Efficient simulation environment of wireless radio communications in MEMS modular robots

Nicolas Boillot, Dominique Dhoutaut and Julien Bourgeois
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, dominique.dhoutaut, julien.bourgeois}@femto-st.fr

***Abstract*—Modular robots needs networking for coordination and it is particularly true for MEMS microrobots. A promising communication technology is nanowireless networking which could be integrated directly into MEMS microrobots, in our case, the catoms of the Claytronics project. A first step towards this objective is to design a wireless simulator able to deal with modular robots. This simulator called Vouivre is integrated in DPRSim a modular robot simulator developed by Intel Research. This paper describes Vouivre and its integration in DPRSim which is an interesting case of integrating different timelines in one simulator. Experiments validate our design and show the interest of using wireless communication in modular robots.**

I. Introduction

Modular robots exist in various sizes and shapes but always rely on a supporting network to allow either centralized or distributed control. Due to their nature and design, modular robots are often unable to act or to move by themselves alone, but instead have to collaboratively interact with their immediate neighbors, thus increasing the importance of the network. This is even more important for MEMS modular micro-robots and more generally for distributed intelligent MEMS projects [1].

Within the Claytronics project [2], [3], a new type of modular MEMS micro-robots have been designed for realizing programmable matter [4]. Each MEMS micro-robot is a sphere called catom (i.e. Claytronics atom) that can stick to and move around its neighbors. An individual unit has very few possibilities but an ensemble of catom is able to act collectively. The communication is therefore of the uttermost importance. As an example, one could take an ensemble of catoms that is required to take a given shape. Individual catoms have to move around and attach to their neighbors, with which they can communicate directly. Intermediary and final positions of the catoms can be predetermined or can be computed on the fly. The structure will hold as long as the current positions and binding allow a fair distribution of the physical forces. If direct communications with the neighboring catoms is possible, all the others robots are potentially reachable trough a routing protocol. As the 3D shapes are built for various concrete purposes (mechanical resistance, motions), they are rarely optimal from the point of view of the routing and network layer. In fact the network topology is constrained by the physical structure. Some nodes may occupy a central position and act as bottlenecks in the communication scheme. On the other hand, communication capabilities of other network nodes may be under-exploited. This is also the case of most modular robots where designs make use of standard wired network layers, such as I^2C, CAN or SPI buses. Depending on objectives and design choices, those buses can be shared directly between all the individual elements, or can be only common to a few neighboring components.

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)

In fact, wired networks are facing other problems when applied to modular robots. They are unable to reach a unit that is not in contact with the ensemble and convergence of distributed algorithm can be difficult to achieve when unit are moving, and therefore constantly restructuring the entire logical topology.

Our long-term objective is to integrate a wireless network inside the catoms to overcome the limitations of the wired communications. There have been some progresses [5], [6] in the design of a wireless device that could be integrated within catoms but there are still many challenges to reach this goal. The first one, is to be able to design a modular robot simulator enabling wireless communications. The second one, is to define a propagation model tailored for the Claytronics environment. Finally, given the constraints of nano-wireless networking, communication layers have to be tailored for the Claytronics applications or any other IoT (Internet of Things) applications, as it is envisioned that IoT applications will be among the the first usage of nanowireles networking [7]

The objective of this paper is to present the progresses made in the design of a wireless communication module within the DPRSim simulator [8], [9].

II. Wireless network simulation for modular robots

There are many network simulators that are proposing wireless network simulations. The two most used in the research community are NS3 [10] and OMNeT++ [11]. They offer support for mobility and this characteristic could enable their usage for network simulation of modular robots. Unfortunately, these simulators don't offer physics and real-world modelling: mobile network nodes are only 2D objects with no support for collision and so on. However, it would be possible to plug an external simulator which could take care of the physics but the network simulation is too detailed for our needs which limits the scalability to thousands of node. The same comments apply to other simulators like SSFNet [12], OPNET [13], QualNet [14] or J-Sim [15].

Many modular robots simulators have been designed over the years. Very first works are simulators which are dedicated to specific hardware like Molecubes[16] or CrossCube [17] and therefore lack of genericity. More recently, there have been more generic simulators proposed like Player/Stage [18], gazebo [19], USSR [20] or MORSE [21] but network access is not sufficiently detailed, as focus is made on the robots themselves. The most interesting initiative is ARGoS [22] which includes its own network simulator or that could be plugged to NS2 or NS3 [23]. ARGoS is fast and can simulate up to 100,000 of robots. However, the target of ARGoS is swarm robots and not really modular robots.

Finally, DPRSim appears to be unbeatable for the scalability as the most efficient simulators report 100x less simulated number of nodes.

III. Claytronic's catoms and DPRSim, their timeslicing based simulator

In the Claytronics project, MEMS micro-robot called ”catoms” (prototype photo 1) are covered with structures called ”features”. Those features are used as mean of both attachment (by electromagnetic or electrostatic force [24]) and direct communication.

Fig. 1: Claytronics catom prototypes (left one with magnetic actuation, right one with electro-static actuation)

By actuating features, catoms are able to move arround their direct neighbors similarly as a stepper motor (see figure 2).

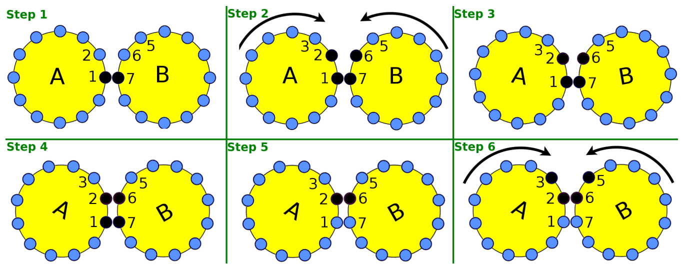


Fig. 2: Actuating the catoms like a stepper motor (2 dimensional motion example)

Dynamic Physical Rendering Simulator (DPRSim) [8], [9] has been developed by Intel since 2006 for the Claytronics project of the Carnegie Mellon University.

