Wireless Sensor Networks and Satellite Simulation

P.-Y. Lucas[1](✉), Nguyen Huu Van Long[2],
Tuyen Phong Truong[2], and B. Pottier[1]

[1] Université de Bretagne Occidentale, Brest, France
pottier@univ-brest.fr
[2] Can Tho University, Can Tho, Vietnam

Abstract. Connecting wireless sensor networks (WSN) by the air becomes attractive due to advances in Satellites and Unmanned aerial vehicle (UAV). This work focus on specification and simulation of situations where several distant WSN have gateways visited periodically by a mobile on a static path. To develop and evaluate collection and control services, it is first needed to run system level simulation. This paper reports on producing automatically representations of complex topology where mobile cooperate with sensor fields with respect to timing constraints from both sides. Simulation programs are produced for graphic accelerators (GPU), and concurrent process architectures.

Keywords: Simulation · Satellite · Sensor network · Distributed algorithm

1 Introduction

Wireless Sensor Networks (WSN) are known to solve practical problems in emergency and environment monitoring. Most of WSN are deployed for managing aspects of *smart cities*: parking allocation, transportation, pollution, home services. They can group thousand of nodes in places where communication systems are abundant. They are of less common use in distant areas, such as shores, deserts, mountains, polar regions, due to the lack of energy and communication supports. In these cases mobile visits, including satellites, enable to collect data periodically, to data center, or to provide remote control on sensor fields.

Beside geostationary satellites, low earth orbit (LEO) satellites systems are operated for purposes such as observation, positioning, rescuing or commercial global communications. Well known LEO systems include manned orbiters and the International Space Station (ISS), geo-positioning system (GPS) variants, Iridium constellation, or Argos services. Very small satellites called *CubeSats* were initiated in USA, in 2003 [3]. They are more and more considered to build valuable experimentations for at least two reasons: energy budget and solution cost. The low altitude, from 160 to 2000 km, means low energy budget for launching and for communications. LEO are also associated to high travel speed, bound

P. Pillai et al. (Eds.): WiSATS 2015, LNICST 154, pp. 185–198, 2015.
DOI: 10.1007/978-3-319-25479-1_14

to short orbital periods from 1 to 2 h, and short distances suitable for remote sensing. The compactness, low cost, and standardization of devices, have motivated several international projects such as CubeSat, *QB50*, or *Outernet*.

1.1 Simulation Definitions

This paper presents preliminary *system investigations*, with the perspective to simulate access to *large sensor fields* grouping thousands of nodes distributed in several remote area. Unmanned mobiles, such as LEO satellites, are visiting these fields. They control remote data collection, distribute synthetic information to ground stations, and they control sensing operations.

Work Hypotheses. We take the hypothesis of sensor systems permanently sampling physical processes, communicating with neighbours periodically, using a mesh organization. This is common in 802.15 wireless solutions. Sensor fields group any number of sensors, anywhere, with any connectivity, but they are required to share a common abstract clocking system to schedule communications. Mobiles travelling on predictable paths, at predictable speed will visit the sensor fields, establishing communications with one or several gateways.

Objectives. They are to establish specification methods to describe sensor fields from geographic environments. From this, we can infer or tune radio ranges, deduce ground network topology. From mobile path, we can infer meeting schedules with the ground. This enable exploration and design of cooperative algorithms for remote control and data collection. Expected metrics are correctness taking account of ground and air relative speeds, latency, risk of failures, energy budget for communications.

Work Contents. Simulation requires high performance computing and enough flexibility to associate mobiles, sensor field and mobile interaction algorithms. Intermediate level models for these activities are produced with automatic translation of system organization for parallel execution as MIMD and SIMD programs. General Purpose Graphic Processing Units (GPGPU) have been found to be a solution to simulate the sensing activity, the network activities, and interactions with satellites.

According to this context, the paper will firstly describe the WSN-Satellite problem, then will describe the simulation flow applied to 3 collection algorithms.

