the environment studied. Figure 25(c) shows for all walkers, the cumulated number of new discovered cells acquisition by walkers in their own internal map. As a consequence, it shows the map knowledge propagation speed. However, it must take into account that this is an average and some walkers may acquire a lot of knowledge while others are isolated.

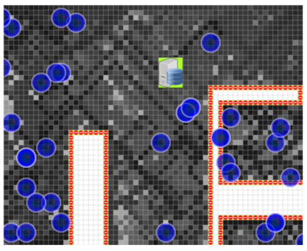
Simulations 7 and 8 show quite similar behaviors



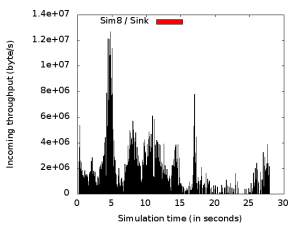
(a) After 2 seconds



(b) After 5 seconds

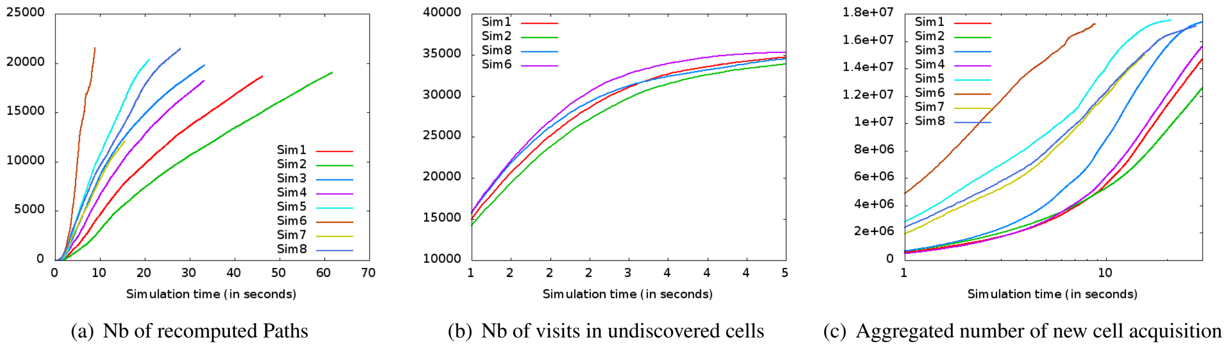


(c) After 6.8 seconds



(d) Incoming throughput (sink view)

FIGURE 26: Zoom of oracle view near the sink in simulation 8



(a) Nb of recomputed Paths (b) Nb of visits in undiscovered cells (c) Aggregated number of new cell acquisition

FIGURE 25: Quality of map exploration (oracle statistics)

due to their operating mode. Note however that the simulation 8 includes 30 % more data (10% on the self-discovered map and 20% on the network acquired map) to improve the map local sharing. We can see a slight advantage of the simulation 8 related to simulation 7 in Figure 25(c). However, the level of mobility walkers is still not enough high to significantly highlight the performance benefit.

Figure 26 highlight effects of the map structure over networks exchanges. The sink is located in north of the map in a cluttered area, at the cross between two ways forcing walkers to get around obstacles to go to new exploration areas. Turning around obstacles need time which temporarily concentrates the walkers around the sink. Is thus noted that after 5 seconds of simulation, the sink receives a lot of data (see 26(d)). However Figure 23(f) does not show additional new cells acquisitions which means that useful data has been previously transmitted via relaying. Indeed, simulation 8 has 15 % more data for rollover so even the most long away map informations are eventually exchanged.