The simulator is designed to support a potentially very large number (up to several millions) of Claytronics micro-robots called ”‘Catoms”’. DPRSim internally makes use of ODE (Open Dynamics Engine, a rigid body dynamic library) and can provide detailed physical simulation for its virtual world (albeit activating physical simulation significantly reduces the number of simulated elements). An optional 3D graphical interface is also provided through Drawstuff, the default 3D environment visualization tool included in ODE. For performance considerations, DPRSim is written in C++ and can also work in mono or multi-threaded mode.

In DPRSim, each catom is individually represented by an object instance, and the code running on a catom is called a ”CodeModule” which is also individually instantiated. Someone designing a new Claytronics application has to write its own CodeModule and then load it into the simulated catoms.

CodeModules can be written in C++ but, part of the claytronics project, is also the Meld language. Meld is a declarative language aiming at easing the development of large scale distributed applications. One can thus write Meld programs that are automatically converted into CodeModules able to run in DPRSim.

To reach the simulator large scale objectives, necessary simplifications have to be done, and understanding their implications is mandatory to both a correct use of DPRSim and a correct interpretation of the results it produces. DPRSim uses of a time-slicing approach, 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, especially on the messaging system.

If activated, the physical simulation (ODE library) uses its own time-slicing - which is independent but has to be configured conjointly with the main DPRSim's time-slicing.

Each Catom owns a MailboxManager in which several mailboxes are registered. In the MailboxManager, for each CodeModule of the catom, a specified Mailbox can be linked to a callback function. DPRSIM messages are exclusively designed to be sent through a Catom feature and don't have a catom recipient address. They only have a recipient mailbox and feature. Because their recipient is a low level feature interface, they can be considered as low level messages. But, at the same time, they are directly processed by CodeModules. This effectively makes them also belonging to the application layer. As we will see, this will be challenging to overcome when implementing wireless communications.

Figure 3 graphically shows how the DPRSim's main simulation loop works. All the catoms and their network interfaces are referenced in a ”CatomWorld” object which has to be initialized first. Before starting the main loop, the simulator calls to the simulationStart() method of each CodeModule. Then, for each simulation tick, specific callback functions from all catoms and network interfaces will be called sequentially by the main loop. Each tick processing is thus effectively divided into a few sequential steps denoted A to D on Figure 3:

- Step A: Getting the current position of all catoms from the ODE library and computing the neighboring lists.
- Step B: Individual processing of each catom (detailed later).
- Step C: Updating the states of the magnets depending on the actions undertaken in step B.
- Step D: Advancing the physical simulation by calling ODE (catoms may move only during this step). ODE may use multiple physical steps, but their sequence is then uninterruptible and no message or CodeModule processing can happen during step D.

Step B involves the sequential processing of each catom, this processing itself being a sequence of sub-steps. All sub-steps for a given catom (denoted from 1 to 5 on Figure 3) have to be finished before going to the next catom. This is of utmost importance for the messaging model as we will see by detailing the sub-steps:

- Sub-step 1: Calling the newTick() method of each NetworkAdapter of the current catom. This essentially reset the capacity counter (the number of bytes this Adapter will be able to send during the current tick).
- Sub-step 2: Calling the newTick() method of each CodeModule of

the current catom. Those are user-defined functions that traditionally decides to start movements or communications.
• Sub-step 3: If outgoing messages have been enqueued during sub-step 2, then the system try to send them (the capacity counter is reduced of the size of the messages until not enough capacity is left; eventual remaining messages are kept in queue for a sending in the next tick). The messages which could be sent during a tick are stored in a special queue. Their reception will NOT be processed during the current tick, but during the next one (delayed mechanism).
• Sub-step 4: If messages have been sent during the PREVIOUS tick, then a callback function, defined by the user, is called on the corresponding mailboxes. This is the place a CodeModule traditionally reacts to the messages it receives.
• Sub-step 5: Calling the endTick() method of each CodeModule. An application programmer can put here any code he wants to be executed AFTER all incoming messages for this tick have been processed.

The order in which the catoms are processed (step B) is important. Should the same order be used every tick, we would get a strong bias on the communications (the firsts catoms to be processed would always send their data before those at the end of the list, potentially depriving them of network access). To prevent this bias, DPRSim implement a basic randomization of the catom processing order. This, coupled with the time-slicing mechanism, acts as synchronization barriers. The catoms have to wait for all the others to have been processed in the B step before advancing further.

For the message-passing system, this synchronization means that, whatever their size, messages are received at best during the next tick. The capacity counter for an interface is reduced by the amount of bytes sent. Should it reach zero, any message not already sent would be kept in the transmission queue and further processed in the next tick. A very big message could correctly capture the link as its transmission would also continue over as many ticks as necessary.

The delaying mechanism - the fact that messages are received at the earliest during the next tick - exists to prevent a potential problem. Should the transmission be ”immediate”, the callback method on the receiving catom would be called from the sending method of the one transmitting. But, if this receiver callback starts a transmission, we could have a chain of calls that do not care anymore about scheduling, synchronization or duration of other messages. Forcing the reception at the earliest in the next tick solves the problem, but implies that a duration of, at least, one tick is necessary to react to any incoming message.

IV. Radio simulation in DPRSim

Most of current modular robots and especially Claytronic’s catoms have been designed to communicate exclusively through ”contact” interfaces. Those ”wired” interfaces, should be seen as full-duplex dedicated communications links. As each pair does not share the medium with other interfaces, those links can be simulated independently. With such networks, the arrival date of a message depends mostly of its expedition date, its size, and the eventual presence of other messages in the transmission queue. In Claytronics, as already seen on figure 2, Features are used to allow catoms to stick together (either using magnetic or electro-static forces [24]). DPRSIM is able to simulate two or more remote Features physically attracting each others and attaching themselves.