1.2 Simulation Flow

The flow for simulation has 4 main steps shown Fig. 1.

1. Using geographic tools for specification of satellite path and sensor fields.
2. Building an abstract network and formal representation for the sensor fields. The abstract network hold information such as node names, communication ranges, channels, geographic positions.

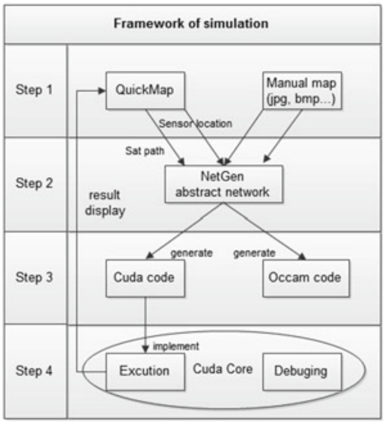

Fig. 1. Framework for simulation

3. Generating concurrent code in CUDA or Occam that represents network topology of sensor fields (no behaviours).
4. Implementing or reusing behavioural distributed algorithms for activities inside WSN, and with satellites. The result is a simulation program that will be executed on powerful platforms enabling step by step analysis.

The objective is to enable ambitious services for a promising domain [1].

2 Geographic Specifications

2.1 Orbiters and Radio Links

Two reference commercial satellite systems are Argos and Iridium. Argos data collection and location system focus on surveillance as well as protection of environment and wildlife, meanwhile, the Iridium network is a global satellite communication service for subscribers from government agencies and public citizen. Each system was established as a global satellite constellation of the low Earth orbiting (LEO) satellites flying at about 700–900 km above the Earth. A typical satellite's footprint is thousands of kilometers in diameter. Besides, both systems have already supported all three links namely up-link, down-link and cross-link to provide the high reliability of the communications network and to remain unaffected by natural disasters such as hurricanes, tsunamis and earthquakes, etc. The systems are often designed with L-band antennas to meet the requirement on high performances, low power supply, compactness, low cost.

Orbits. Due to the Earth's rotation, the swath shifts around the polar axis at each revolution, as shown Fig. 2. As a result, the overlap between successive swaths allows satellites enough time to visit ground stations in footprint vision for sending and receiving data several times per day. As a complement, ground infrastructure ensures 24/7 for controlling and monitoring on all components: transceivers, gateways, interconnections, and terminals. Even with these obvious advantages, the services are still expensive and closed.

Satellite Prediction. Several software packages allow to retrieve and interpret satellite path information from public repositories. As a reference example, GPredict is a public domain software with lot of operational capabilities based on the *prediction* of satellite paths: real time track characteristics, schedule table, footprint, communication establishment, control of antenna. Computation of positions and speed is based on keplerians elements of its orbit, these parameters being provided by public servers.

Miniature Systems. Now, unmanned aerial vehicles (UAVs), airships, balloons and small satellites are experimented for collecting and delivering data. In a common scenario, a CubeSat pico-satellite carries sensors for significant scientific research, and drifts along low attitude polar orbit resulting in easy monitoring the Earth's surface. The velocity of most CubeSat and the corresponding diameter of footprint are about 10 km/s and around 500 Km at attitude 100–200 km respectively. Initially, each CubeSat just performs individually, but then came projects for interconnections and to establish satellite constellations. This also aims to achieve better performance in severe conditions, e.g. magnetic storms, and natural catastrophes [9].

Radio Links. In previous years, the CubeSats have had two radios for different links VHF- 2 m band for up-link and UHF-70 cm band for down-link because of the limit of receiver load, power resource on board. Many digital modulations have been proposed to reduce the noise and to gain higher effective communication. CubeSat communication systems recently tends to perform on S-band with frequencies range from 2–4 GHz using smaller size of antennas in both air and ground segment.

