Simulation and compiler support for communication and mobility for environment sensing

Tuyen Phong Truong

THESE DE DOCTORAT DE

L'UNIVERSITE
DE BRETAGNE OCCIDENTALE
COMUE UNIVERSITE BRETAGNE LOIRE

ECOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies
de l'Information et de la Communication
Spécialité : *Informatique*

Par
Tuyen Phong TRUONG

Simulation and Compiler Support for Communication and Mobility for Environment Sensing

Thèse présentée et soutenue à Université de Bretagne Occidentale, le 29 Août 2018
Unité de recherche : Lab-STICC UMR CNRS 6285
Thèse N° :

Rapporteurs avant soutenance :

Congduc PHAM	Professeur, Université de Pau et Pays de l'Adour
Tanguy RISSET	Professeur, INSA-Lyon

Composition du Jury :

Vincent RODIN	Professeur, Université de Bretagne Occidentale / Président du jury
Congduc PHAM	Professeur, Université de Pau et Pays de l'Adour / Rapporteur
Tanguy RISSET	Professeur, INSA-Lyon / Rapporteur
Simona NICULESCU	Maître de conférences - HDR, Université de Bretagne Occidentale / Examinateur
Directeur de thèse	
Bernard POTTIER	Professeur, Université de Bretagne Occidentale
Co-directeur de thèse	
Hiep Xuan HUYNH	Professeur - HDR, Université de Can Tho

Acknowledgement

I would like to express my sincere gratitude to my supervisors, Prof. Bernard POTTIER, from the University of Brest (UBO), and Prof. Hiep Xuan HUYNH, from Can Tho University (CTU) Vietnam, for giving me the opportunity to work on this topic and their excellent supervision. Their invaluable advice and great supports inspired me to make the right decisions at most of the important moments. I am glad to have a chance to work with them.

Special thanks go to Prof. Vincent RODIN (UBO) who has encouraged my elaboration participating in work validation and gracefully accepted to chair the defense committee. I would like to thank Prof. Congduc PHAM, from Université de Pau et Pays de L'Adour (UPPA), and Prof. Tanguy RISSET, from Institut National des Sciences Appliquées de Lyon (INSA-Lyon), who accepted to review the thesis document. I would also like to thank Dr. Simona NICULESCU (UBO) for being a distinguished member of the defense committee with her scientific development in relations between information system and geography.

I greatly appreciate Madame Michele Kerleroux and staff officers at École Doctorale SICMA MathSTIC and Lab-STICC CNRS, UMR 6285, supporting my activities during the study.

I am thankful to my colleagues at Department of Electronics and Telecommunication Engineering, College of Engineering Technology, Can Tho University and in STIC Asia Modeling for Environment and Simulation (SAMES) project, who encouraged me to achieve complete investigations. Many thanks to all my friends and Ph.D. companions for their discussions and fruitful exchanges.

Last but certainly not least, I am extremely thankful to my family for their love and unconditional support throughout the years.

Brest, France, August 29th, 2018
Tuyen Phong TRUONG

Contents

Acknowledgement **3**

Foreword **11**

Chapter 1. Introduction **13**

1.1 State of the art for consideration components 14
- 1.1.1 Communication in wireless sensor network 14
- 1.1.2 Radio signal coverage 17
- 1.1.3 Cellular automata 18
- 1.1.4 Low Earth Orbiting satellite for remote sensing applications 19
- 1.1.5 Massively parallel computation 20

1.2 Contributions in sensing simulations 21
- 1.2.1 Radio coverage prediction 21
- 1.2.2 Physical simulations 21

1.3 Contributions in sensing activity developments 22
- 1.3.1 Mapping application on sensing systems 22
- 1.3.2 Modeling and optimization of mobility 23

1.4 Outline of this thesis 24

Chapter 2. Cellular Automata and Terrain Complexity **25**

2.1 Terrain Complexity Analysis 26
- 2.1.1 Geographical Information System and Digital Elevation Model 26
- 2.1.2 Terrain Ruggedness Index 27
- 2.1.3 Topographic Position Index 28
- 2.1.4 Accuracy of terrain complexity analysis 29
- 2.1.5 Designing for long distance radio coverage 30
- 2.1.6 Observing physical phenomena 33

2.2 Mapping geographic space into cell systems 35
- 2.2.1 Geographic map and user interfaces 35
- 2.2.2 Definition of cells 37
- 2.2.3 Cellular automata principles 38
- 2.2.4 Cellular automata parallel execution models 39
- 2.2.5 Cellular systems work flow organization 40

Chapter 3. Distributed Algorithms on Cellular System 45
3.1 Cellular system 45
3.1.1 Cellular system topologies 46
3.1.2 Algorithmic models 50
3.1.3 Mapping algorithmic models on different architectures 51
3.2 Flash flooding simulation on a complex terrain 52
3.2.1 Managing space 52
3.2.2 Transition rule 55
3.2.3 Algorithm of rainwater distribution 57
3.3 Case Studies 57
3.3.1 Fist case study in Guadeloupe, France 57
3.3.2 Second case study in Morlaix, France 59
3.3.3 Conclusion 59

Chapter 4. Signal Propagation and Communication Modeling 67
4.1 Semtech's LoRa technology: a brief overview 67
4.1.1 Principles and issues 68
4.1.2 LoRa: bandwidth 69
4.1.3 LoRa: Coding rate 70
4.1.4 LoRa: Spreading factor 71
4.2 Parallel algorithms for cellular long range coverage computations 71
4.2.1 General idea 72
4.2.2 Vertical model 72
4.2.3 Horizontal model and Directed Breadth-First Search 73
4.3 Radio signal propagation on complex terrains 74
4.3.1 Free space path loss model 75
4.3.2 Single knife-edge diffraction model 76
4.3.3 Okumura-Hata model 78
4.4 Four case studies on LoRa coverage prediction 78
4.4.1 Case study 1: Pont de Plougastel 79
4.4.2 Case study 2: Place Albert 1^{er} 80
4.4.3 Case study 3: Roc'h Trevezel 82
4.4.4 Case study 4: Brest harbor 83
4.5 Revendications 85
4.5.1 Multi-scale data space 85
4.5.2 Latency in long-range communications based on LoRa technology . . . 88
4.5.3 Quality of communications in LoRa networks 89

Chapter 5. Parallel Algorithms for Simulations 91
5.1 Breadth-First Search and Directed Breadth-First Search 91
5.1.1 Basic Breadth-First Search 91
5.2 Directed Breadth-First Search 94

5.2.1 Evaluation the accuracy of LoS computation 100
5.2.2 Influence of multi-scale data on simulation accuracy 100
5.2.3 Comparison of accuracy, performance between BFS and DBFS 104
5.3 Communication coverage optimization 105
5.3.1 Coverage optimization algorithms 106
5.3.2 Performance comparison of coverage optimization algorithms 109

Chapter 6. A Development Method for Sensing Systems 111
6.1 Organization structure of distributed sensing systems 111
6.1.1 Top-down approach for development distributed sensing systems . . . 112
6.1.2 A complete process oriented system organization 114
6.1.3 Programming considerations 114
6.1.4 Details on: radio communications 116
6.1.5 Details on: services 117
6.2 Code organization 117
6.3 Round-robin scheduling in LoRa star networks 118
6.4 Validation and Revendications 120

Chapter 7. Modeling and Optimization of Mobile Connections of Dynamic Sensor Field 123
7.1 BT-DYNSEN 124
7.1.1 BT-DYNSEN model of DSF 124
7.1.2 BT-DYNSEN model of LEO satellite connection 125
7.1.3 Computation of connection time 126
7.1.4 Definition the operation modes of DSF 128
7.1.5 Constraint 128
7.2 Optimization method 129
7.2.1 Verification of the connectivity of DSF 129
7.2.2 Determination the sensorsets corresponding to each sub-satellite point 129
7.2.3 Selecting the gateways of DSF 130
7.2.4 Finding shortest path for data dissemination 131
7.2.5 Parallelization of RNN algorithm to determine sensorsets 133
7.2.6 Parallelization of search algorithm to find the shortest path for data dissemination 133
7.3 BT-DYNSEN implementation 134
7.3.1 Block diagram of DYNSEN tool 134
7.3.2 Data normalization 134
7.3.3 Input data and Satellite communication model 135
7.3.4 Connection time optimization 136
7.4 Experiments 136
7.4.1 Experiment 1: Verification of network connectivity 136
7.4.2 Experiment 2: Determination of the sensorsets 137
7.4.3 Experiment 3: Selecting gateways 137

7.4.4 Experiment 4: Routing optimization for DFS 138
7.4.5 Experiment 5: Parallel RNN algorithm 138
7.4.6 Experiment 6: Parallel Dijkstra algorithm 139
7.5 Conclusion 140

Chapter 8. Conclusion and Perspectives **141**
8.1 Conclusion 141
8.2 Perspectives 142

Bibliography **145**

Appendix A. High Level Architecture **163**

Appendix B. Computing radio coverages: experiments on archipelagoes **167**
B.1 Practical effects 169
B.1.1 Coverage exploration 170
B.2 A summary of tool flow to predict communication coverage 171

Appendix C. Preliminary Results on Scheduling Algorithms of LoRa Network **175**
C.1 Basic algorithm 175
C.2 A study case of scheduling LoRa networks 177

Appendix D. Communication boundary of WSN: an implementation of parallel 2D convex hull algorithm **181**

Appendix E. System tools presentation **183**
E.0.1 Boards and components 183
E.0.2 Software interactions 184

Appendix F. Software tools presentation **189**
F.0.1 Graphical front-ends: Quickmap and Pickcell 190
F.0.2 Core tools 190
F.0.3 Advantage tools 191

Appendix G. Froggy LoRa library **195**
G.1 SPI library 195
G.2 LoRa library 196
G.3 Network library 196

Appendix H. inAir LoRa library **199**
H.0.1 spidriver.module 199
H.0.2 spilib.module 199
H.0.3 inaircad.module 200

H.0.4 networkcad.module 200

Appendix I. A LoRa star network experiment: package structure and occam-pi programs 203

I.1 Package structure for LoRa communications 203

I.2 Occam programs of a sink nodes and sensor nodes 204

Appendix J. Experiments on reconfigurable hardware for distributed remote sensing systems and LoRa modulation features 209

J.1 Dynamic configuration 209

J.1.1 Reconfigurable hardware platform in wireless sensor networks 209

J.2 LoRa spectrum observation with FUNCube Dongle Pro+ 212

Publications 215

Foreword

This thesis was developed from 31/08/2015 to 29/08/2018 in a cooperation between several partners from a *STIC-Asia program* project, called SAMES, mainly in Université de Bretagne Occidentale, hosting the work, and Can Tho University. The thesis was funded by Vietnam Ministry of Education and Training (MOET) with the support from the French Embassy. The initially chosen topic is *"Simulation and Compiler Support for Communication and Mobility for Environment Sensing"*.

During Ph.D. study period, I experienced in many concepts: concurrent and parallel programming, wireless sensor networks, virtual machines for embedded systems, and LoRa radio protocol. This was very motivating experiences above my initial background in computer engineering.

During the thesis we published several articles listed in *Publications* section, culminating with an in-depth article about the relation between distributed sensing and physical processes to be observed. This article was published in Sensors journal with the title *"Cellular Simulation for Distribute Sensing over Complex Terrains"* [1].

Chapter 2, 3 and 4 expand this article, giving more details on cellular systems, parallel and distributed algorithms for physical process simulations and radio signal propagation simulation.

Chapter 5 describes in detail parallel algorithms operating on cellular systems [2, 3], implemented on concurrent process system executed on multi-cores and GPU accelerators, following a synchronous messaging model.

Chapter 6 reports on mapping flow suitable to support distributed applications on wireless networks that have been explored at low levels including virtual machines and reconfigurable platform, briefly described in Appendix I and Appendix J. This work was validated by laboratory's developments still needing a platform for testing actual large-scale applications.

The case of a mobile vehicle controlling sensor fields is reported in Chapter 7 [4, 5, 6, 7]. This work was achieved at the beginning of the thesis, both in Vietnam and Brest.