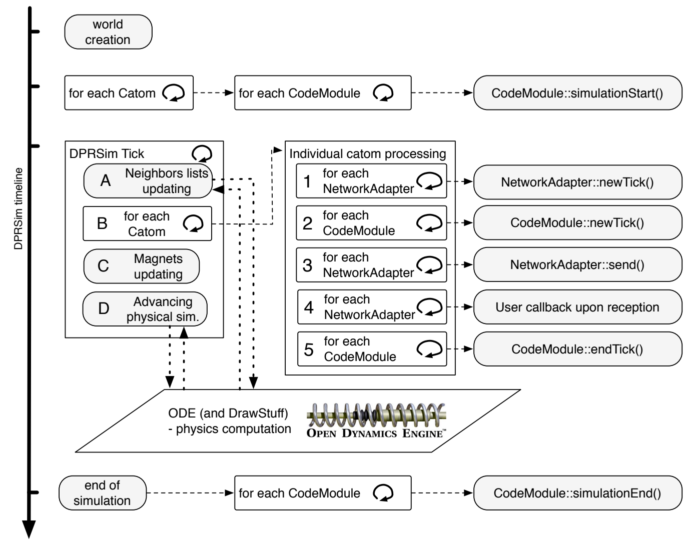

world creation
for each Catom
for each CodeModule
CodeModule::simulationStart()
DPRSim timeline
DPRSim Tick
A Neighbors lists updating
B for each Catom
C Magnets updating
D Advancing physical sim.
Individual catom processing
1 for each NetworkAdapter
2 for each CodeModule
3 for each NetworkAdapter
4 for each NetworkAdapter
5 for each CodeModule
NetworkAdapter::newTick()
CodeModule::newTick()
NetworkAdapter::send()
User callback upon reception
CodeModule::endTick()
ODE (and DrawStuff) - physics computation
Open Dynamics Engine
end of simulation
for each CodeModule
CodeModule::simulationEnd()

Fig. 3: DPRSIM timeline & internal principle : For each stage, callbacks are executed sequentially for all catoms

But Features are also used for communications and thus incorporate low-level communication interfaces (called NetworkAdapter in DPRSIM). The final physical communication process remains unclear and multiple possibilities have been envisioned. This is still quite important as it may change the nature of the links from a network point of view. By modulating the current of an electric field at a high frequency, it becomes possible to use its variations to transmit data. In this case, this network connection behaves as a wireless connection potentially able to reach multiple distant NetworkInterfaces. The link may not be a simple peer-to-peer and full duplex link anymore, and would require a much more complex medium access control mechanism. The actual range and number of reachable interfaces would depend on the physical and geometrical properties of the caroms. In a radically different approach, communications could be made trough conductive surfaces smaller than the feature itself. This surface would act as a connector socket and its position and shape would constrain the link to a peer-to-peer nature. Although a feature of this type would be physically binding, it would prevent connecting three separate catoms on the same ”connector”. This would lead to indirectly consider two types of ”features” : one for physical motion, the second for communication.

Also, as more than 2 features can attract each other, from the networking code point of view, it is possible to associate more than two NetworkInterfaces together. In that case, as the available bandwidth is calculated as the sum of the individual reception capabilities, the more features sharing a link, the higher the bandwidth ! Hopefully, this situation is usually prevented by the physical geometry of the catoms, that does not leave enough space for more than two features to be in contact.

Regarding the simulation of the simpler peer-to-peer communications, as they are independent from each others, they can be more easily processed by a time-slicing simulator such as DPRSim.

However DPRSim display very strong synchronism barriers because of its ”Ticks”. All messages sent during one tick will be received - and processed - at the earlier during the next tick. This can radically change the execution chronology in case of very small messages designed to trigger immediate reaction of the neighboring catoms.

Simulating a shared medium at the usual time-scale of DPRSim (the ticks) cause problems, as the events and actions undertaken by network interfaces are usually of a much smaller time-scale. Some actions may take a few micro-seconds or less, whereas some other may take hundreds of milli-seconds. If large time-slices are used, many small scale actions will take place at the exact same time, ignoring their real chronology. If much smaller time-slices are used, the correctness of the simulation improves at a cost of a lot more computation overhead (affecting the maximum size of the simulation)

Communications through shared wired bus (such as the CAN bus) or through radio waves requires the implementation of a network medium access layer (MAC), a suit of protocols and mechanisms supervising concurrent access. In order to exhibit realistic latency delays and errors transmissions to the application layer, it is necessary to simulate the time in a continuous manner, enforcing anteriority and real duration on all the actions. We thus must simulate the time independently of DPRSIM and with a much greater precision than its usual ticks.

The current design of DPRSim comes from the need to simulate very large numbers of catoms (millions of them), and time-slicing helps a lot as it eases the parallelization of each catom processing. On the other and, much more realistic network simulators tend to not scale that much by multiple orders of magnitudes. Simulating all the independent events and their interactions on a shared medium is computationally excessively complex as the number of interfaces increases.

To preserve the potential scalability of DPRSim and its current timeslicing operating mode, we decided to develop as a library a new network simulator called Vouivre. This independent simulator is based on the discreet-events paradigm, can be easily bound to DPRSim or other simulators, and implements a suitable tradeoff between complexity and realism of the network. Going for millions of nodes sharing a medium is not possible at this time, but it is still possible to tune it to go largely over the few thousands most dedicated and detailed network simulators can do.