2.2 Sensor Fields

The application deployments cover targets in wide distant area or behind obstacles (mountains, oceans).

A sensor field works like a hierarchical two-stage synchronous network. The first stage is for sampling physical processes, as example for environment monitoring. Then a local distributed decision algorithm take place using radio communications. Practical implementations use radio transceivers for communication standards such as IEEE 802.15, or Zigbee. Several frequencies are available that allow short (100 m) or medium range (10 km) communications between nodes. The first stage produces synthetic information to gateways that participate in the second stage. Several sinks can be embedded in a sensor field to feed visiting mobiles, and their activity can be coordinated. Gateways are expected to have energy support and hardware for receiving and transmitting to the satellite.

Field control implies serious observation of timings since communication can take place only when neighbours have schedule a rendez-vous. The normal status of a sensor is the sleep status awaiting for sensing and communication periods. A good image for this is the cycle for synchronous Time Division Multiple Access super-frames from IEEE 802.15.4.

2.3 Map Browsers and Satellite Tracks

Map browsers are now of common use, and it is an evidence that sensor distributions necessitate such tool to ease sensor field description for preliminary exploration. When interactions with mobile flights are considered, additional supports include mobile path description and radio link connectivities,

Maps are downloaded from servers (OpenStreetMap or GoogleMap) that supply tiles of raster georeferenced images. Map browsers are intuitive: the user moves the map with the mouse, by click-and-drag, and zoom level is changed by the mouse wheel. It is therefore possible to display different level of details. The maximum zoom level depends of the service resolution: OpenStreetMap offers 16 levels, while GoogleMap get down to 22 levels. The number of tiles t is a function of the zoom level z: $t = 2^{2z}$.

Map Projection and Quickmap Tool. The Mercator projection is used with the purpose to convert the round Earth with angular coordinates, longitude λ and latitude ϕ, into a flattened map with metric coordinates (x, y). Formula are given in [6] that allows to compute conversions, with applications to sensors and satellite representation on flat surfaces.

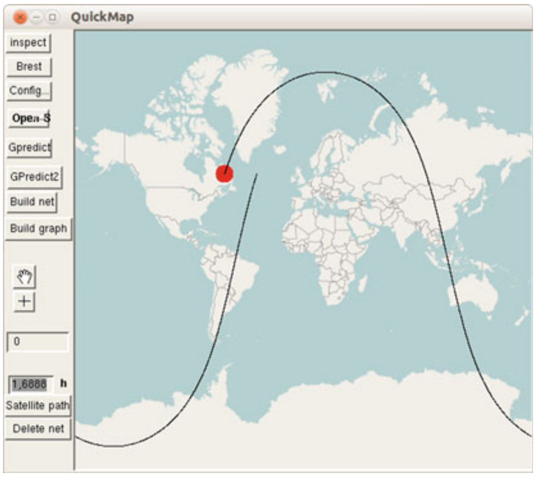

Fig. 2. QuickMap showing an LEO satellite path. This one-period satellite footprint was recorded during approximately 1 h and a half. The shift in horizontal position represents the earth rotation during the period.

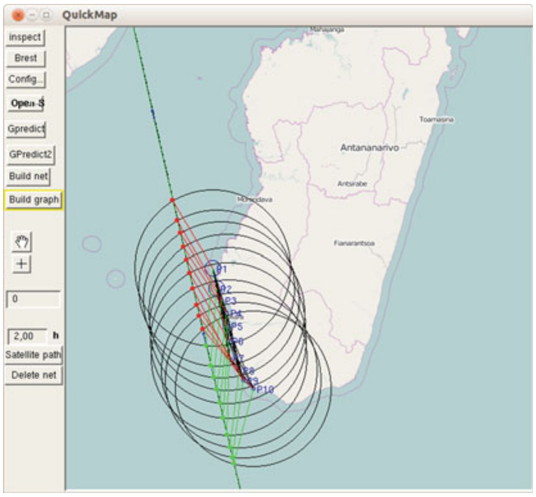

Fig. 3. The satellite path, and a sensor field. The green lines and the red lines represent respectively radio link establishment and deletion (Color figure online).