We have provided appendices for insight into developments, experiments and work coverage.

A prominent point during Ph.D. studies is the central role of computer science with relation existing between specification languages, execution supports such as parallel and distributed architectures, algorithm design, and final applications.

1 Definition for Environmental Monitoring

There is an increasing interest in environmental monitoring due to the serious impacts of climate change on socio-economic factors and human health. In these applications, radio links are the key technology allowing sensors to connect into networks that can operate on various topologies and ranges. In this thesis we propose a framework for easing exploration of radio link possibilities, taking into account geographical topologies. Massive parallel execution on GPUs and cellular method were utilized in order to compute radio propagation and estimate the strength of received power. Several physical simulations could be federated for evaluating their consistency related to the described phenomena areas such as shores or islands, hills and valleys oppose practical difficulties to the design of radio sensor systems by blocking or perturbing communications.

In the context of sensor applications, *coverage* define where information is accessible, whatever they are: radio signal, biological, or physical influences. Therefore *the most precise coverage computation* is essential for monitoring efficiency and this can take various forms, depending on application fields. This is a specific situation distinct from general purpose wireless communication networks, such as cellular phone systems, because the application objectives are limited and well defined, with secured application behavior.

This thesis provides an analysis of relations with spatial geography topology (Section 2.1) and then exposes principles for radio link capabilities (Section 4), with focus on *LoRa*. By modeling geographic space into cellular systems, it becomes possible to represent many physical phenomena such as ocean or river effects, rain effects, and flooding, pollution, etc. Section 3.2 presents the example of a simplified rain flooding simulation, exposing the interest of these studies for sensor network design. Using the same principle, it is also possible to model the behavior of radio signal propagation in relation with effective geography. The algorithm is given section 4.2, which is based on the ability of signals coming from one point to reach another point with *line-of-sight* (LoS) characteristic. Section 4.2 explains how complete radio coverage can be obtained by combining together several cover computations.

Simulation of the control and observation network, in relation to expected physical activities, lead to optimized monitoring systems that can be investigated at Computer-Aided Design (CAD) tools level to prepare new or urgent deployments. In the case of long ranges, there is a direct impact due to predictable latency, thus system performance. Appendix C details latency optimization algorithms allowing to select base station location according to coverage issues, and overall system performance.

Finding communication ranges in presence of obstacles is known to be a compute-intensive task, with 3D complexity. The cellular representation allows developing efficient parallel algorithms that mimic physical behavior. Massive parallel execution of cellular systems is produced using code generators yet presented in [8]. Computations are achieved on Graphics Processing Units (GPUs) producing performances in the real-time order. Furthermore, many such simulations can be composed in frameworks such as *High Level Architecture* (HLA) [9] as shown in [10].

Appendix E presents the tool flow that allows selecting arbitrary study zones from a specific map browser called QuickMap, then a cellular system front-end called PickCell. The example of rainwater flooding (Section 3.2) has used the programming aspects of the flow. Simulated and actual coverage computations were applied and tested on a variety of significant areas in Brittany, France (shores, town, country) as reported in the appendices. Appendix F and F present system tools developed for practical investigations validating the CAD approach in this work.

1.1 State of the art for consideration components

1.1.1 Communication in wireless sensor network

Sensors could be found in a lot of applications for measuring distributed values, in many domains such as reducing traffic by monitoring available parking lots, measuring city pollution, measuring the cycle of water. Radio links are the core technology allowing sensors to connect into networks that can operate on range from meters to kilometers. In the design of sensor networks, the major question is not the network itself, but the objective of observation and control. It means that design methods should expose the physical object to the observation system for validation or optimization.

Nowadays sensor networks are used worldwide in the automatic surveillance of environmental systems. Because of its scalability and cost-effectiveness, wireless connections are dominating wired links in actual deployments, especially for complex geographic areas. As shown in Figure 1.1 the number of smart devices will be connected has grown up to 5 billion from 2015 to 2017. It will be expected to increase exponentially reach 75 billion by 2025, and then surge to 125 billion by 2030 [11, 12].

Currently, there are several competing technologies with various techniques (see Figure 1.2) to achieve long-range, low-power operation, and high scalability. On a technical side, these technologies must be compromised between conflict goals such as communication

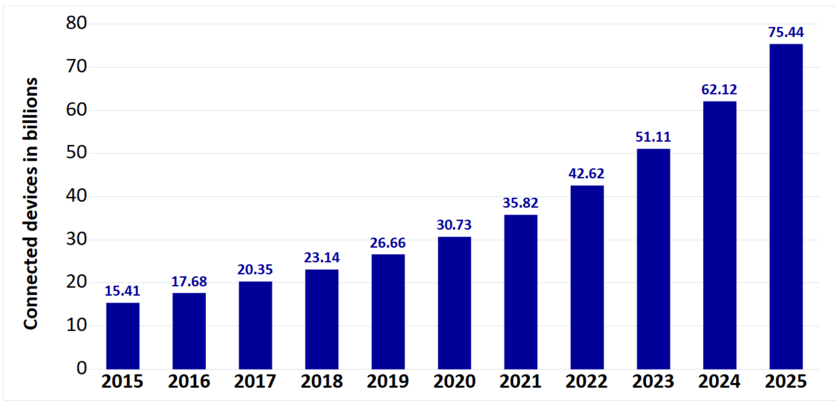


Figure 1.1 – Insight's prediction about the growth up of smart devices has been installed worldwide by 2025. Number of connected devices is expected to increase around three times from 2018 to 2025 [11].

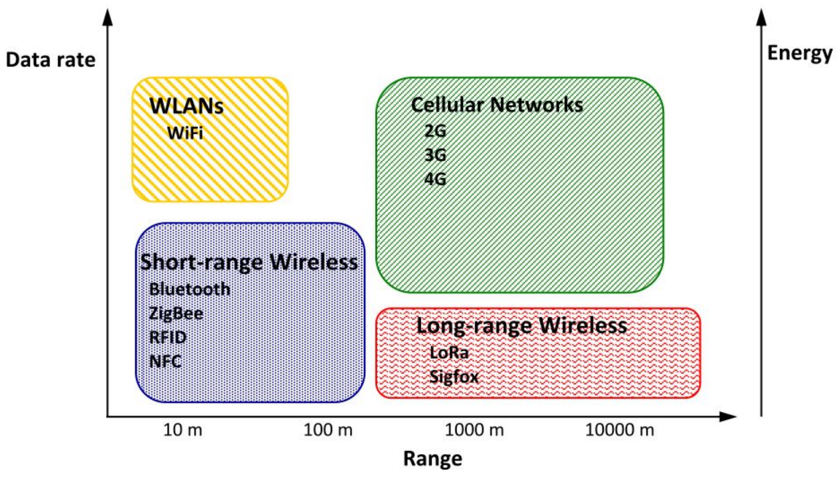


Figure 1.2 – Low-power, long-range radio technologies versus legacy wireless technologies [13].

range, energy consumption and cost of the system.

In contrast with meshed networks providing short range and high throughput, long-range radio technologies allow covering a wide area in a few hundreds of square kilometers, at the lower data rate. To extend communication range most of the technologies operate on low frequencies (sub-1 GHz band) because they have better propagation characteristic through obstacles [13, 15]. Besides, several modulation techniques are employed aiming at more resilient to interference with two common kinds namely narrow-band (e.g. Ultra-Narrow Band, UNB, from Sigfox) and spread spectrum (e.g. Long Range, LoRa, from Semtech). In remote sensing applications, improving the lifetime of the battery power system is a vital requirement. In addition, for each sensor node, communication function consumes more energy than data processing operation, so that scheduling both local processing and communication transaction is an efficient approach to enhance the performance of the systems. Utilizing long-range communications, these networks usually form a star topology (instead of mesh) which enables direct directions between sensor nodes to sink nodes

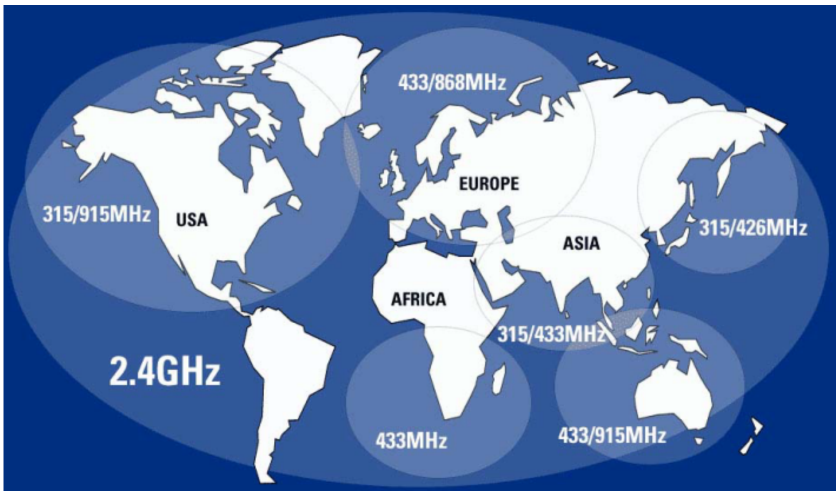


Figure 1.3 – ISM/SRD License-Free Frequency Bands (source: Texas Instruments) [14].

for energy saving advantage and minimizing complexity in protocol design. Reducing the hardware complexity and minimum infrastructure are also concerned aspect of all competing technology. As an example, LoRa systems that are easily deployed as star organizations with nodes consuming as low as 100 mW with a small circuit footprint. Existing technologies enable operation in the licensed band (e.g. NB-IoT, LTE Cat M1), unlicensed ISM band (e.g. LoRa, Sixfox), or both of bands (e.g. Weightless-P). In addition, sensor network over white spaces (SNOW) is an emerging solution because of the availability and advantages of TV spectrum in long-range communication [16, 17]. However, most of the technologies operating in unlicensed spectrum must adopt national radio spectrum rules and regulations about duty-cycled transmission for different countries [13, 15, 18]. Figure 1.3 shows the policy of Industrial, Scientific, and Medical (ISM) frequency bands in different countries over the World.

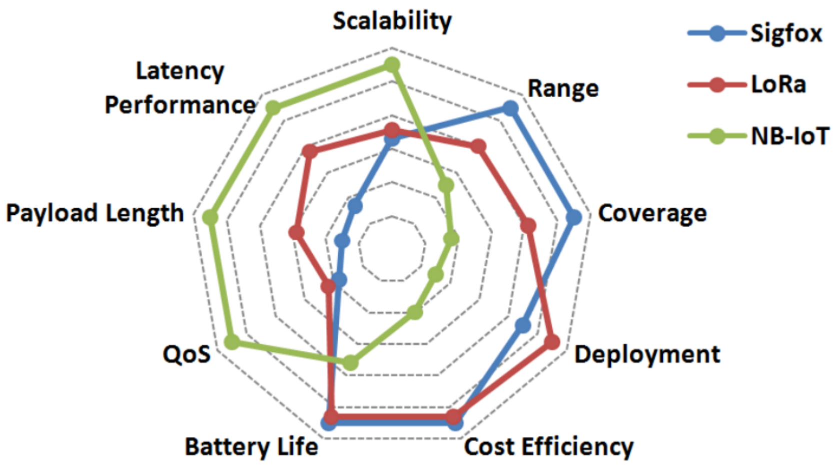


Figure 1.4 – A comparison on technical aspects between Sigfox, LoRa and NB-IoT technologies. It seems that Sigfox and LoRa technology are more suitable for environmental monitoring applications (source: ICT Express) [19].

In spite of providing significant advantages, competing long-range, low-power communication technologies must deal with a lot of challenges in terms of frequency regular, spectrum constraint, coexistence, mobility, scalability, coverage, security, and application-specific requirements such as data rates. Consequently, it is necessary to compromise on several conflict aspects such as spectrum (licensed/unlicensed band), deployment cost, and device cost. Figure 1.4 provides a chart for a visual comparison of technical aspects between Sigfox, LoRa, and NB-IoT.

Long-range sensor network architectures are well suited to situations where monitoring the environment is critical. Mekong Delta in the South of Vietnam is at the first rank of these problems, due to climate change and environmental pollution [20, 21].