To link DPRSim with our new network simulator, 3 majors changes have been brought to DPRSim:

The first one is to invoke at each Tick, a progression of the simulated network time (see figure 4). From the point of view of Vouivre, the duration of a DPRSim's tick is defined as constant. It is important to configure this time jumping duration to be the same as the duration of the intra-tick time interval defined for the physics engine ODE. Consequently, in the figure 4, the duration between t0 and t1 is the same for Vouivre and ODE. Failing to do so, the network time would flow slower or faster than the physics time and the data rates of network interfaces would not have sense anymore. During each tick processing, DPRSim lets Vouivre process the discrete events which have been scheduled for this time interval. From a different perspective, we should consider that the network simulation is periodically stalled to allow DPRSim to simulate CodeModules of catoms and to allow ODE to simulate physics.

The second change brought to DPRSim concerns the deactivation of the previous algorithm dedicated communications on wired interfaces. The existing mailbox system of DPRSim has still been preserved to keep compatibility with already existing CodeModules and Meld applications. To avoid the synchronism barrier induced by the time-slicing operating mode, the existing mailbox callback system have to be used in ”live mode” instead of ”delayed” mode. The figure 7 shows the Vouivre core calling back, in live during the network simulation progress, the receiving function attached to the mailbox of a given catom. This allows for multiple small messages to be exchanged back and forth during the same DPRSim tick. A new type of interface has been added, as the WirelessNetworkAdapter, which enables access to a shared radio channel. NetworkAdapters corresponding to the normal wired interfaces have also been modified to be managed by our network simulator. allowing simulations of wired communication over shared mediums such as wired buses.

The last important change is a mapping between DPRSim structures representing catoms and the corresponding Vouivre network nodes. Each network node can have several network interfaces. They can be wired or wireless interfaces. This mapping is build at the creation of the DPRSim simulation universe called ”CatomWorld”.

To be realistic enough, is is necessary to simulate the packets collisions that can occur on shared medium such as wired buses or radio medium. Collision occurs when multiple transmitters send a message at the same time. But depending on the medium type (wired or radio), the collision may or may not be detected by the transmitters themselves. On a wire, when two transmitters try to send data, they can see that the signal they try to put on the wire is altered by another transmitter doing the same thing. But with the radio medium, the signal attenuation with the distance is much more important. The signal a given transmitter is sending is thus, from its point of view, many magnitudes stronger and completely mask any other concurrent signal. The collision can only occur at the receiver, if multiple signal with comparable strength are received at the same time. This translate into erroneous bits received, which are detected by a CRC and the whole frame is tagged as failed. But the transmitters can not be aware of this directly, and only a missing acknowledgment indicate the collision. In our implementation, an erroneous packet will not be put in the receiver mailbox.

Figure 5 exposes a specificity of radio communications related to other types of shared medium communications: the radio coverage area. Indeed, although the medium is shared, collisions can be located in space. Supposing catoms A and C of the figure 5 are both simultaneously transmitting a message. The range of the radio signal is represented by circles. D will receive the message from A. E will receive the message from C but B will get nothing because from its point of view, the two messages collides and cannot be decoded.

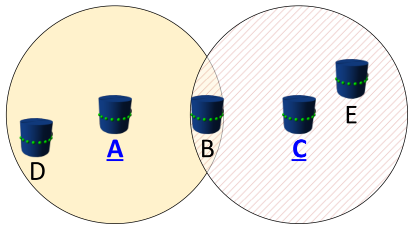


Fig. 5: Collision due to the space position of catoms and the wireless range

More precisely, for each receiver, the attenuation of each signal has to be computed and compared to the others. If a signal is above the reception threshold while being strong enough to mask any concurrent

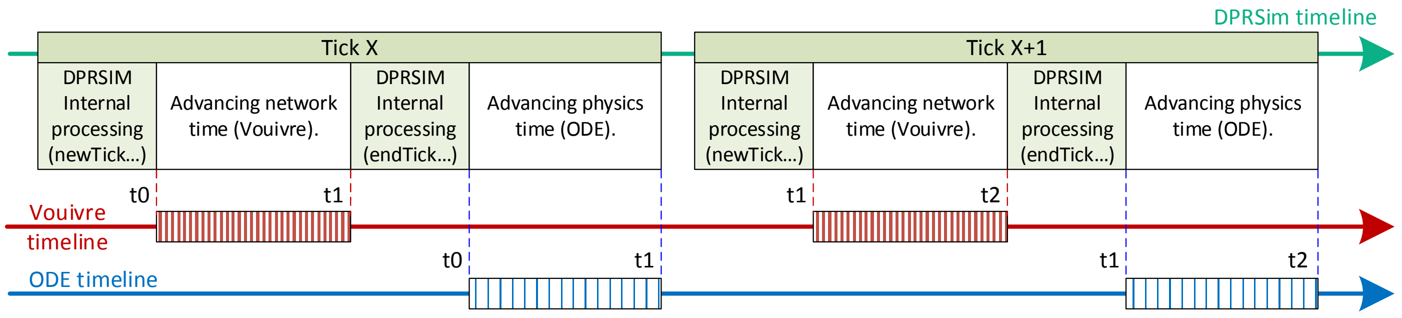

DPRSim timeline
Tick X
DPRSIM Internal processing (newTick...)
Advancing network time (Vouivre).
DPRSIM Internal processing (endTick...)
Advancing physics time (ODE).
Tick X+1
DPRSIM Internal processing (newTick...)
Advancing network time (Vouivre).
DPRSIM Internal processing (endTick...)
Advancing physics time (ODE).
t0
t1
Vouivre timeline
t0
t1
ODE timeline
t1
t2
t1
t2

Fig. 4: Imbrication of Vouivre inside the DPRSIM and timelines

signal, then it is received. We chose, for its simplicity, to implement the FRIIS (Figure 6) free-space radio propagation model. The model of FRIIS takes only in account the position of catoms in space. This position is refreshed from ODE by our network simulator via a callback function. This simple propagation model could easily be replaced by a more complex one.

$$\frac{P_r}{P_t} = G_t \times G_r \times \left(\frac{\lambda}{4 \times \pi \times R} \right)^2 \tag{1}$$

Fig. 6: Friis transmission equation. Pr and Pt are the received and transmit power in Watt. The wavelength λ and R the distance between antennas are in meter. G_r and G_t are the gain of receiving and transmitting antennas in dBm.