Quickmap, shown Figs. 2 and 3 was developped to support sensor layout, radio ranges computations, network topology presentation, as well as satellite tracks and further functionalities such as cellular system synthesis. Its main role is to facilitate sensor interactive location specification. Sensors positions are sent to a companion tool NetGen (step 2 from Fig. 1), that computes the network topology and feed back resulting drawings according to radio range.

Sensors and Satellite Tracks Relations. Geographical information, and time sequencement are the key points to consider these relations.

Practically, satellites have predictable track, that can be described by algorithms or by discrete sequences of events ((time, position) ...). To bind mobiles to sensor fields, it is necessary to share time references and geographic positions, For this reason, QuickMap was interfaced to GPredict, grouping mobile and sensor networks models.

Mobiles to fixed communications are computed using the mobile *discrete timed path*. For each point in the path, a set of reachable ground sensors is computed. For each new connexion, the ground network is completed by and air link, while the loss of a connexion produce a link destruction. A graphical view of a ground network and air radio links is shown Fig. 3.

3 Simulation : Architectures and Behaviours

3.1 Model for System Architecture and Distributed Behaviour

The final objective is defining distributed services, performances, costs and risks.

Several entities appeared at different level such as geographic distribution, interaction with physical processes, distributed behaviour, communications, mobility, and local sensing. The situation is similar to basic separation of concerns proposed and design methods for CAD tools, in particular disjoining organization (circuits) from behaviours (algorithms).

According to general principles in the domain of distributed algorithms [5], the design framework should propose supports for two orthogonal abstractions:

– **the system architecture** describe nodes and their communication links,
– **the behaviours** are local programs cooperaing by message exchanges

When applied to simulation, a general idea is to produce concurrent programs that separate almost completely the algorithmic and organization contributions, enabling to use and control huge number of nodes from distributed algorithms, whatever is the architecture topology.

Mobiles are considered as part of the system itself, having their own logic, as it is the case for satellites. They are moved by specific simulation threads, perhaps statically, perhaps dynamically. Clocking systems within WSN and time continuity of satellite moves help considerably to handle synchronization of ground and air activities.

3.2 Architecture Representations and Concurrent Programs

Network topology are expressed internally on a simple network model, as *named nodes* and *communication links*. Few attributes can basically be added, such as the physical location of a node, or the expected communication range.

A Network is therefore a large data structure grouping nodes and their associated links. Several separated graphs generally coexist in this structure, mobiles being nodes that follow static or dynamic trajectory under external control, eventually taken from the real world.

Given this central data structure produced by front end tools, translators allow to produce equivalent representation in terms of communicating processes either in Occam syntax and CUDA syntax. Occam uses micro threads and blocking channels ([8]). It is suitable for mono and multi-processors simulation of WSN (as shown in [4]), for distributed execution, and for execution on sensors, either in the form of virtual machines or native code. CUDA is a programming language associated with GPGPU. GPGPU are using the notion of *Kernels* executing in parallel a sequential procedure. Nodes behaviour can be represented inside these procedures, providing that the networks are acting synchronously [2].

While Occam represent communications by point to point blocking message sending, CUDA is operated by synchronized exchanges in GPGPU memory. These exchanges are executed by automaton based on a description of copy operations to and from neighbour buffers. The list of these operations is produced automatically by NetGen CUDA translator to represent and use the connectivity. The size of the communication program is adapted to the maximum fan-out inside the network (see [7] for technical details).

A nice effect of NetGen software approach is that the *node behaviours* can be specified, validated, and stored in the form of Occam or CUDA libraries. We have been able to simulate very large system concurrently, at the level of 1000 processes and more than 10000 processes respectively (see [2,4]).