4.3. *Map analysis*

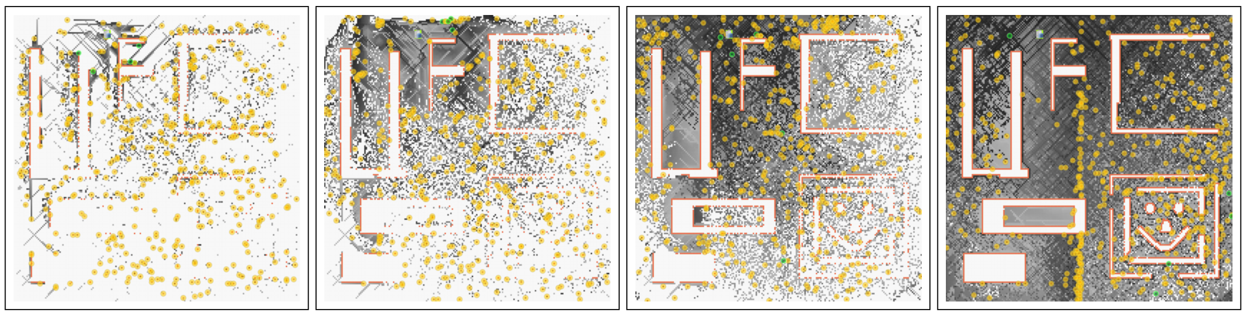

(a) After 1 seconds (b) After 3.2 seconds (c) After 7.4 seconds (d) After 20.8 seconds

FIGURE 27: Sink view in simulation 5

Figures 28 and 28 shows the evolution of map seen by the sink related to time for simulations 3 and 5. We can found two very distinct behavior : The first one for simulations 1, 2, 3, 4 relays progressively the map to sink. The throughput is low, map sharing is lower quality, slower thereby avoiding walkers to converge locally. In contrast, for simulations 5,6,7,8, the map sharing is faster. All walkers have a good knowledge of the map and the exploration of new cells is much slower than the map existing knowledge sharing. This map sharing is too high quality which causes local convergence of walkers. An additional application layer is necessary to avoid local convergence but mostly to maintain a stable network topology.

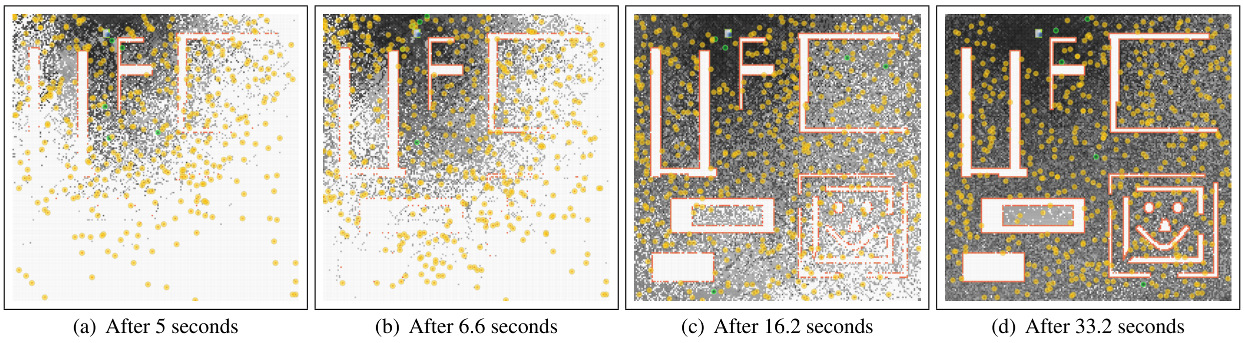

(a) After 5 seconds (b) After 6.6 seconds (c) After 16.2 seconds (d) After 33.2 seconds

FIGURE 28: Sink view in simulation 3

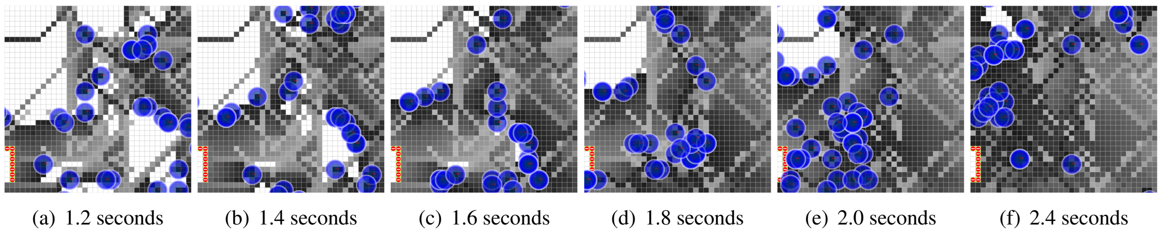

(a) 1.2 seconds (b) 1.4 seconds (c) 1.6 seconds (d) 1.8 seconds (e) 2.0 seconds (f) 2.4 seconds