Standards and high level network protocol stacks such TCP/IP, IPX/SPX, WiFi or protocols close to hardware equipment such as CAN or I^2C do, for several obvious reasons, a strong distinction between network messages, packets and frames. There is a fragmentation and encapsulation of data which brings many advantages from the point of view of the network. The network model that we developed is simple and aims to simulate the lower network layers avoiding to fragment and reassemble messages as it would be costly to simulate.

To reduce the risk of collision, we adopted a simple system based on CSMA / CA (carrier sense multiple access with collision avoidance). Before sending a message, a random delay called backoff is chosen by the radio interface (this delay is based on a random number generator independently seeded for each interface). This waiting period is expressed in time-slots of fixed duration. At each time-slot, radio interface listen to detect if another radio transmissions is already present. If another transmission is detected, the current interface postpone its own transmission (deferring period). When the medium is free again, it resumes the decreasing. When the backoff successfully expires, the transmission begins. This mechanism reduces the number of collisions by ensuring fair access to the communication medium. Most collisions are avoided but still possible.

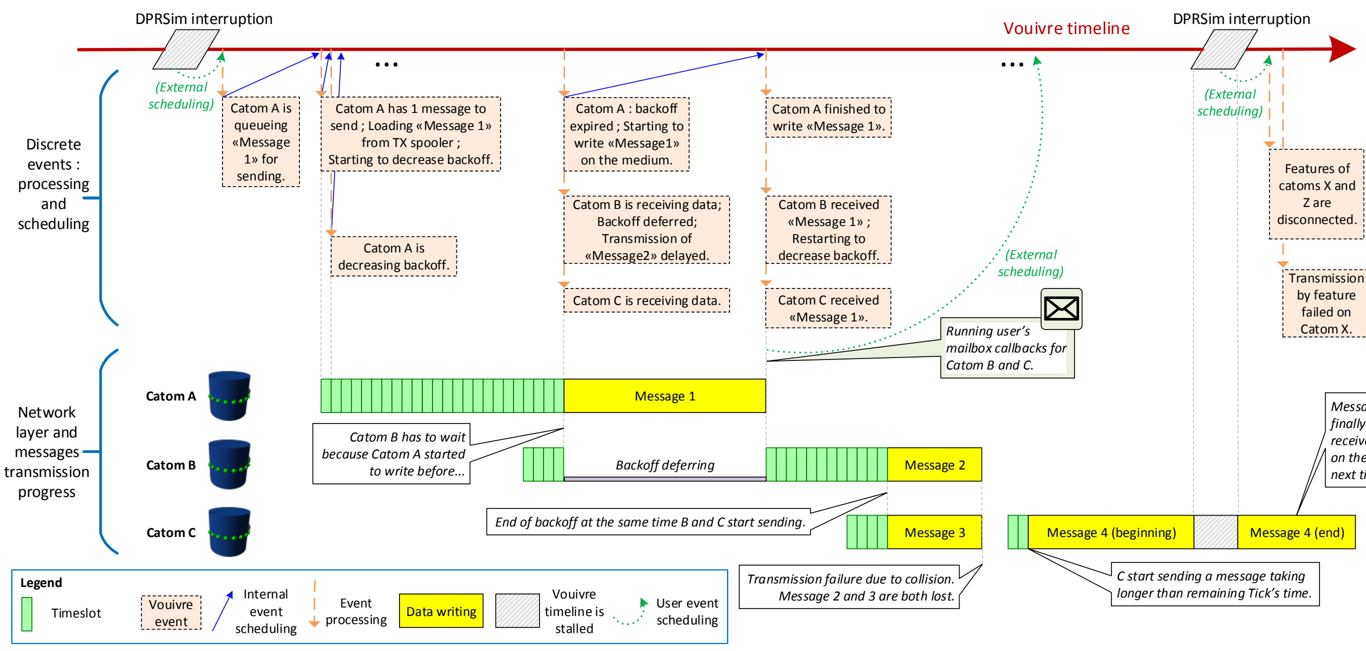
The lower part of the figure 7 shows an example of a network simulation in which three catoms named A, B and C are communicating together without radio coverage difficulties. We can see that timeslots are skipped one after another by network interfaces of catoms before to send messages. ”message 1” is sent and received correctly after is backoff period. The backoff for ”message 2” started before the transmission of ”message 1”, but is delayed (deferred) and then resumes after the transmission of ”message 1”. However, by coincidence, the end of its backoff coincides with the end of that of ”message 3”. Their transmissions start at the same time and a collision occur.

As previously explained, Vouivre internally use a discreet event simulator. This allows for an almost arbitrary precision, along with the ability to be very fast when no events have to be processed. Events represent actions such as starting to send or receive a message, but also more detailed mechanisms such as backoff-slots mentioned earlier.

Once instanciated, events are stored in a date ordered list, where they await their processing. Events can be created outside of the Vouivre simulator as shown by arrows labeled ”External scheduling” in figures 7 and 8. This is the usual way for applications to initiate communications, where their data are encapsulated into a ”Message” object which is then serialized and processed by Vouivre. On successful reception, the message is de-serialized and handled back to applications through the mailbox mechanism. A strong point of Vouivre is the ability for an event to trigger the scheduling of one or more other events. This allows for very tight causality chains to be created (such as very small messages triggering almost immediate responses and new messages from other catoms). This is illustrated in figures 7 and 8 with arrows labeled ”Internal scheduling”. Note that the duration of DPRSim ticks does not impact the chronology anymore.