Behaviours. Node behaviours group several activities for *sensing*, for making local decisions that will appear as a change in the node state, and for exchanging with neighbours. This also follow the more common way to describe distributed algorithms, and particularly synchronous, self timed behaviours [5]. In this model each node executes cycles for change of state (C_i), message out (M_i), and message in (N_i).

Our simulation program will execute similarly sensing and sleeping, communication phases, and making of decisions. There is only one stage for communications since they are executed by broadcast, each node sending at its turn.

Such behaviours are strongly bound to the synchronous model, the handling of communications on links being executed by procedure calls. The programming pattern is a loop grouping communications, buffer analysis and sense data analysis to obtain local change of state, then starting of sensing/sleeping activities.

Occam simulations do not explicitly make use of time, but are only sequenced by inter-process synchronizations. By removing the delays, this enable respect of causalities inside networks, and asynchronism between separated networks. Small set of nodes are running faster than large ones because the diameters and communication work load are fairly smaller. The simulation progresses are observed by sending node status to a trace process that is embedded automatically in the architecture. Graphic presentation can be observed by filtering the trace flow to produce annotations.

4 Mobile to WSN Collection Algorithms

Three algorithms for satellite and sensor fields cooperation have been experimented.

- **anti**-cipated. In the first case, the proximity of the arrival of a satellite is known. Therefore, nodes anticipate the interaction, preparing data to be sent. The nature of the computation can be static, or passed from the previous satellite visit.

  We will not detail this algorithm which is simply implemented from an downward and upward *breadth first search* algorithm started from the expected visited gateway.
- **trans**-action. In the second case, The satellite send a command at its first contact with the sensor field, which propagate the command and execute distributed computation dynamically. This time, the command must flow inside the sensor field with results sent back to the satellite. This takes at least two diameters period to execute, and the satellite speed becomes a larger constraint.

– **flow**. In the third case, the satellite send a command which is processed on the fly in the sensor network, with the results forwarded to an exit node where the satellite can receive it.

Algorithms have been initially coded into Occam. The mobile is not present, being represented by WSN gateways. Many synchronous loops have been executed for random networks of different size $10, 50, 100, 200, 500$. This number allows to compare the elapsed time in the network with the satellite visit delay. This comparison is critical for verification of system correctness.

As these algorithms are expressed following the synchronous model, the number of loops executed to achieve complete network computation is at least *diameter* steps. The following sections describes algorithms **trans** and **flow** in terms of Occam automaton.

4.1 Transaction

The Problem. Mobile arrives over a sensor field and obtains a connection with one gateway. It sends a command to the gateway asking the execution of a global computation implying a visit of every reachable node. After a moment, it is expected that a result message will be sent back by the gateway to the mobile. Each node contributes to the computation in a commutative way, as examples: computation of the field bounding box, global status, maximum.

Algorithm Informal Description. A breadth first search (BFS) is built dynamically using forward (downward a tree) and then backward (upward a tree) propagation to implement a global computation.

Downward Visit. During this stage, a root node sends a search message carrying the command. Each other node will receive the command at some step of the synchronous loop. At the first time and only at this time, one parent (and only one) is elected and informed. Each other neighbour will receive "search" at the next step. After diameter step, all the node in network have received and propagated successfully the search message. In addition, the relationship between nodes: parent index and list of children is established for next backward algorithm to propagate the result.

Upward Back Propagation. The upward operation starts in nodes after search propagation, with no parent notification from any outgoing link. At this time, a local result is produced and sent to the parent node. As soon as the parent node receive the partial results from children, it computes its local contribution and waits for next result It will send the its final contribution upward as it collected and computed all the results from children. By this way, after diameter step in computing, the Root will receive and update the result to make a global result before sending it to mobile.