In these wireless sensing applications, radio signal estimation is critical because of the geography topology. A zone being reachable from an emitter is defined as the cover of the emitter. Inside cities, the cover can be estimated following statistic measures. In nature, the main obstacles are the geographic profile, climate, and vegetation variations. To ease deployments in natural conditions, it is necessary to develop tools allowing to estimate precisely covers in order to design radio networks [22, 23]. Radio signal propagation can be considered as a physical fact, or as a logical connectivity between nodes. The radio link is securely obtained in the LoS conditions. The flowing Section 1.2.1 summarizes contributions to efficient deployment of WSNs for remote sensing applications with respect to scales adapted to the environments, and more specifically and high-performance LoS algorithm [24].

1.1.2 Radio signal coverage

Because of radio coverage prediction providing many remarkable benefits for wireless network deployment, it is emerging as an attractive topic with a lot of literature research. Some literature surveys present a comprehensive review of the coverage problem in WSNs [25, 26, 27, 28]

To facilitate, optimize and manage the complex systems from WSN behaviors, abstract networks are used as the formal representation for the sensor fields. These abstract networks can be classified into two major types namely binary and probabilistic model according to boolean (0/1) values and the probability of monitoring events respectively. From a technical viewpoint, to satisfy the requirements of monitoring targets, the network of sensor nodes coverages must be deployed in order to cover completely either a definite area, a concerning target, or constructing a barrier to detect intrusion attempts [29, 30].

The wireless network of sensing applications is usually installed in complex geographical areas where are so hard to approach, if not impossible, composed of various land-form, heterogeneous vegetations. Consequently, coverage strategies must be taken into topographic complexity for better estimation results. Several researches on the propagation model [31, 32] took into account irregular terrain effects. In [33, 34, 35], authors proposed algorithms and a graphics tool enabling coverage estimation of the public mobile communications network. The quality of radio wave signals in 2G/3G/4G technology was evaluated with respect to the impact of terrain heterogeneity. The investigation on coverage enhancement for distributed mobile sensors was conducted [36]. Coverage strategies for WSNs in a three-dimensional (3D) or 2.5D environment were also investigated in [37, 38, 39] aiming at improving the reliability of simulation in reality.

1.1.3 Cellular automata

Figure 1.5 presents the application fields of CA technology, of course not exhaustive list.

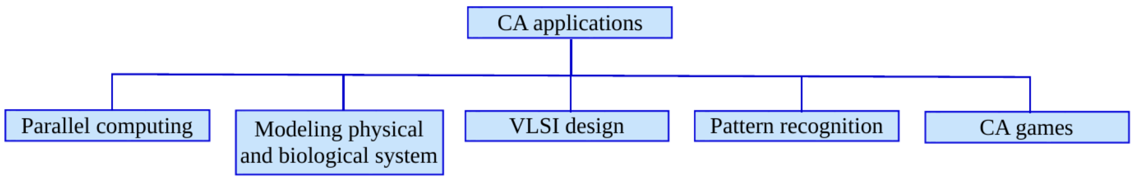


Figure 1.5 – The broad application fields of cellular automata [40, 41].

In the last decade, researches on applications of cellular automata (CA) have focused on several main directions as follows:

Almost studies in energy saving technology for wireless sensor networks (WSNs) aim at design a wireless node can self-organize based on communication state or switching between states to reduce the energy consumption of network and prolong the time service of networks [42, 43, 44].

Besides, many papers present about routing aspect in WSNs based on CA technique. These researches focus on how to implement proper models and the simulated network protocol and topology [45, 46]. Some researches also address the related issues of encryption technique to enhance the security for data communication, as can be seen in [47, 48].

On another approach, many surveys show the research topics in the optimization of coverage based on a cellular approach that attracts so many researchers. They work on both sensing and communication coverage of nodes which was introduced in so many technical papers such as [49, 50, 51].

Cellular methods have been utilized in many different fields of science and engineering such as physics, chemistry, biology, computer science, and communication [18, 52, 53]. It is only synchronous CA but also asynchronous CA to be considered in literature [54]. In recent years, large-scale wireless network designs have been emerged as an attractive topic because of the current innovation solutions such as LoRa and sensor technology progresses [55, 56].

Particularly, use of a cellular automata approach for modeling of natural phenomena in terms of terrain complexity has attracted the interest of several scientists. Climate change impacts and sea level rise, leading to severe weather, salinity intrusion and natural disasters are described in [57, 58, 59, 60, 61]. From these studies, proper plans and forethought can support adaptation and resilience. Besides, other publications have investigated hydrological modeling with respect to homogeneous of landform types, for example, surface water flow simulation, complex river system modeling, flash flood forecasting [62, 63, 64]. Solutions for problems related to land use, land erosion, and landslide have been proposed by [65, 66, 61].

Another related research topic is environmental pollution such as oil spill, air pollution, and so on [67, 68]. Many natural disasters (e.g. volcano, earthquake, tsunami, snow avalanches, etc.) could be forecasted through computer programs based on cellular technology [69, 70, 71, 72]. Modeling wildlife propagation and urban in relation with food, environmental variables are also research fields for CA [73, 74, 75].

In this research, we propose a framework for easing exploration of radio link possibilities for long-range communication, taking into account topology in geography. High-performance processing is employed on Graphics Processing Unit accelerators (GPUs) to enhance the speed of process [76, 77].

1.1.4 Low Earth Orbiting satellite for remote sensing applications

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 [78]. A typical satellite's footprint is thousands of kilometers in diameter. Besides, both systems have already supported all three links namely up-link, downlink 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.

The LEO satellite communication can be used in mobile satellite communications, surveillance of the Earth surface, geological surveys, so on [79, 80, 81]. In the last decade, research on communication services provided by LEO satellites has focused on several main directions such as optimizing the mechanics, interconnections, electric circuits, power supply. On another approach, many surveys show the research topics in design trajectory, handover traffic, and constellation, as well as design protocols, radio frequencies, onboard transceivers and antenna designs [79, 81]. However, the direct radio links between sensor fields and LEO satellites are not considered in the literature. In recent years, it emerges as an attractive topic because of the current innovation solutions such as LoRa and solutions from vendors QB50 [4].

Furthermore, the graph-based model has emerged as well as an approach to present the structure and to elaborate the performance of wireless sensor networks [82, 83]. For example, random geometric graph [84] was used to determine the probability of the whole network being connected. Secure communications between a large number of sensor nodes in WSNs can be elaborated on expander graph [85] and finding transmission path in the network was performed based on Pascal graph [86, 87]. Furthermore, hyper-graph [88] was utilized to support reducing the transmission energy consumption and improving the fault-tolerant ability of the system.

Additionally, in recent years parallelization on GPU platforms is an emerging strategy to improve significantly performance speed [89, 90]. When the graph consisting of a vast number of nodes, it is necessary to process in parallel based on NVIDIA CUDA technology to improve search speed [91, 92, 93, 94, 95].

1.1.5 Massively parallel computation

The architecture of CUDA [96] is SIMD (Simple Instruction Multiple Data) based on many threads run in lockstep. Otherwise, Occam [97] programs are developed as MIMD type programs (Multiple Instruction Multiple Data) where processes synchronize by guard channels.

Parallel computation

A GPU consists of hundreds, if not thousands, of cores to process parallel work-loads efficiently. GPUs have different types of memory such as global memory, local memory, shared memory, texture cache, constant memory, CUDA array, and registers. Each type of memory provides specific use cases associated with appropriate access methods. In GPU architecture, all threads execute the same instructions. This operation can be classified into Single Instruction Multiple Data (SIMD) parallel architectures. Consequently, the SIMD nature of GPUs is a proper approach to simulate the inherent data parallelism existing in the synchronous execution of physical phenomenon.

Concurrent computation

It exists various programming languages providing concurrent processes corresponding with different models of concurrency such as multi processes as operating system processes, multi-threads within a process. In these concurrent computing systems, communications between concurrent components play a key role that could be handled either implicitly or explicitly. In explicit communication method, Occam is known as one of the early concurrent programming languages based on message-passing synchronization primitives in contrast with the other namely shared memory communication.

Occam-pi, a derive of Occam inspired by binding Pi-calculus and Hoare's CSP, was developed by Kent Retargetable Occam Compiler (KRoC) at University of Kent, United Kingdom [98]. Occam and Occam-pi provide a full set of primitive constructs are facilitate to handle not only processes in serial but also in parallel as well as multiple events to a process using construct SEQ(uency), PAR(allel), and ALT(ernate) respectively. Apart from that, applying the CSP idiom of sending (!) and receiving (?) synchronous communications between two concurrent processes over point-to-point nondirectional channel are handled [99, 100, 97, 101] with safe channels, barriers as well as other innovation paradigms for dynamic mobile processes. These facilities allow describing sufficiently both local behaviors/processes, for example sensing activities of nodes and their communications between two processes running on two separate nodes either via integrated network links or buffers in software.

1.2 Contributions in sensing simulations

1.2.1 Radio coverage prediction

With a given zone one of the serious matters for WSN deployment that is how to maximize the coverage of the network. In the WSN domain, coverage is usually classified into two kinds: sensing coverages and communication coverages. Generally, sensing coverages depend on which values need to be concerned and on which technologies sensors are built. In more details, these coverages are related to which mean to be used in measurement such as capacity, microwave, spectrum, etc. and how to capture analog values in nature and then convert them into digital data. Communication coverage refers to a zone consisting of places where radio signals can reachable from a given emitter. In theory, it is indicated by a circle with a certain radius considering as maximum communication range. Due to the complexity of geography and obstacles such as building, vegetation, so on the actual coverages in the real world are seldom as uniform dishes. For this reason, it occurs unpredictable gaps as deploying sensor nodes to cover all sensing area. Similarly, to achieve the robust radio links for gathering sensed data it is necessary to take account of geographic topology since planning and deployment. From these reasons, it is a critical role of CAD tools which help to figure out properly and rapidly the number of sensors and positions for network deployment.

Our tool-set offers a complete flow to fulfill these requirements. A special map browser (called QuickMap) was employed in order to provide us with a friendly visual interface for selecting a zone in their deployment. Other tools are a PickCell and NetGen which help to customize and produce automatically data for high-performance simulations on multi-core CPUs or Graphics Processing Units (GPUs). It also allows to dispatch data to simulators, invokes simulation massive parallel processes. Simulation results are retrieved and shown directly on PickCell at each step of the process or at some concerning steps. For example, in radio coverage prediction simulation after selecting a point on the map to put an emitter (associated with a sensor node) high-performance simulator turns out in real time actual coverage in which signals can reach taking into account of geographic complexity. From these visual results, we are suggested to select proper places within the coverage for placing other nodes to ensure the connectivity. Further steps could be followed in the same way. At any time in design, the whole coverage of network can be realized and visible on user's interface conveniently. Based on the visual results, we know a number of emitters/nodes that must be deployed to cover concerning areas with respect to obstacles along transmission paths, especially in long-range wireless networks with distance tens of kilometers. By following this method, an efficient plan to deploy a wireless network for environmental monitoring is obtained rapidly [102].

1.2.2 Physical simulations

Due to the sophistication of physical phenomena in nature, modeling and simulation tools play a key role to study insight into these processes in terms of characteristics and behaviors. In addition, the natural world is essentially concurrent so that modeling these physical events based on cellular automata technology is wholly suitable for studying them carefully. By applying proper simulations and observations, it can expose so many interesting issues during interaction among phenomena in physical systems [103].