The upper part of the figure 7 illustrates the scheduling and processing of events corresponding to the transmission of multiple messages. We can see ”Message 1” entering the TX spooler of a network interface. After that, a chain of backoff decrementation events takes place and the effective transmission start. At the same time the backoff decrementation of ”Message 2” is deferred and resumes later. On Figure 7 we also present how collisions are handle, with the backoff of ”message 3” unluckily ending at the exact same time as the one of ”message 2”. Vouivre detects this situation and fails the reception of those messages wherever their reception level is too similar according to the radio propagation model. Still during the same DPRSim tick starts the transmission of ”message 4”, which is too long to be completely transmitted during this tick. This is not a problem, as its transmission will correctly continue during the next tick. ”Message 4” will be received at the correct date, only dependent on the time the transmission started, the message size and the network bitrate.

As presented on Figures 7 and 8, three different timelines coexist in the modified version of DPRSim. The first one is the program running inside catoms (CodeModule). At each DPRSim tick, special


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 decrease backoff.
Catom A is decreasing backoff.
Catom A : backoff expired ; Starting to write «Message1» on the medium.
Catom B is receiving data; Backoff deferred; Transmission of «Message2» delayed.
Catom C is receiving data.
Catom A finished to write «Message 1».
Catom B received «Message 1» ; Restarting to decrease backoff.
Catom C received «Message 1».
Running user's mailbox callbacks for Catom B and C.
Features of catoms X and Z are disconnected.
Transmission by feature failed on Catom X.
Network layer and messages transmission progress
Catom A
Catom B
Catom C
Message 1
Catom B has to wait because Catom A started to write before...
Backoff deferring
Message 2
End of backoff at the same time B and C start sending.
Message 3
Message 4 (beginning)
Message 4 (end)
Message finally received on the next tick.
Transmission failure due to collision. Message 2 and 3 are both lost.
C start sending a message taking longer than remaining Tick's time.
Legend
Timeslot
Vouivre event
Internal event scheduling
Event processing
Data writing
Vouivre timeline is stalled
User event scheduling
Fig. 7: Discrete events and their effects on the network layer : communication examples

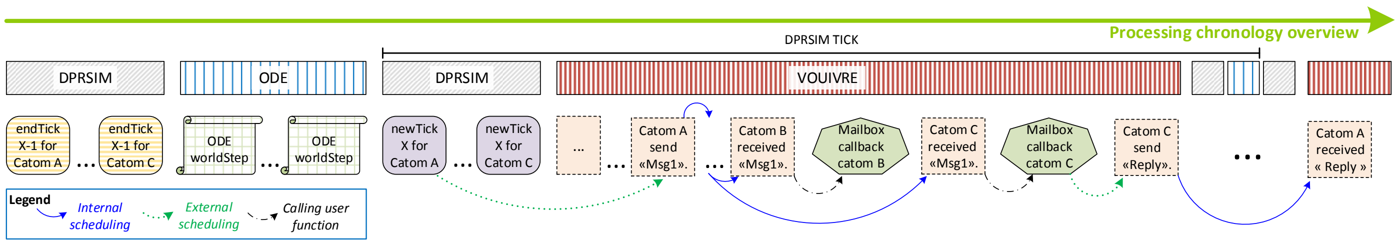


Fig. 8: Global processing overview and mailbox callbacks

function of the CodeModule are still executed one time for each catom (newTick() and endTick(), as on Figure 3).

The second domain is the physics, simulated by ODE. ODE internally operates iteratively. Its internal ”world-steps” have to be equal or smaller than DPRSim ticks. Their duration affect the behavior of the physic as the faster the motions, the smaller the ”world-steps” have to be in order to prevent unrealistic behaviors.

The last domain is the network, simulated through Vouivre. Vouivre makes use of Watt, Hertz and bytes/seconds units. As previously stated, as calls to Vouivre and ODE are included into DPRSim ticks, it is of utmost importance for them to simulate the exact same duration. A strong advantage of having multiple time-scales is that now, computation, communication and physic are independent from each other.

V. Test scenarios

In addition to wired contact connection already existing, wireless radio communications can bring benefits and increase networks abilities. This section highlights some situations and problems which can be optimized by addition of wireless communications. Please note that in order do simplify the development of illustrative applications, 2D catoms have been used, but the benefits would old in 3D.

A. *Converging walkers*

Situations exist where catoms are not able to communicate through wired connections. In fact it is quite possible to have isolated catoms or groups of catoms. Many applications would still require those catoms to regroup and assemble despite the fact that truly isolated catoms cannot even move by themselves. A full explanation of the model we developed would be out of the scope of this paper, but obtained results are still very relevant to this section. Using wireless communications and appropriate - multi-layered - control algorithm, it is indeed possible to allow the detection of isolated groups of catoms. They are then able to collectively decide for a convergence point (such as a barycenter for example) where they will regroup. Also, isolated catoms can announce their presence and small groups of other catoms can come to their ”rescue”. This is illustrated on Figure 9. The name ”walker” comes from the ability of a small group (3 or more catoms) to progressively move by iteratively adjusting the position of its constitutive elements.

B. *Multi hop flooding*

In many scenarii, because of their broadcast nature, wireless communications have a strong advantage in term of information propagation velocity. In the following scenario, an initial catom will send a 53 bytes packet to its neighbors. This packet will then be

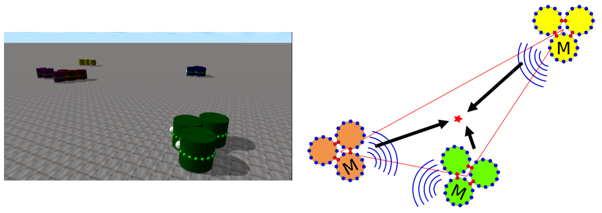


Fig. 9: All walkers send their position via wireless and converge to the on-the-fly computed barycenter

retransmitted hop by hop. To better grasp the behavior of the protocol, receiving this packet will trigger the coloring of the catom on the screenshots. The color itself will depend on the last re-transmitter. To prevent an everlasting flooding, the message contains a sequence number and can by re-transmitted only one time per catom.