Obviously, combining two algorithms, upward and downward, the total time Root node will receive the global computation result is at most $diameter \times 2 + 1$ steps. *As consequence, connection between mobile and gateway must be active over this delay.*

Protocol and Messages. Messages in the abstract algorithm are $null, request, parent, result$. For Occam implementation, a protocol construct associates data to typed messages. With a customization for bounding box computation it comes: *BBoxRequest BBoxParent BBoxResult SendNull.*

State Definition. Nodes retain their local state in a set of variables: index or identities for possible parent and children, Boolean to recall previous visits, etc... The sensing status is figured by the geographic position in an initially unknown bounding box rectangle. In Occam, the state is retained in a set of variables for neighborhood description and automaton stages.

Downward Stage Automaton. The local automaton state is kept in a variable `state`, with possible values:

StateInit, StateSendRequest, StateReceiveParent, StateEnd, $initiallystate = StateInit$.

Upward Stage Automaton. the possible values are:

StateReceiveResult, StateSendResult, StateIdle, $initiallystate = StateReceive Result$. Following the current state, the local automaton fill message buffers to the neighbours according to the computation status. In the initial state, nodes are waiting for $BBRequest$. Upon reception, output buffer are filled with BBParent, or BBRequest messages. To illustrate how the algorithm works in communication rounds, an example network with 50 nodes is presented Fig. 4 The root node R1 of this tree has id = 9. The red link is the downward part and the blue link is the upward part. The satellite sends the command signal at R1 and receive back the global result at R1 also.

Obviously, this algorithm proposed a better solution than the first **anti** algorithm in receiving command from a gateway, executing the command and sending back the result for satellite. However, it has the critical drawback: the race between forward-backward computing and movement of satellite.

4.2 Flow

The Problem. In the previous algorithm, a critical problem is time constraint. A better solution, is to use a traversal propagation in computation and transaction. The strategy is to merge two BFS to receive command and propagate result to a sink. As result, we have more time related to satellite crossing sensor field.