For example, by observing a surface groundwater flow simulation for an area with serious variations of elevations can provide some places where floods caused by dense rains usually occur. These results are interesting issues that are really useful network deployments in order to monitor water levels. The areas constituting a large amount of water (flooding points) are often located in places with low elevations so that it is hard to deploy robust wireless networks to collect measured data from sensors at the points. For these reasons, it is vital to find suitable places to locate base stations concerning both good qualities of radio links to all sensor nodes as well as demands of measurement activities at the interesting points. The order in invoking simulators are probably in different relied on the relationship of physical factors in nature and the objectives of application. A package of simulations designed for a certain purpose will be run successively if the later simulation requires former one's results as its input data, rain flooding and wireless sensor network deployment as an example. In the other cases, physical simulations of the interactions between several simultaneous simulations during the process are critical to describe the behaviors of systems as in the real world. For instance, air pollution causes by dirty smoke, dust produced from a forest fire. This can be achieved by utilizing a run-time infrastructure (RTI), called High-Level Architecture, which enables to exchange data between all federated processes based on standard federation protocols. Adopting this idea, the complete simulation tools for the physical phenomenon are implemented inside our research work based on cellular automata. Flooding and radio coverage predictions are given as examples. In remote sensing domain, observing the behaviors of processes during interaction together to obtain clearly knowledge insight into essential principles of the natural world is as important as capturing realistic values of the phenomenon for validation and evaluation.

1.3 Contributions in sensing activity developments

1.3.1 Mapping application on sensing systems

The objective of this work is to propose a framework for effective implementation of distributed applications such as sensor networks for environmental monitoring. Distributed algorithms are developed and bound as the local behavior of node (Occam-pi programs). Simulators help to validate and evaluate these algorithms. These Occam-pi programs are then transformed to execute on real hardware. This is feasible thanks to a virtual machine (called Transputer Virtual Machine) and KRoC compiler [98, 104] which allow running extended Transputer bytecode on different platforms.

Figure 1.6 shows a stack architecture consisting of

Behavior level distributed applications consists of many distributed services that can be spread over a network of nodes having processing capability. This method allows

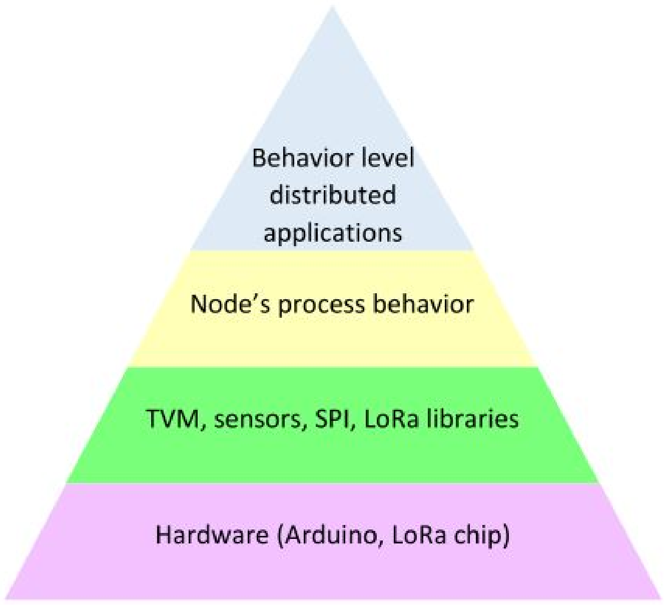

Behavior level distributed applications
Node's process behavior
TVM, sensors, SPI, LoRa libraries
Hardware (Arduino, LoRa chip)

Figure 1.6 – A stack architecture enables distributed applications spread over a network of virtual machines being connected via LoRa radio links.

massively parallel computation to be provided by a large number of nodes of distributed sensor networks.

Node's process behavior are procedures running on each node, for example, Occam-pi programs to collect data from sensors or control actuators.

Transputer Virtual Machine (TVM) is a virtual machine interpreter which enables to execute Occam bytecode. SPI and LoRa libraries help to mount hardware modules such as a LoRa transceiver, sensors, etc. into the virtual machine.

Target hardware is dedicated embedded systems composing of a small processor, sensors, a radio transceiver and a power supply.

An experiment was done to implement a simple network with two nodes using Arduino boards plus LoRa transceivers. In this network, a sink node can dispatch several services on a sensor node to control collecting data from connected sensors (Appendix I).

1.3.2 Modeling and optimization of mobility

The orbit of an LEO satellite is shifted in a westward direction around the polar axis at each revolution [105, 4], as shown in Figure 7.1. This leads to the meeting points of a gateway on the Earth's surface and the LEO satellite will be changed over time. Moreover, gateways can only communicate with LEO satellites when the satellites in their visibility region, generally in a short time approximately 5-10 minutes [105]. With a static sensor field, it can be occasionally unsuccessful in communication with the LEO satellite because the meeting time does not enough for data exchange. Hence a dynamic sensor field (DSF) [79], which has the ability to redetermine its gateway to adapt to the shifts of orbits, aims at improving the connection time.

1.4 Outline of this thesis

This section gives a brief overview of the content of the rest chapters.

Chapter 2: Cellular Automata and Terrain Complexity
Chapter 3: Distributed Algorithm on Cellular System
Chapter 4: Signal Propagation Modeling
Chapter 5: Parallel Algorithms for Simulations
Chapter 6: Mapping Application on Sensing Systems
Chapter 7: Modeling and Optimization of Mobile Connections of Dynamic Sensor Field
Chapter 8: Conclusion and Perspectives

2 Cellular Automata and Terrain Complexity

(also in [1] pages 3-12)

Climate change and natural evolution seriously impact on several social and economic aspects, including living conditions, human health, and development. Autonomous observation is a key point for understanding evolution and the management of territories. This is achieved using small stations equipped with processors, sensors, transceivers, and power supplies. Distributed sensors sample physical behavior. They enable to analyze environmental changes and to predict short-term and medium-term transformations.

Wireless Sensor Networks (WSNs) offered several frameworks to connect sensors. Radio standard such as 802.15.4 [106] propose short range mesh connected topologies suitable for indoor or local deployments. Recent innovations include *low power long range radio* systems, such as *LoRa* [107] or Sigfox [108]. Long range means covering a large surface, more risk during transmissions, long signal latencies, and more difficulties coming from the ground topology. *Star organization* where a central node addresses remote sensors is the more natural technique to collect data on surfaces within a range of ten thousand square meters.

Complex geographic areas such as shores or islands, hills, valleys oppose physical difficulties to the radio propagation. At the sensor level, the *coverage* also defines zones where information is accessible, whatever it may be, wildlife including insects, physical elements such as water or gas. Therefore, a *precise coverage computation* is critical for monitoring efficiency which matches *radio connectivity objectives*, and sensing accuracy in regard to physical phenomena. This article describes principles of a general method based on geo-localized cellular systems. It explains how a radio coverage or a physical coverage can be computed from the same framework, allowing to obtain a better match between observation engines and observed phenomena.

A cellular approach was chosen to address the variety of entities appearing in physical situations. Geography variation is one of them, with elevation accidents, rivers, lakes, forest, shores. By fragmenting geography into cells, it becomes possible to isolate different behaviors, to simulate, then to reassemble these independent behaviors using higher level simulation paradigms, as shown in Figure 2.1 and discussed in [10].

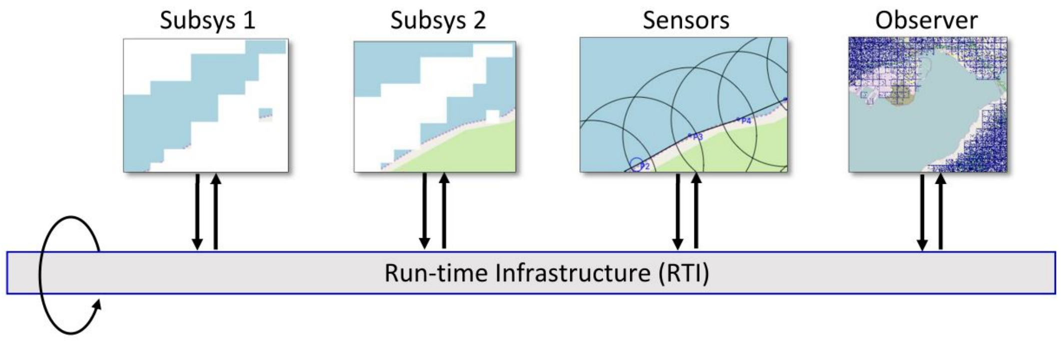


Figure 2.1 – Federation of two complementary cellular subsystems, a sensor network, and some display for observation. A software bus (RTI) provides services for data exchanges, sequencing and synchronization. This framework is called *High Level Architecture*, standardized as IEEE Std 1516-2000 [109].

The approach of fine grain cellular structure and coarse grain partitioning appears to be flexible, allowing different components to coordinate. It is also compatible with geographic information since each cell is localized.

The introduction firstly gives an overview of principles behind three components in the methodology: (i) the terrain analysis (Section 2.1), (ii) the signal propagation and coverage (Section 2.1.5), and (iii) the representation of physical phenomena behaviors (Section 2.1.6). Generic algorithms for sensing deployments can be built in relation to metrics for these components. This section will refer to a grid of points having geometric and geographic coordinates. Preliminary examples present a large region with mountains and creeks along the seashore, likely to be equipped with sensors. This level of explanation matches application needs at an engineering level.

Further sections will detail tool principles, with the presentation of the cellular methodology (Section 2.2), then algorithms which simulate the physical case of flooding (Section 3.2), and long-range radio propagation (Section 4.2). Appendices will present validation results obtained by these algorithms.

2.1 Terrain Complexity Analysis

2.1.1 Geographical Information System and Digital Elevation Model

By utilizing a dedicated analysis on a geographical information system (GIS), digital elevation models (DEMs) can provide a mean value of elevations for a specified zone or discrete values of elevation [110].

Roughness index is commonly defined as various standard deviations of topography [111]. For example, roughness in Equation 2.1 is the difference between the maximum and the minimum elevation of a given cell and its neighbors.

$$Roughness = Max(elev_{(i,j)} - elev_{(0,0)}) - Min(elev_{(i,j)} - elev_{(0,0)}) \tag{2.1}$$

where $elev_{(0,0)}$ is elevation of cell under evaluation. $elev_{(i,j)}$ are elevations of neighbors, and n is number of neighbors in grid cells.

Terrain Ruggedness Index (TRI) and *Topographic Position Index* (TPI) are two common metrics for geographic evaluations. These metrics for DEM provide a quantitative measure of surface turbulence [112].

2.1.2 Terrain Ruggedness Index

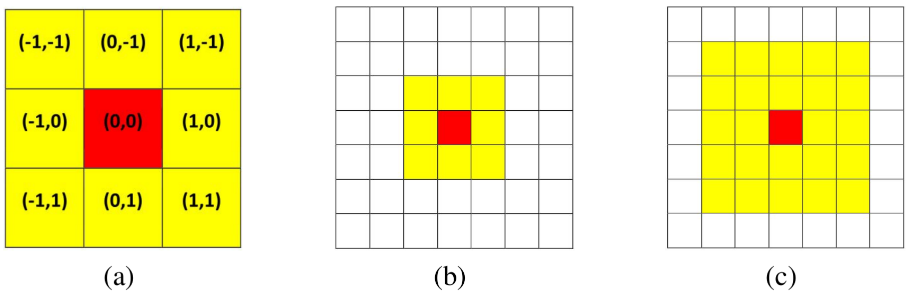


Figure 2.2 – (a) A topographic roughness model describes the relation of elevations between a cell (red cell) and sounding cells highlighted in yellow color. (b) Moore neighborhood for distance 1: a center cell has 8 surrounding cells. (c) Moore neighborhood for distance 2: the center cell has 24 surrounding cells.

The terrain ruggedness index (TRI) is a topographic roughness metric to depict the elevation difference between a center cell and all immediately surrounding cells in a digital elevation model. TRI is the square root of the summed squared deviation in elevation between a cell and its eight neighbor cells (Equation 2.2) [113]. The main value of this topographic roughness index is to provide a rapid measurement of average elevation change between a given cell in the cell system and its surrounding area. This gives a more informative comparison between areas about geographic complexity.

$$TRI = \left[\sum \left(elev_{(i,j)} - elev_{(0,0)}\right)^2\right]^{1/2} \tag{2.2}$$

The TRI values also can be displayed in the form of maps that clearly reveal the distribution of terrain heterogeneity (see Figure 2.3). In our work, we proposed a color scale for displaying TRI values on the map as shown in Figure 2.4.