Figure 10 shows 6 phases of retransmission on a square area covered by catoms. In this scenario, in order to further reduce the number of retransmissions, only black catoms (best seen on the upper-left screenshot) are allowed to retransmit. From phase to phase, we easily see how multiple catoms are reached at each retransmission. Please note that the number of catoms and the wireless range are voluntary kept small to enhance the readability of the pictures, but the benefits from wireless retransmission would only grow with increasing range and proper retransmission management.

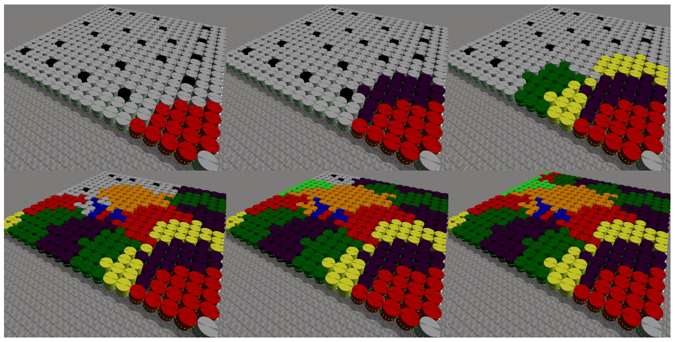


Fig. 10: Simulation 7 : A radio asynchronous flooding starting from the lower right corner

More irregular structures would also benefits from punctual wireless transmission capabilities. Much time can sometime be gained by directly crossing a gap instead of taking the long way around. This is illustrated with the scenario presented on Figure 11 and simulation results shown on Figure 12. Scenario T3 makes only use of wired neighbors communications. It takes the longest time from the initial transmitter until all catoms have received the message. Scenario T1 uses wired communication, but 4 catoms near the center of the cross also have wireless capabilities that enable the message to "jump" and propagate in the others branches. This "jump" is quite visible on Figure 12 around time=14s. After that, as copies of the message propagates in parallel in the branches, it takes less time to be received everywhere. Scenarii T2 and T4 are similar but also use some wireless transmitters on the branches themselves, thus speeding the propagation through the network.

We also conducted more experiments on the denser (20x20 catoms) macro-structure described on Figure 10. The parameters used

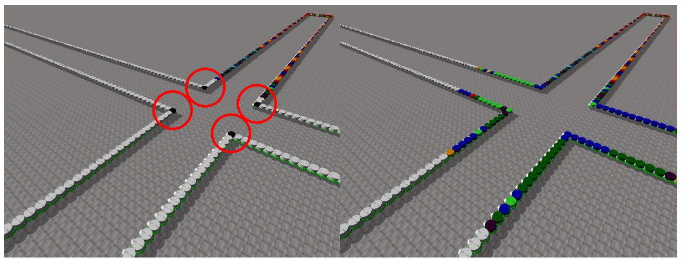


Fig. 11: Layout for simulations T1,T2,T3 and T4 with the "cross" macro-structure. The left picture shows the position of the radio relays for simulation T1.

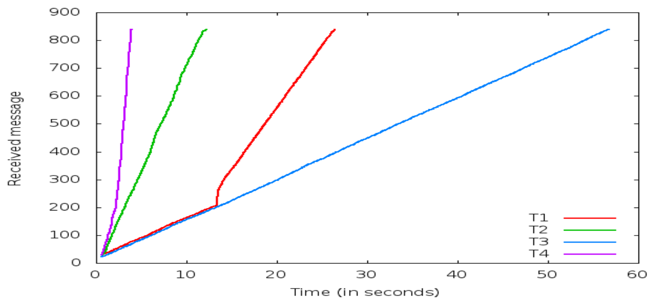


Fig. 12: Progression of the colorization wave the "cross" macro-structure

are presented in Figure 13 and the corresponding results on Figure 14. Scenario 6 with its much smaller backoff and increased troughput obviously sees the fastest propagation. Scenario 2 is wired only and takes the longer time to reach a full coverage of the network with the data packet. Scenario 3 has catoms with a larger radio coverage, propagation is faster, but collision rate is also higher and impact the time required to reach the full coverage. Scenarii 4 and 5 are similar, except for the density of radio transmitters. Propagation is initially faster in scenario 4 where every catom re-transmits, but then suffer from collisions; scenario 5 and its regularly spaced re-transmiters achieves the full coverage earlier. Scenario 7 behaves very well with its hand-picked transmitters (chosen in order to minimize collisions and maximize surface covered by each retransmission). Scenario 8 behaves very well also by using regularly spaced radio transmitters complemented by wired communications that mitigate the effect of collisions on the radio network.