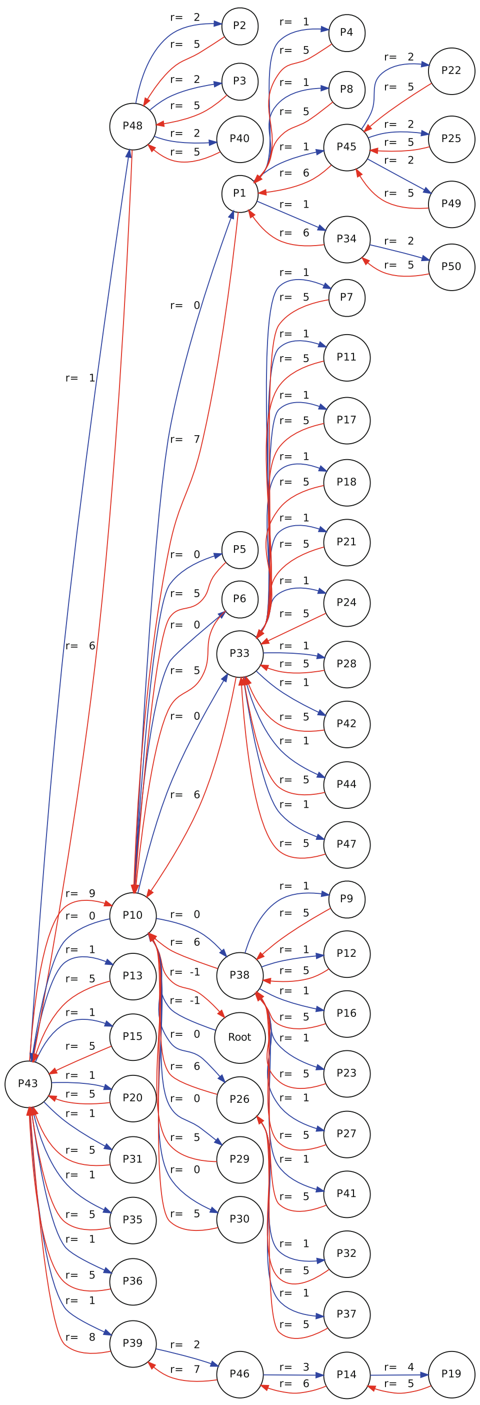

Fig. 4. tran: Data transaction and rounds on a 50 node network

Algorithm Informal Description. Thus two BFS trees are built inside the network. According to the BFS trees, the first Root is elected as the first gateway to receive the command signal of satellite when it passes over the node field. In contrast, the second Root is elected as the second gateway to send back the result of computation just before satellite leaves the node field. We also consider that we have already built both BFS similarly as for second algorithm before satellite enters the field. Therefore, the structures of protocol, messages are similar to **tran** algorithm. The maximum number of steps is $diameter \times 4$.

State Definition. The state set is similar to **tran** with some changes. Instead of forwarding result to Root of first tree, we forward the partial result to Root of second tree. Therefore, we need a other collections for list of children and the index for parent node in second tree. In addition, the local automaton state is kept in a variable `state`, with possible values:

$StateWaitForSignal, StateRequestComputing, StateReceiveResult,$
$StateSendResult, StateIdle, initiallystate = StateInit.$

Message Production and Automaton Management for Downward and Upward Propagation. According to the state, the local automaton will fill message buffers to the neighbours according to the following patterns. In the initial state, nodes are waiting for *BBRequest*. Upon reception, output buffer are filled with BBRequest messages. In special case, the state change can be optimized when a node is a leaf for both tree. The leaf node can send back the result in next round by change from StateWaitForSignal to StateSendResult.

As a result, the third algorithm appears as a best solution for transaction protocol for satellite and sensor network. This algorithm is adaptable and reliable for a real deployment. In next section, we analysis these presented algorithms by testing the performance in execution time based on different scenarios. In addition, we also prove why the third algorithm is adaptable solution to deploy for transaction protocol.

4.3 Simulation Performance

Performance for distributed computations depend on the network topology and maximum number of channel in/out for nodes. These factors can be controlled inside NetGen and QuickMap by checking parameters such as Network zone definition, Node range and Number of node in network. Several scenarios were used in Occam simulation to obtain random distribution according to these parameters. The result were checked to be correct. As an assessment, (Fig. 5) displays performance with a fixed range of node of 800 meters with a varying number of random nodes covering an instrumented region.

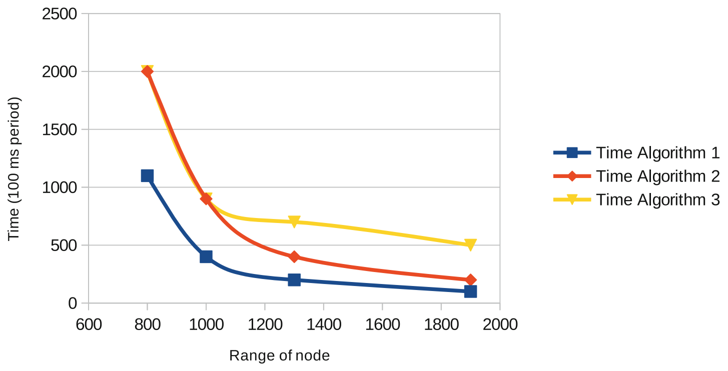

Fig. 5. Test results in time execution

5 Environment, Results and Perspectives

Simulations can be run as Occam or CUDA programs. CUDA code is distributed into few files related to the CPU and GPGPU behaviour description. One file is automatic generation from NetGen and includes the communication description table. The development platform is a variant of Smalltalk supporting dynamic generation and use of shared libraries. It enable to manipulate network structure stored on GPGPU to reflect changes such as mobile to field connectivity.