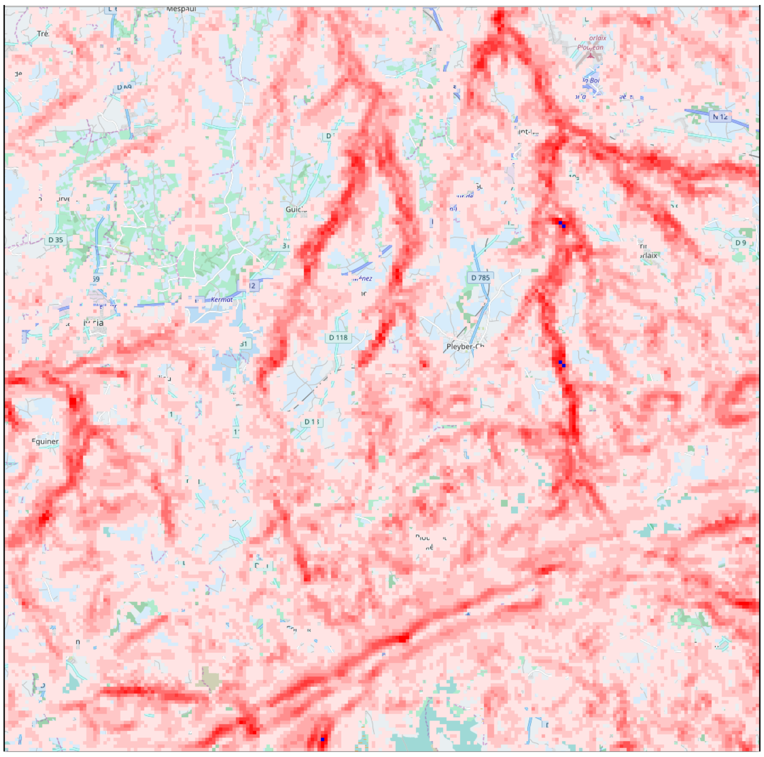


Figure 2.3 – Terrain complexity analysis for the Roc'h Trevezel using TPI metric. A color scale, as in Figure 2.4, is proposed to provide visual results on the map of zone. This grid is 262 $\times$ 226 cells, representing a zone about 50 $\times$ 40 kilometers. This shows remarkable landform being split in the North-South direction.

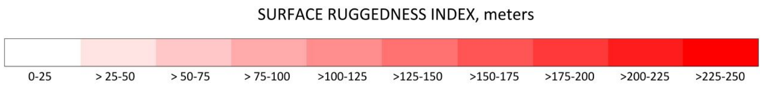


Figure 2.4 – A gradient red color scale with 10 equal steps that represents the TRI values from 0 to 250 meters.

2.1.3 Topographic Position Index

Topographic Position Index (TPI) is the difference in elevation value of a center cell and the mean of eight adjacent neighbor cells (Equation 2.3) [114].

$$TPI = elev_{(0,0)} - \sum_{n} \Big(elev_{(i,j)}\Big)/n \tag{2.3}$$

here n is number of neighbors of $cell_{(0,0)}$.

Because the TPIs may be positive values or negative values, we propose another scale (see Figure 2.6) used the gradient of two colors (blue, and red) to facilitate recognizing the magnitude of TPI for all cells in a zone.

The positive value of TPI refers that the cell is higher while a negative one means it is lower than its adjacent neighbors. It is important that TPI values are essentially very

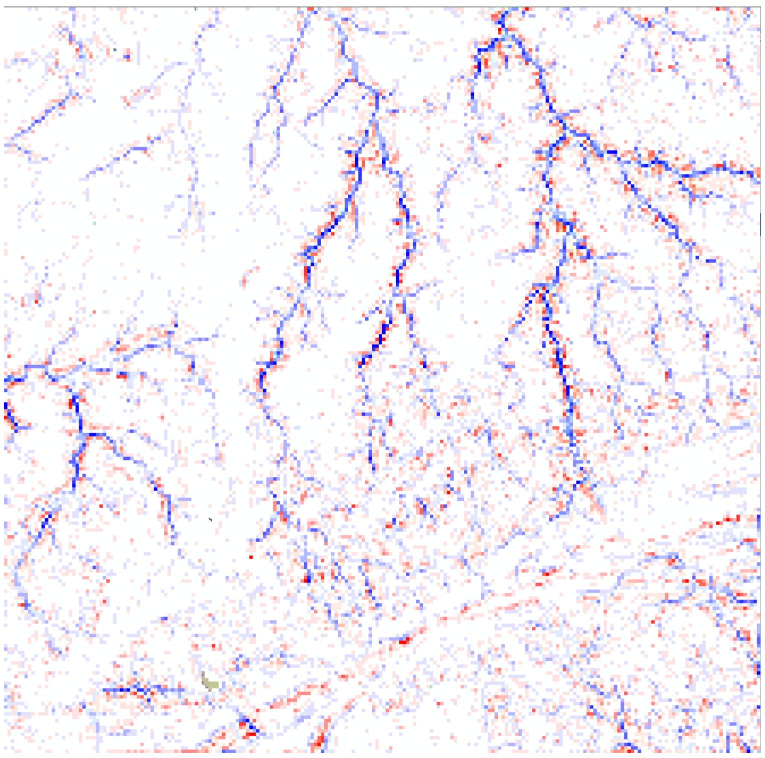


Figure 2.5 – The topographic complexity of Roc'h Trevezel area is quantified using TPI. Red lines represent higher points, difficult to overcome and blue lines are for lower points, difficult to reach. The white zones refer to flat zones almost without remarkable obstacles.

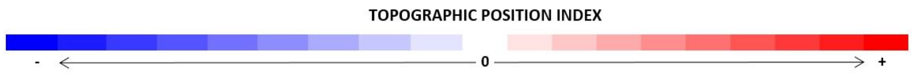


Figure 2.6 – A particular scale based on gradient of red and blue colors represents the TPI values which may be white, red or blue color corresponding to zero, positive, and negative values respectively.

scale-dependent. TPI is usually used to classify the landscape into landform category and define boundaries with respect to its relative position with surroundings.

2.1.4 Accuracy of terrain complexity analysis

It should be aware of the potential errors of data generated from DEMs, particularly when computing and analyzing ruggedness indices. All DEMs contain inherent inaccuracies due to the method to capture elevation were used to generate the DEM data. Apart from this, to represent the elevation at any location most DEMs perform the interpolation and filtering of elevations. In theory, DEM provides the greatest accuracy in smooth terrain and decreases in rough terrain [110].

Another interesting characteristic that is the scale factor for TRI and TPI analysis defined by neighborhood which is perfectly matched with the concept of relationship in cellular automata technology as shown Figure 2.2. With each physical simulations, it should consider intensively which scale is most appreciate with both phenomenon and landscape being analyzed.

2.1.5 Designing for long distance radio coverage

Sensor investigation in a zone almost certainly starts with a glance at a geographic map or an aerial photography. If the zone includes mountains, hills, rivers, or shores, two questions will arise, about the sensing point locations, the reachability of an infrastructure network, the adequacy with the sensing objectives. Design tools will help to take into account the geographical characteristics and possible obstacles to signal propagation while keeping track of these objectives. Sensing systems are application specific, point to point, and differ a lot from mobile radio communication based on regular coverages.

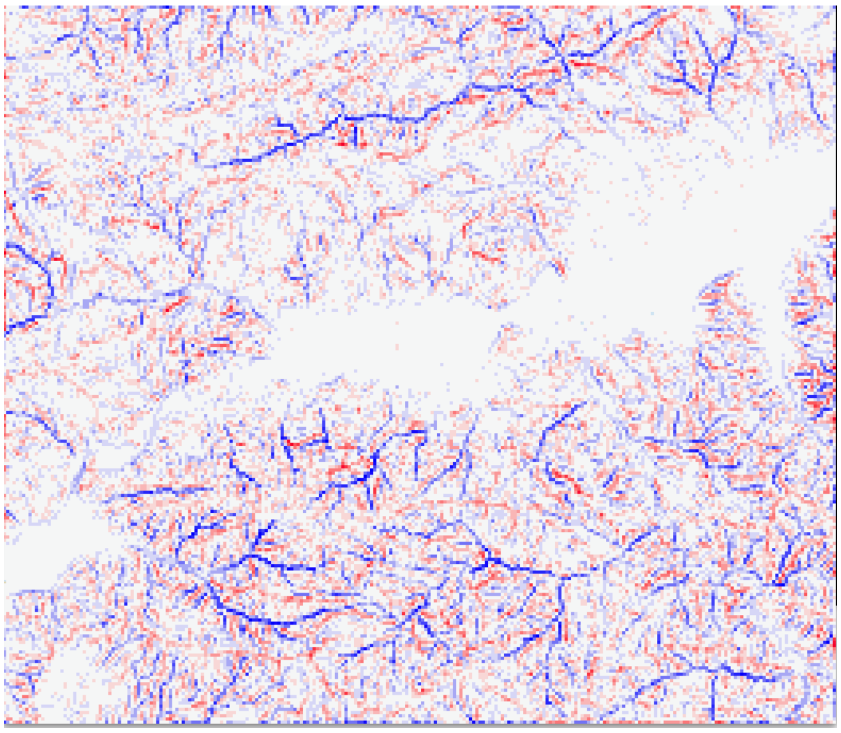


Figure 2.7 – TPI terrain complexity for the Soummam river in Algeria (see also Figure 2.10). Red lines represent higher points, difficult to overcome and blue lines are for lower points, difficult to reach. The white zone signals a flat ground without remarkable obstacles, the case of Soummam banks. This grid is 262 $\times$ 226 points, representing 30 $\times$ 25 kilometers.

Terrain complexity metrics allow to measure ground irregularity, and offer the possibility to split and isolate zones of similar characteristics. Figure 2.7 displays complexity for a complex zone[1] based on Digital Elevation Models (DEMs) analysis.

Both TRIs and TPIs grids illustrate the distribution of terrain heterogeneity and may be displayed in the form of a map as shown in Figure 2.7. Figure 2.8 display analysis result

[1]Soummam Valley, leading to Bejaia city, the north of Algeria.

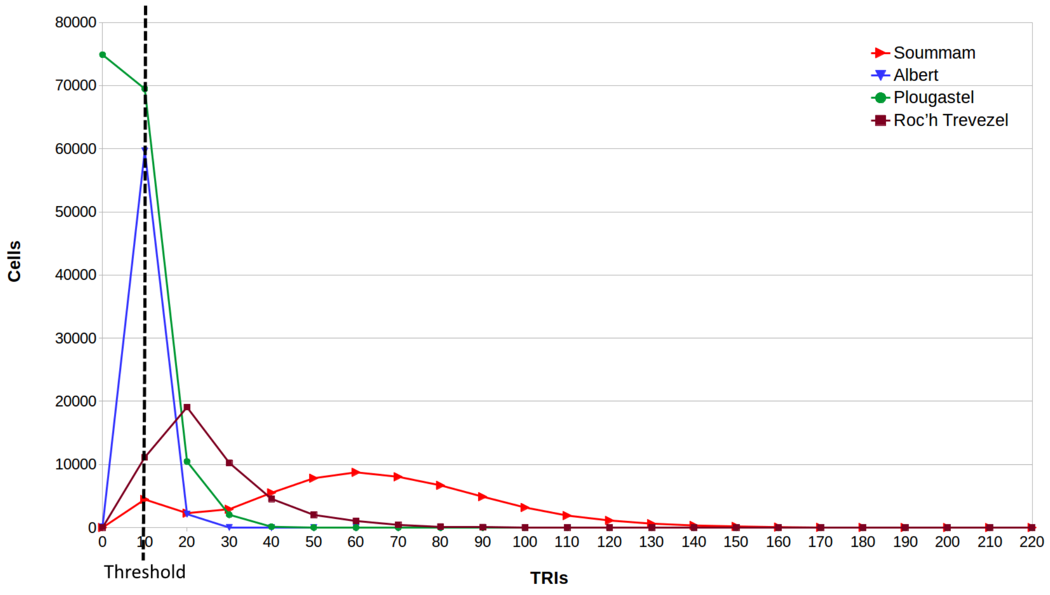


Figure 2.8 – A histogram of topography ruggedness. Four zone analysis with high variability (red, the Soummam), low variability (blue, the City of Brest, with several deep valleys and the shore), a zone with an high percentage of sea surface (green, Brest bay), medium variability with low size hills (brown, the Arrée mountains).

for four cases, also showing how terrain characteristics allow separating concerns about sensor layouts. Partitioning the ground according to terrain complexity is a preliminary operation for network layout and selection of the pair to pair radio transmissions, to be obtained through computer programs.