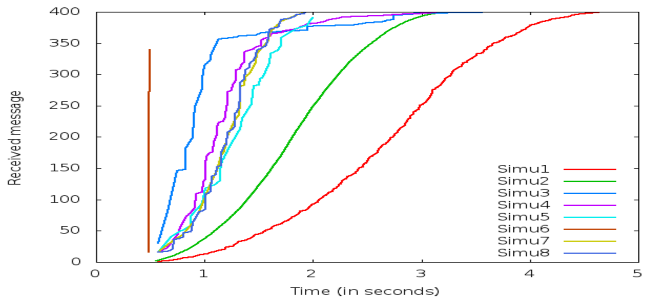


Fig. 14: Progression of the colorization wave in the time in the macro-structure "checkerboard 20*20"

Simulation identifier	1	2	3	4	5	6	7	8
Type of interface used	W	W	R	R	R	R	R	W+R
Radio transmitting power	-	-	2mW	1mW	1mW	1mW	1mW	1mW
Choice of relaying catoms	all	all	all	all	RG	RG	HP	all wired, RG radio
Backoff range (for radio)	-	-	30	30	30	30	30	30
Timeslot duration (for radio)	-	-	4ms	4ms	4ms	1.5μs	4ms	4ms
DataRate of wired interface	6 kbs	6 kbs	-	-	-	-	-	6 kbs
DataRate of radio interface	-	-	6 kbs	6 kbs	6 kbs	6 mbs	6 kbs	6 kbs

Fig. 13: Parameters used in simulations with the checkerboard structure. Legend: HP for hand-picked selection; RG means regular in grid; W means wired only; R means radio only; W+R for hybrid that means using both interfaces; kbs=kilo bits per second; mbs=mega bits per second

VI. Conclusion

We have presented the integration of a wireless simulator called Vouivre in DPRSim, a simulator for modular robots. The major challenge we faced was the difficulty to integrate three parts that have different timing requirement: physical (ODE), DPRSim and Vouivre. This has been solved by using and managing three timelines. Vouivre has been designed as a library and can therefore be integrated in others simulators. Furthermore, integrating it in DPRSim gave us an experience which can be reused. Experiments have shown that Vouivre is easy to use and that wireless integration in MEMS modular robots could be useful for various kind of applications either for extending the communication possibilities, with the distributed walker scenario, or by optimizing communication times, with the broadcast algorithm. Our next step will now be to integrate in Vouivre the nanowireless channel model developed in [25].

References

[1] J. Bourgeois and S. Goldstein, “Distributed intelligent mems: Progresses and perspectives,” in *ICT Innovations 2011*, ser. Advances in Intelligent and Soft Computing, L. Kocarev, Ed. Springer Berlin / Heidelberg, 2012, vol. 150, pp. 15–25.

[2] S. C. Goldstein, J. D. Campbell, and T. C. Mowry, “Programmable matter,” *IEEE Computer*, vol. 38, no. 6, pp. 99–101, June 2005.

[3] S. C. Goldstein, T. C. Mowry, J. D. Campbell, M. P. Ashley-Rollman, M. De Rosa, S. Funiak, J. F. Hoburg, M. E. Karagozler, B. Kirby, P. Lee, P. Pillai, J. R. Reid, D. D. Stancil, and M. P. Weller, “Beyond audio and video: Using claytronics to enable pario,” *AI Magazine*, vol. 30, no. 2, July 2009.

[4] M. E. Karagozler, A. Thaker, S. C. Goldstein, and 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.

[5] J. C. Pujol, J. M. Jornet, and J. Sol-Pareta, “Phlame: A physical layer aware mac protocol for electromagnetic nanonetworks,” in *Proc. of the 1st IEEE International Workshop on Molecular and Nano Scale Communication (MoNaCom)*, 2011.

[6] J. M. Jornet and 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.

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

[8] M. P. Ashley-Rollman, P. Pillai, and M. L. Goodstein, “Simulating multi-million-robot ensembles,” in *ICRA*, 2011, pp. 1006–1013.

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

[10] A. Varga and 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*, ser. Simutools ’08. ICST, 2008, pp. 60:1–60:10.

[11] D. M. Nicol, B. Premore, and A. Ogielski, “Using simulation to understand dynamic connectivity at the core of the internet,” in *Proceedings of UKSim 2003*, Cambridge University, England, April 2003.

[12] “http://www.opnet.com/products/modeler/.”

[13] “Qualnet simulator,” http://web.scalable-networks.com/content/qualnet.

[14] J. Kačer, “J-Sim – a Java-based tool for discrete simulations,” in *Proceedings of the 23rd International Autumn Colloquium ASIS-2001: Advanced Simulation of Systems*. Náměstí Msgre Šrámka 6, 70200 Ostrava, Czech Republic: MARQ, September 2001, pp. 135–141.

[15] V. Zykov, P. William, N. Lassabe, and H. Lipson, “Molecubes extended: Diversifying capabilities of open-source modular robotics,” in *IROS-2008 Self-Reconfigurable Robotics Workshop*, 2008.

[16] Y. Meng, Y. Zhang, and Y. Jin, “Autonomous self-reconfiguration of modular robots by evolving a hierarchical mechanochemical model,” *Computational Intelligence Magazine, IEEE*, vol. 6, no. 1, pp. 43 –54, feb. 2011.

[17] B. Gerkey, R. Vaughan, and A. Howard, “The player/stage project: Tools for multi-robot and distributed sensor systems,” in *Proceedings of the 11th international conference on advanced robotics*, vol. 1. Portugal, 2003, pp. 317–323.

[18] N. Koenig and A. Howard, “Design and use paradigms for gazebo, an open-source multi-robot simulator,” in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3. IEEE, 2004, pp. 2149–2154.

[19] D. Christensen, D. Brandt, K. Stoy, and U. Schultz, “A unified simulator for self-reconfigurable robots,” in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 2008, pp. 870–876.

[20] G. Echeverria, N. Lassabe, A. Degroote, and S. Lemaignan, “Modular open robots simulation engine: Morse,” in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, may 2011, pp. 46 –51.

[21] C. Pinciroli, V. Trianni, R. O’Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Caro, F. Ducatelle, M. Birattari, L. Gambardella, and M. Dorigo, “Argos: a modular, parallel, multi-engine simulator for multi-robot systems,” *Swarm Intelligence*, vol. 6, no. 4, pp. 271–295, 2012.

[22] M. Kudelski, M. Cinus, L. Gambardella, and G. Di Caro, “A framework for realistic simulation of networked multi-robot systems,” in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, 2012, pp. 5018–5025.