FIGURE 29: Simulation 8 : A massive assault on the same undiscovered islets causes local convergences of walkers. (oracle view)

Figure 29 shows a zoom of the map of the oracle at different moments of the 8 simulation. Walkers are randomly located in the space at the beginning of the simulation. When walkers starting their journey, they cut the map with their path history into small islets. The local exchange of network map is optimal. So all the walkers know their close environment. Gradually the smaller islets will be explored which few walkers. The larger islands need more duration to be explored, then walkers will be merged together in bigger walker groups. There is an immediate overall convergence but several successive local convergence.

As a consequence, the network topology is changed. Figure 30 shows the view of the sink and the view of the oracle after 27.8 seconds for the 8 simulation. There is still a large number of data routes to the sink and all nodes are able to send data to the sink. However, the data routes are more complex and relaying data requires more hops. On Figure 23(b), we can see that the number of messages received by the sink become stable at the end of simulation. Some walkers have the full map. In this case, they stop to move and they consequently stop to acquire new map exploration by them-selves. Walkers converged and it reinforce their local knowledge. Relaying new data is more difficult. In Figure 30, we can see that the sink view is hightly out date. This is however less visible on the simulation 5 due to a different operating mode.

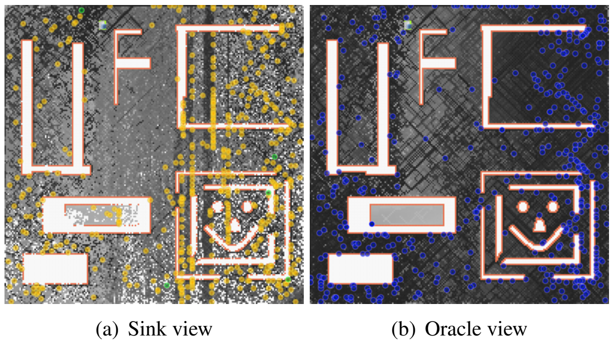

(a) Sink view (b) Oracle view

FIGURE 30: Simulation 8 after 27.8 second : The network topology changed ; Multi-hop relaying need more hops to reach the sink ; Sink view is out of date

5. Conclusion

This paper has presented an application where many small robots explore and cooperatively build a map of their environment. This application shows how beneficial nano-wireless communications can be. Their strong points lie especially in the very low power required and in the large available bandwidth, allowing for high simultaneous communications.

Building complex applications involving many micro-robots is difficult, and dedicated tools are required. We used DPRSim, the original simulator from the Claytronics project, linked to our own Vouivre communication simulation library. The nano-communications were simulated using a discrete events approach, allowing for a level details and precision. On the other hand, DPRSim offers a detailed vision of the walkers physical moves, along with problems concerning obstacle detection algorithms and local backtrack.

The first sets of very detailed simulations we showed, validate walkers motion and allow for shifting to more abstract and large scale simulations. Results of these large-scale simulations highlight a strong entanglement between the map structure, routing algorithm, network topology and the used map exchange network policy.

Together, DPRSim and Vouivre were able to give a very good insight of the behavior of a large scale cooperative application, allowing for the analysis of both large scale (building of the global map, information propagation in the network) and local scale (individual walkers movements and neighboring communications).

Future work would require to consider an intermediate network abstraction layer. It should allow a better isolation between the application and the maintenance of the network topology. At the same time, it should still be aware of the physical constraints to help and to prevent exploration algorithm to mistakenly fragment the network.