Setup of long distance radio communication links is difficult if there is an obstacle of the geographical topology. In smooth terrain areas, radio propagation can be considered as *near line-of-sight (NLoS)*. In this case, the communication distance is expressed in relation to power reduction along the path. *Free space path loss (FSPL)* equation was proposed for this aim (see Section 4.3 and Equation 4.7 in Section 4.3).

Concerning complex terrains, the choice of a radio position can be application context dependent, topology-dependent, or it can come from a layout algorithm. Thus, it is critical to obtain a description of the zones covered by a signal, plus other characteristics such as propagation delay from the source, and quality of the signal. Section 4.3 will explain how this can be done using cellular algorithms which mimic the spatial physical propagation over the terrain.

Figure 2.10 illustrates the results of a coverage computation, from an expected source in the middle of the figure. The irregular yellow shape reflects the difficulty to choose source and destination given that any position between them can interrupt the signal propagation. Deploying a network of several nodes on long distances is not tractable by hand.

Further experiments were done for Albert 1^{er}, Plougastel at Brest city in France and the Soummam river in Bejaia. The obtained TRIs and TPIs are summarized in Table 2.1.

According to Riley [113], Albert 1^{er} and Plougastel are considered to represent a level surface. It is a near level surface for Roc'h Trevezel and intermediately rugged surface for

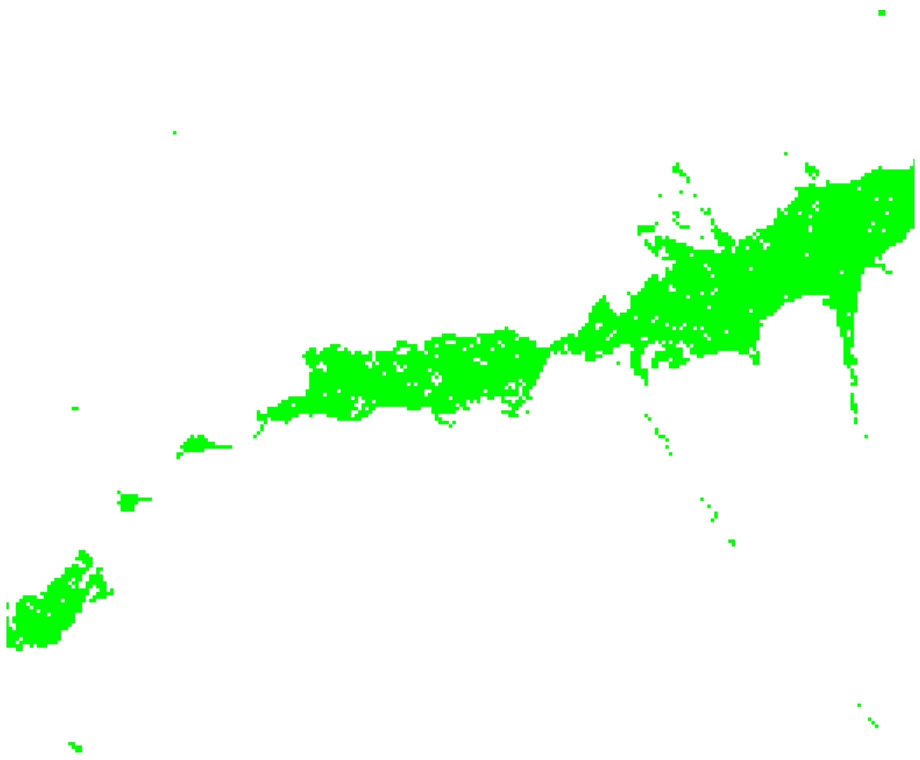


Figure 2.9 – A subsystem of low terrain complexity was extracted from Figure 2.7.This is a flat zone around the river Soummam, with complexity below the threshold line on the left in Figure 2.8.

Soummam area.

In all previous experiments, the ruggedness index values were obtained by executing massive parallel computation on GPUs using DEM (Digital elevation model) data from OpenStreetMap [115] based on cellular technology. The complexity of geography provides reasoning explains for radio signal propagation in the real world. It is a critical metric to evaluate the effectiveness of routing algorithms, LoS algorithm as an example, for robust communications in WSNs. This is a critical reference in order to deploy efficiently wide area coverage networks for environmental sensing, especially in complex topographic areas.

With the cutting-edge of environmental control and surveillance, automatic observation systems are usually employed in complex geographic areas where are so hard for people to

Table 2.1 – Topograhic indices: TRIs and TPIs

				TPIs (m)	
Zone	Number of cells	Resolution (pixels)	TRIs (m)	Maximum	Minimum
Albert 1^{er}	61984	5@5	24.332	5.340	-2.975
Plougastel	156992	2@2	47.202	9.500	-8.000
Roc'h Trevezel	48618	5@5	95.076	27.100	-29.400
Soummam	58725	3@3	214.995	47.820	-44.825

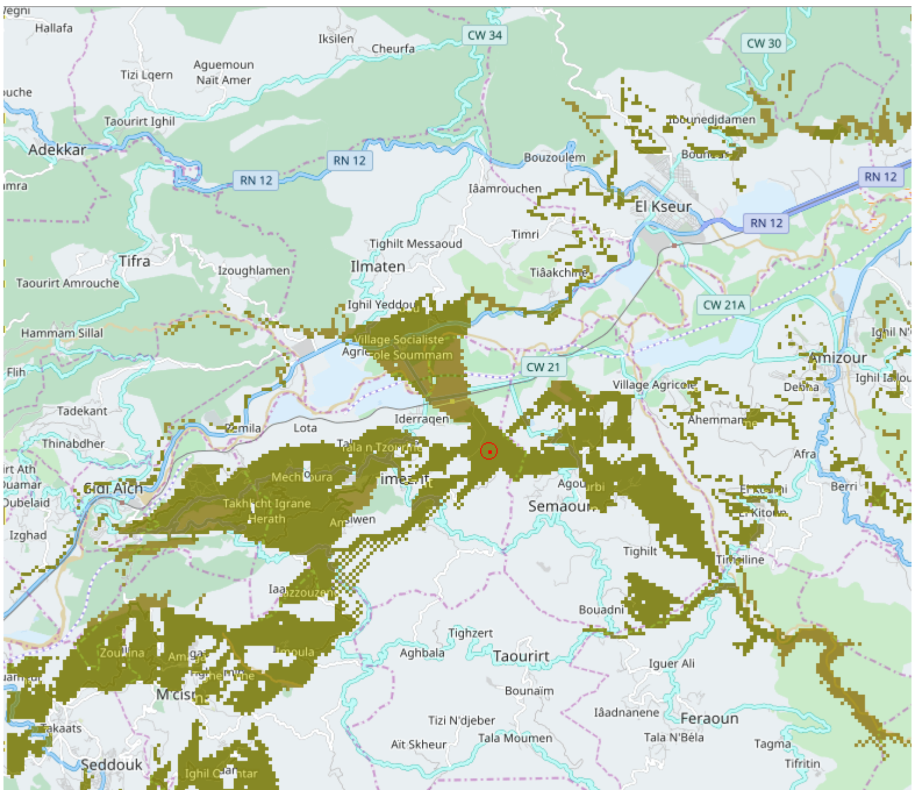


Figure 2.10 – Display of a radio coverage for an emitter (red point), located above Soummam *(36.622141028, 4.799995422)* elev:165.0. Note that geographic positions are proposed in *(lat, long)* form compatible with familiar map navigators. Coverage is shown in dark yellow, spreading over 30 km on the zone width. This location was chosen at random, giving a percentage of 32% grid points receiving the long-range signal.

approach, if not impossible. The knowledge on the topographic complexity of zone helps to facilitate WSN deployments for monitoring environment in such areas. It also gives the overview that networks deployed in complex terrains obviously demand more nodes leading to its cost higher than in smooth areas. For these reasons, terrain complexity analysis provides critical references for network deployment aiming at enhancing the robustness and range of system and reducing the cost as well.

2.1.6 Observing physical phenomena

Another component for observation methodology is the necessary focus on the phenomenon to monitor. The nature of sensing varies a lot, depending on application fields. Sensing can be related to biology, climate, geology, human activities, or composition of parameters. Complex terrains will bring a complex behavior, as it is the case for radio signal propagation.

To illustrate the observation problem, let us consider heavy rain in the same region as in Section 2.1.5. The physical fact to be monitored is water streaming from the mountain to the river, then to the sea. Rain can be predicted from the meteorology services simulations, or just observed using radars, satellites, or ground devices. Causality for water flooding is the nature and shape of the terrain, sharing characteristics with terrain complexity and signal propagation.

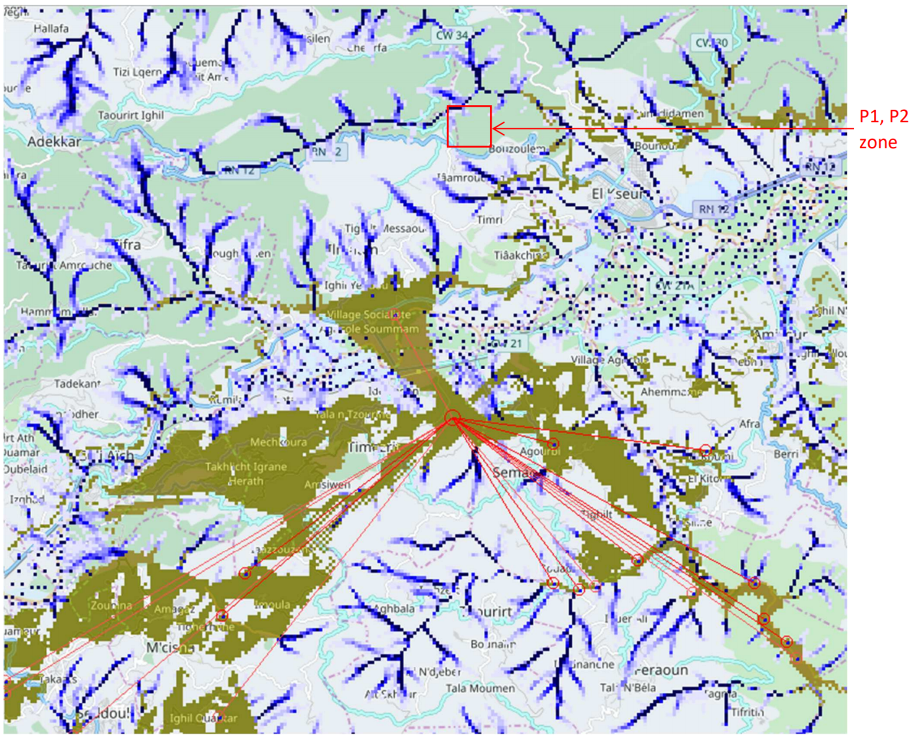


Figure 2.11 – Heavy rainfalls context: a physical simulation produced positions with the risk of flash flooding (dark blue color). The communication coverage of a base station with a star network is predicted for sensor nodes monitoring level of water in these positions. An algorithm select reachable sensor positions from a network sink, sorts, then extracts 15 positions according to the flooding result simulation. More details about accuracy are given in Section 3.2.

Cellular simulation presented Section 2.2 will compute propagation of water given an expected, or existing rain history taken from a public service (this case), or extracted from radar networks [116] or satellite observations [117]. Figure 2.10 displays the water streaming from the mountain down to the river in this particular case of a heavy storm which profile is shown in Figure 2.12.

In the resulting maps, high values represent flooding and possible risks. This analysis demonstrates that simulation allows establishing dependencies between risks and positions. Choice of real sensing positions can be produced by computer programs. For example, the control that appeared in Figure 2.11 can connect to radio coverage zones and obtain measures from critical positions in the rivers. The difference in communication latencies is another problem in the long range. This is also a place where computer tools can help to produce a schedule of sensor communication automatically.