We have run simulation on Linux machines with NVIDIA graphic cards (GTX480, GFX680) providing respectively, 500 and 1500 processing elements. Both machines are equipped with Intel(R) i7 processors. As for Occam, random distributions over rectangular surface were generated to reflect deployments over wide area such as deserts, polar regions, countries or oceans.

For each set, the distributed simulation begins by producing a number of connected sensors, representing ground collective computations. Experimental behaviour is to elect leaders thus providing the number of isolated sub-networks. By observing the maximum fan-out, the simulation provide an additional information on the credibility of the network in terms of communication load (balance between sensor density and communication range) (Table 1).

Finally the execution time provides an idea on what can happen in long simulations (several orbits over the earth). Here the satellite path is controlled from Gpredict runs with a period of 1 s, during 15 min.

The outline of this simulation framework reveal the interest of two major components. **QuickMap** handles maps and navigation. It also allows specification of sensors or gateways over geographic presentations. **NetGen** handles network models, code generation, and control of simulation including the mobile moves. NetGen also send back annotations to QuickMap, and notably network and mobile communication graphic display.

System simulations are activated by exchange of messages between nodes and mobiles. The paper has provided a simple example with the presentation of a

Table 1. Results on GFX680

Number sensors	100	200	400	600
Real participants	91	186	331	434
Number of networks	17	19	69	123
Max fanout	6	7	6	7
Execution time (s)	52.8619	143.016	467.402	688.439

synchronous model Bounding Box computation. A number of distributed algorithm have been developed and tested over the Occam language giving confidence in the possibility to produce ambitious toolbox for Satellite to WSN problem. An evidence is the need to design protocols for uploading and downloading data appearing in gateways with systems that includes ground stations and final users. These simulation tools help considerably in understanding geographic deployment properties, time constraints, and even communication energy budgets.

The footprint of a LEO satellite is defined by the altitude of its orbit. Its velocity is in inverse proportion of the altitude. As a result, access windows time depends on both satellite's altitude and evaluation angle of ground station. Selecting proper radio frequency and protocols, high gain antenna and mechanic structure for satellite are also key factors in satellite communications. These factors are to be considered to design direct links between sensors fields and small satellites, with the capability of ground networks to elaborate synthetic data collectively. Ground speed is bound to the frequency of sensor networks, and number of hops, if any.

References

1. Celandroni, N., et al.: A survey of architectures and scenarios in satellite-based wireless sensor networks: system design aspects. Int. J. Satell. Commun. Netw. **31**, 1–38 (2013)
2. Dutta, H., Failler, T., Melot, N., Pottier, B., Stinckwich, S.: An execution flow for dynamic concurrent systems: simulation of WSN on a Smalltalk/CUDA environment. In: DYROS/SIMPAR 2010, Darmstadt (2010)
3. Heidt, H., Puig-Suari, J., Moore, A., Nakasuka, S., Twiggs, R.: CubeSat: a new generation of picosatellite for education and industry low-cost space experimentation. In: Proceedings of the AIAA/USU Conference on Small Satellites (2000)
4. Iqbal, A., Pottier, B.: Meta-simulation of large WSN on multi-core computers. In: DEVS 2010, in SpringSim SCS Conference, Orlando, USA (2010)
5. Lynch, N.: Distributed Algorithms. Morgan Kaufmann, San Mateo (1996)
6. Pridal, K.P.: http://www.klokan.cz/projects/gdal2tiles/
7. Thibault, F.: (UBO): NetGen : un générateur de code pour CUDA : principes, implémentation et performances. Technical report (2010)
8. Welch, P.H., Barnes, F.R.M.: Communicating mobile processes: introducing occam-pi. In: Abdallah, A.E., Jones, C.B., Sanders, J.W. (eds.) Communicating Sequential Processes. LNCS, vol. 3525, pp. 175–210. Springer, Heidelberg (2005)
9. Wright, D., Grego, L., Gronlund, L.: The Physics of Space Security. American Academy of Arts & Sciences, Cambridge (2005)