6. Acknowledgments

We would like to thank Dr. Josep Miquel Jornet and Prof. Ian Akyildiz for sharing with us their models of the terahertz band. This work has been funded by the Labex ACTION program (contract ANR-11-LABX-01-01) and ANR/RGC (contracts ANR-12-IS02-0004-01 and 3-ZG1F) and ANR (contract ANR-2011-BS03-005)

Références

[1] S. C. Goldstein, J. D. Campbell, T. C. Mowry, Programmable matter, IEEE Computer 38 (6) (2005) 99–101. URL `http://www.cs.cmu.edu/ claytronics/papers/goldstein-computer05.pdf`

[2] S. C. Goldstein, T. C. Mowry, Claytronics : A scalable basis for future robots, in : RoboSphere 2004, Moffett Field, CA, 2004.

[3] M. E. Karagozler, A. Thaker, S. C. Goldstein, D. S. Ricketts, Electrostatic actuation and control of micro robots using a post-processed high-voltage soi cmos chip, in : IEEE International Symposium on Circuits and Systems (ISCAS), 2011.

[4] N. Boillot, D. Dhoutaut, J. Bourgeois, Efficient simulation environment of wireless radio communications in mems modular robots, in : iThings 2013, IEEE Int. Conf. on Internet of Things, Beijing, China, 2013, pp. 638–645.

[5] N. Boillot, D. Dhoutaut, J. Bourgeois, Using nano-wireless communications in micro-robots applications, in : NANOCOM 2014, 1st ACM Int. Conf. on Nanoscale Computing and Communication, ACM, Atlanta, Georgia, USA, 2014, pp. 1–9.

[6] N. Boillot, J. Bourgeois, D. Dhoutaut, Parameter study and characterization of wireless nanonetworks through simulation, in : Communications and Networking (BlackSeaCom), 2014 IEEE International Black Sea Conference on, IEEE, 2014, pp. 43–47.

[7] M. E. Karagozler, S. C. Goldstein, J. R. Reid, Stress-driven mems assembly+ electrostatic forces= 1mm diameter robot, in : Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on, IEEE, 2009, pp. 2763–2769.

[8] B. D. Rister, J. Campbell, P. Pillai, T. C. Mowry, Integrated debugging of large modular robot ensembles, in : ICRA, 2007, pp. 2227–2234.

[9] M. P. Ashley-Rollman, P. Pillai, M. L. Goodstein, Simulating multi-million-robot ensembles, in : ICRA, 2011, pp. 1006–1013.

[10] J. M. Jornet, I. F. Akyildiz, Low-weight channel coding for interference mitigation in electromagnetic nanonetworks in the terahertz band, in : Proc. of IEEE International Conference on Communications (ICC), 2011.

[11] J. Jornet, I. Akyildiz, Channel modeling and capacity analysis for electromagnetic wireless nanonetworks in the terahertz band, Wireless Communications, IEEE Transactions on 10 (10) (2011) 3211–3221. doi :10.1109/TWC.2011.081011.100545.

[12] J. Jornet, I. Akyildiz, Channel capacity of electromagnetic nanonetworks in the terahertz band, in : Communications (ICC), 2010 IEEE International Conference on, 2010, pp. 1–6. doi :10.1109/ICC.2010.5501885.

[13] T. Henderson, S. Roy, S. Floyd, G. Riley, ns-3 project goals, in : Proceeding from the 2006 workshop on ns-2 : the IP network simulator, ACM, 2006, p. 13.

[14] A. Varga, R. Hornig, An overview of the omnet++ simulation environment, in : Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops, Simutools '08, ICST, 2008, pp. 60 :1–60 :10.

[15] http ://www.opnet.com/products/modeler/.

[16] R. E. Lyons, W. Vanderkulk, The use of triple-modular redundancy to improve computer reliability, IBM Journal of Research and Development 6 (2) (1962) 200–209.

[17] J. Knox-Seith, A redundancy technique for improving the reliability of digital systems, Stanford Electronics Laboratory.