The flexibility of cells embedding local parameters makes possible to represent many physical phenomena such as ocean or river wave effects, rain effects and flooding, air or water pollution, species behavior, sound, and alerts reachable zones, etc. Parallel algorithms were

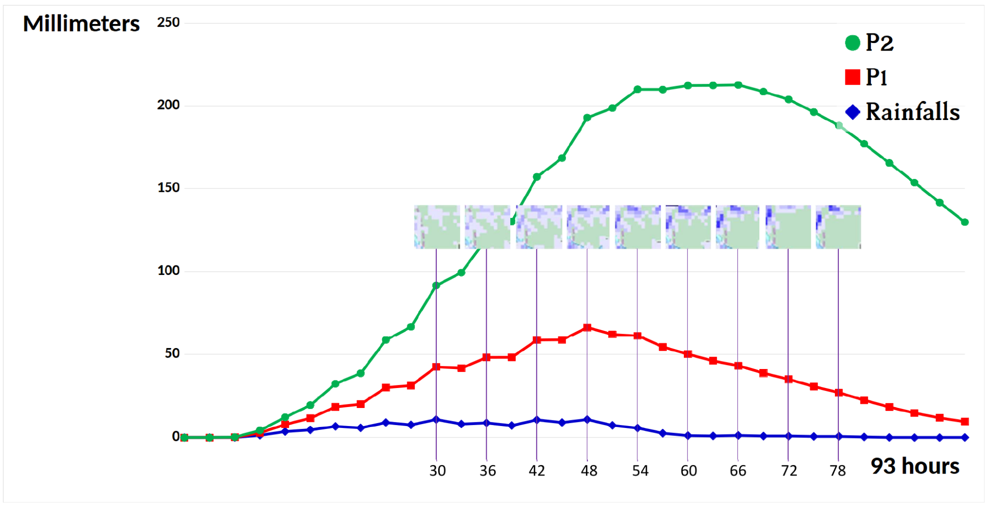


Figure 2.12 – Chart of rainfalls and water levels at two positions in the Soummam river zone during a tropical storm from 13 to 16 November 2017. The blue line shows rainfalls during 4 days recorded every 3 hours [118]. The two lines in red, green color present water level at P1, P2 respectively (also see Table 2.2). The distance between P1 and P2 is 300 meters. Due to the significant slope of the ground surface in this complex terrain, a large amount of water was accumulated at lower points, causing a flash flood, and possibly a catastrophic landslide.

Table 2.2 – Geo-locations of points where values of water level were recorded.

Point	Color	Latitude	Longitude	Elevation (m)
P1	Red	36.589895361	4.925651550	554.7
P2	Green	36.587414366	4.925651550	536.8

designed to simulate rain flooding (Section 3.2)and propagation of the LoRa radio protocol (Section 4.3.1), the objective is to obtain high performances on this kind of problems.

Now, the next section explains the geographical space structuration based on cellular systems.

2.2 Mapping geographic space into cell systems

2.2.1 Geographic map and user interfaces

Several domains such as physics, medicine, or biology need sensing systems and geometric references. The case of geography includes necessary *projections* of a spherical surface into flat maps, with many options depending on applications, precision and considered location. Software tools exist for conversions [119]. Designing observation systems implies to consider distances and geometry relating to the applications, and therefore use of one or several reference systems to manage physical phenomena and their perceptions coherently. This section proposes an explanation of map building and displays to support the meaning of map contents, and data abstraction for geo-localized information additionally fetched from other sources, including simulations.

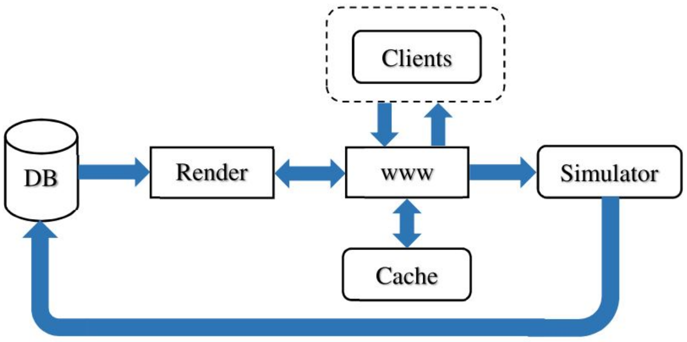


Figure 2.13 – Tile map server architecture: tiles are requested from the web page server that either return a cached image or ask the rendering engine to compose it from a data base of geographical objects.

Tiled web maps are used to represent geographic data and to display information as flat graphics. This appears in proprietary software, or free access map systems such as OpenStreetMap. In *Tiled web maps*, the Earth is accessed by a projection rule called *Web Mercator* [120], and most often by a 3D designation mechanism *XYZ* used by web browsers and tile servers. The Z parameter provides a *zoom factor* from the whole Earth to more and more detailed view of the Earth fragments. At a given zoom factor Z, XY gives the 2D index of a tile inside the rectangle of tiles of this level Z. Based on this, tiles of graphical information is composed and delivered by servers to clients, usually in the form of 256 $\times$ 256 pixel images (Figure 2.14 presents a map composed from such tiles). These images are composed by a server rendering engine that extracts objects from a database and draws them according to some style (see Figure 2.13).

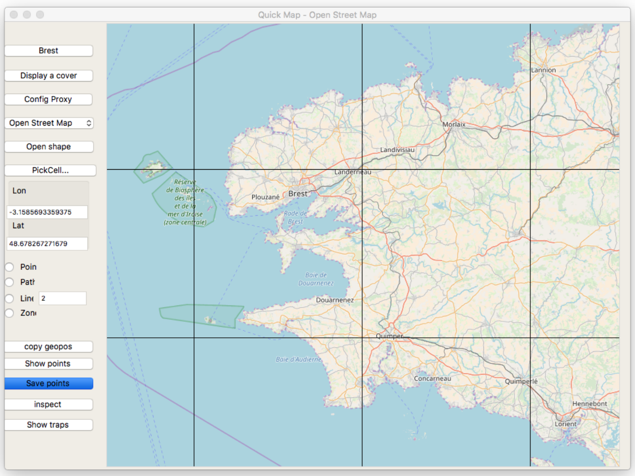


Figure 2.14 – Quickmap tool [121] showing a tile coverage. For this case, zoom factor is 9, and last tile bottom right has x=251, y=177 indexes.

The browser *Quickmap* [121] supports standard map tiles, including OpenStreetMap, and a variety of other items, either for maps or aerial images. This tool also allows to describes sensor systems, communication links, and mobile trajectories, for example, Low Earth Orbiting (LEO) satellites [4].

At a map browser level, users specify or observe *geometric points relative to a zoom level*. Real points are geographical positions, with coordinates specified according to the current projection system. In the case of QuickMap, they are double precision floating point numbers for latitude and longitude. Another concern is the necessity to use real distances in meters, over the Earth surface, for a task such as coverage computation. In the case of Web Mercator transformations between addressing mechanisms for *(lat, long)* absolute geographical specifications (x, y) distance in meters, (x, y) in pixels and XYZ tile access operations can be managed with formula from [122].

2.2.2 Definition of cells

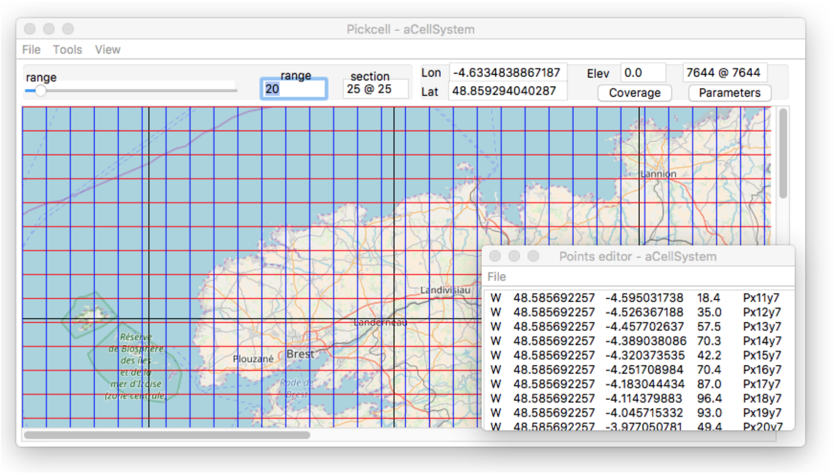
Geographical cells can now be defined as *objects* grouping local data, local system behaviors, and graphic representation. This is described as an object-oriented class, as variables, and methods that are strictly local to the cells.

- *Cell locations* are bounded to geometric locations on maps, in relation to tile containers, and when possible to geographical locations.
- *Basic cell contents* are extracted from the map or image fragment as supported by the browser tool.
- *Cell content extensions* are obtained from external databases. Most of the cases are digital elevations, and also climate, or weather characteristics and historical data.
- *Cell size* is chosen to match a particular physical phenomenon or sensing requirements.
- *Cell behaviors* are local procedures operating on a cell state. They need to be programmed to produce simulation data that in turn can be sent back to databases, or displayed on tiles.

Examples in Section 2 illustrate how geographical abstraction, representation, and simulation can interact (Figures 2.7, 2.10, and F.1).

Given a partitioning of a geographical zone into a cell system, we need to relate the observation system and physical behavior. The possibility to simulate physical processes will be of great help to correctly characterize a layout of sensors in connection with physical evolution.

Cellular automata (CA) allow modeling physical activities as processes that exchange information and evolve according to transition rules. The cell concept binds geographical fragments to such processes whose assembly is generated automatically.



```
W  48.585692257  -4.595031738  18.4   Px11y7
W  48.585692257  -4.526367188  35.0   Px12y7
W  48.585692257  -4.457702637  57.5   Px13y7
W  48.585692257  -4.389038086  70.3   Px14y7
W  48.585692257  -4.320373535  42.2   Px15y7
W  48.585692257  -4.251708984  70.4   Px16y7
W  48.585692257  -4.183044434  87.0   Px17y7
W  48.585692257  -4.114379883  96.4   Px18y7
W  48.585692257  -4.045715332  93.0   Px19y7
W  48.585692257  -3.977050781  49.4   Px20y7
```

Figure 2.15 – Presentation of a cell system organization over the tiles of Figure 2.14. Each cell has an identity produced from its location inside the window, and a geographical location. The text window bottom right also displays a plus parameter for the elevation. The cell size is 25 × 25 pixels, representing 7644 × 7644 meters.

2.2.3 Cellular automata principles

Synchronous systems

Cellular automata were invented by John Von Neumann and colleagues with the aim to build a self-reproducing machine abstraction [123]. To support this goal, a two-dimensional space was configured with automata governed by a small set of states. This representation of space was used in several scientific domains and bound to physical behavior having similar properties [124]. CA can be described, and specified as a discrete space which associates cells.

In *synchronous cellular automata*, cells synchronously evolve, steps by steps, following a discrete time. Such CA have been described in a variety of languages and executed on specific machines [125] following four simple patterns:

- The *cellular space* is represented by an assembly of similar cells. A common notion of *neighborhoods* defines local communications following observed physical dependencies. The spatial organization can be either regular or irregular, possibly with disconnected subsystems as shown in Figure 2.17, item 3.

- The evolution of each cell is defined in a *set of states* as observed in a real system (quantities, colors, boolean). Change of states are operated by procedures associated representing transition rules from step to step: $State_t \rightarrow State_{t+1}$.

- The *neighborhood* represents physical dependencies, for example, signal propagation, or downward flooding. These dependencies are connectivities from cell to neighbor cells. Common neighborhoods are Von Neumann and Moore with 4 and 8 cardinal directions respectively. Item *"Process architecture"* in Figure 2.17 illustrates a Moore neighborhood.

- The *transition rule* defines the behavior of each cell evolution under influence of its neighborhood and local *sensed* influences. The state of the whole cell system synchronously changes, time step by time step.

Nondeterministic behaviors include random variation at the physical level. This was used in lattice gas simulation [126] and to represent life cycle and species evolutions [127].

Variability in large systems and asynchronism

Considering large systems, synchronism and massive data parallel execution can be an obstacle, both in behavior modeling and performances because of sparse data spaces.

Asynchronous cellular automata were proposed to represent reactive systems where events are propagated in a way similar to Communicating Sequential Processes [128]. However, in the case of physical simulation, sampling and time references are often mandatory. Variability of computation frequency can be obtained by isolation and simulation of subsystems. This was implemented on GPUs as related in [129].

The High Level Architecture (HLA) also allows to sequence cellular sub-systems at a different speed and to provide data exchanges between them. For example, [10] shows a countryside simulation with forest, forest fire, sensors and river pollution composed together.

Thus, we can consider the grouping of cells into synchronous subsystems as an efficient way to manage variability. Composing simulations at a high level is shown in Figure 2.1, and was technically discussed in [10].

2.2.4 Cellular automata parallel execution models

A sequential execution model will read states in an array A_t for time t, and loop over, writing a similar array A_{t+1}. Once the step is completed, the two arrays are exchanged, and a new turn begins. CA is easy to parallelize preserving this behavior, grouping operations together and observing turns completion. Parallelism is mandatory because problems are large, if not huge, a lot of small cells can be critical for simulation precision, and some applications require to examine very large geographical regions.

- The synchronous distributed messaging model [130] can support parallel computation by associating cells to communicating processes. In this case, process progress by locked steps, based on messages being sent and received to or from neighbor nodes. The steps are split into two phases, one for communications with the neighborhood, the other to execute the transition rule. Figure 3.8 shows an internal node representation and the outside connection with three input and output links. This model does not need to specify the relative speed of processes, and can therefore be used for multi-cores or supercomputers.

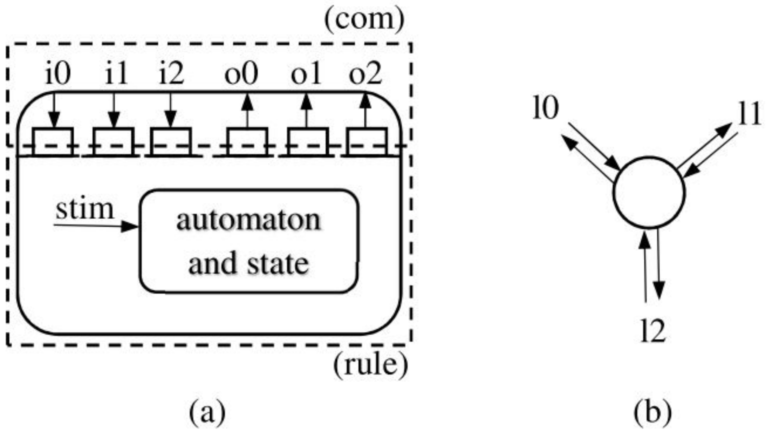


Figure 2.16 – Cell node representation: (a) is the internal architecture with an automaton (*rule*) operating on incoming values and local stimuli (*stim*), based on a local set of variables, filling output communication buffers. Communications (*com*) are operated to and from the input and output buffers. (b) is the external point of view that only shows bidirectional links a cell node.

- Another way to take advantage of parallelism is to use data parallel Single Instruction Multiple Data (SIMD) processors to execute a group of processes simultaneously. Current graphics accelerators propose solutions up to two thousand processors working concurrently, and exchanging data synchronously in shared memory. State spaces must be copied to the accelerator memory, then a loop of steps can be run completely on the acceleration, which is very efficient.

Production of code from cell systems to this schema has been done for two targets:

1. Asynchronized Occam communicating processes [131, 132] targeting KRoC compiler and multi-cores [104],

2. CUDA code production [8] targeting *NVIDIA* tools and accelerators.

2.2.5 Cellular systems work flow organization

Most of the work described in this article is supported by a set of dedicated tools from the University of Bretagne Occidentale (UBO). The design flow can be summarized as follows:

1. **Zone selection** is done by moving a graphical window anywhere, with any level of zoom. The tool extracts graphic tiles, and display the contents.

2. **Cell segmentation** is obtained by splitting the view into rectangles of a given size specified as a width × height value. Cells will carry an image and a geographical location from the underlying image.

3. **Binding cell together** and producing a cell system implies the choice of a connectivity (Moore, Von Neumann), and possibly filtering cells by colors or elevation. This step also injects external values from a variety of sources, including elevation.

4. **Adding behavior** is programmed into the cellular system at the local level, given a cell system architecture that step 3 (Figure 2.17) automatically produces.

Steps 1 to 3 are interactive and can be achieved *in minutes*. Step 4 is a concurrent programming activity specific to a concurrent platform which requires elaboration of CA transition rule and coding for the target platform.

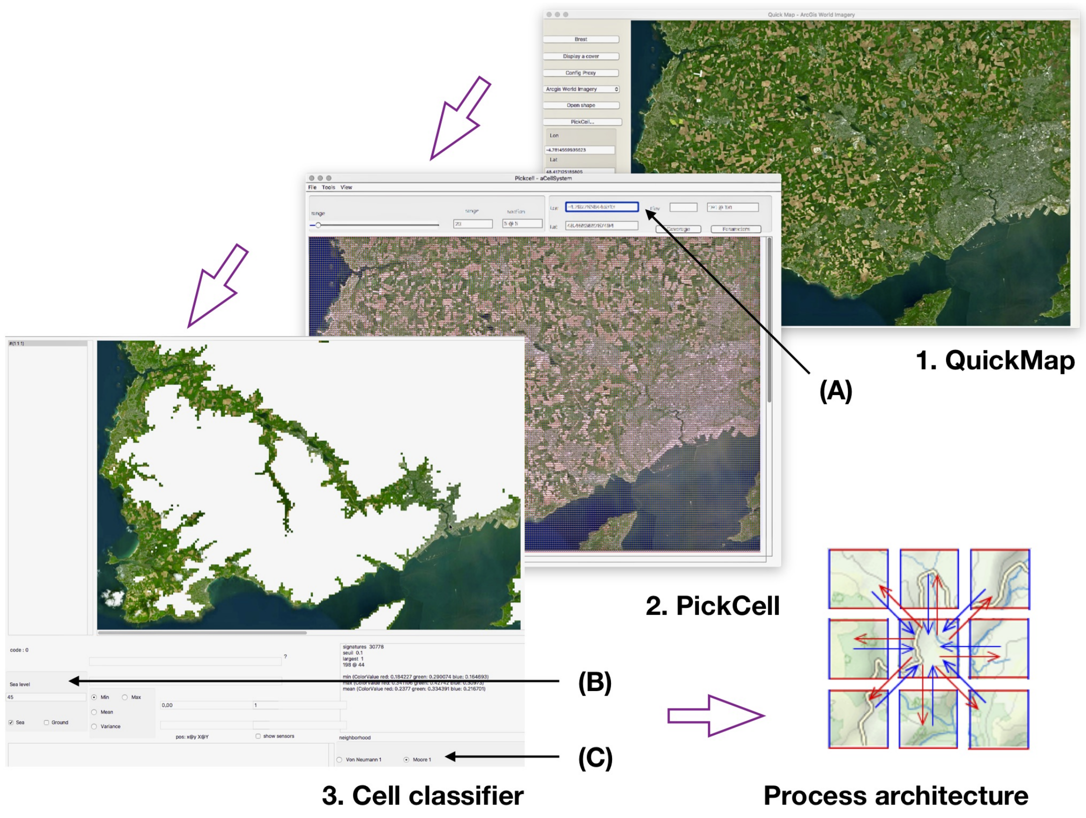


Figure 2.17 – Cell synthesis flow: (1) a zone was located from a QuickMap navigation, and (2) was segmented into cells, then a subsystem was extracted filtering cells with elevations less than 45 m. (3) A cell system was generated following Moore topology. Annotations show controls for geographical positions with a cell size of 5×5 pixels representing a 191×191 m^2 surface (A), classification was operated for elevation (B). The neighborhood was Moore, radius 1 (C).

Physical phenomena simulation can reveal places of interest for sensing. In the case of flooding, simulation track accumulation and circulation of water.

In complex terrain areas, heterogeneity of topologies seriously impacts the quality of radio links, especially for low power and long-range communication networks. Different propagation models allow representing the median of the expected path loss such as the Longley-Rice model, the ITU model, the Okumura-Hata model, as described in Section 4.3.

Table 2.3 – Three areas with different complexity terrains. *Resolution* and *Actual size* columns give the size in pixels associated with actual size in meters of each cell in regular grid data. The maximum communication in each experiment is shown by column *Range*. *Match* and *Mismatch* column provides a number of visible points which are right matched and mismatched respectively between simulation results and obtained values in real measurements.

Area name	Description	Resolution	Actual size	Range	Match	Mismatch	% Error
Albert 1^{er}	Urban area	5×5 pixels	24 m	3 km	64	13	20.21%
Plougastel	River, its banks	5×5 pixels	96 m	9 km	129	4	3.10 %
Roc'h Trevezel	Mountain area	5×5 pixels	191 m	11 km	45	8	17.78%

Section 4.4 gives details on experimental measurements, with the communication distance checked to be around 5 km in complex urban area and up to 20 km in rural and shore ones. Experiment results shown in Table 2.3 confirm the interest of a computer-aided approach. It may be concluded that communication in a LoRa network is strongly dependent on the considered environment. Hence, the actual coverage prediction must take topographic complexity into account as a critical factor. An assessment of physical simulation was also conducted for flash flooding problems (Section 3.2). This algorithm was applied to the practical case of an intense rain found on June 3, 2018, and compared with flooding observations. The rain occurred in a rough terrain urban area and the simulation was able to retrieve major flooding places and level of water. There is a major interest in this kind of physical simulations that reveals places of interest for sensing and establish causality between events.

Table 2.4 – A characterization of space and signal relations.

Datagrid	Model	Resolution (m)
SRTM30	2D/3D	30 [2]
SRTM90	2D/3D	90 [3]
Weather broadcast	2D/3D	3000[4]
LiDAR	3D	5[5]
Coastal ocean	2D/3D	1200 [6]
Earth magnetic	2D/3D	3700 [7]
Ocean acoustics	2D/3D	2500 [8]
Road traffic noise	2D/3D	0.5 [9]
Pickcell	2D/2.5D	30

The UBO tool-set is easy to use for cellular system generation. Implementing cellular transition rules, tuning and verifying these rules necessitate in-depth investigations. Several domains were investigated, from sound propagation to insect behaviors. Thus the methodology appears very general and flexible (see Table 2.4).

Pickcell/NetGen tools enable selecting either Von Neumann or Moore neighborhood for distances 1 and 2 [133, 134] as segmenting and weaving processes to generate cell system. Grid data produced by NetGen [135], the scale is determined associated with a neighborhood of regular grid cells. For this reason, it should consider carefully which scale is most appreciate with both phenomenon and landscape being analyzed.

Parallel programs implement described algorithms very efficiently (Tables 2.5 and 2.6. Efficiency is mandatory if space exploration strategies are to be developed.

Table 2.5 – Execution times for three processes. There are 58725 cells in a regular squared grid with a cell size 3×3 pixels corresponding to actual area 115×115 meters.

Processes	Number of rounds	Time (s)
Compute ruggedness index	1	0.026
Flood simulation	32	0.103
Coverage prediction	261	0.211

Table 2.6 – Execution times for flood simulations which are performed on Linux Ubuntu PCs with IntelCore i3-4005U CPU @ 1.70GHz x 4, 8 GiB DDRAM, card NVIDIA GeForce 820M (96 CUDA Cores), Intel Core i7 CPU 920 @ 2.67GHz x 8, 4 GiB DDRAM, card NVIDIA GeForce GTX 680 (1536 CUDA Cores) and Intel Core i7-7700K CPU @ 4.20GHz x 8, 16 GiB DDRAM, card NVIDIA GeForce GTX 1070 (1920 CUDA Cores).

		Execution time		
Resolution (pixels)	Number of cells	820M	GTX 680	GTX 1070
3x3	58725	28.168 (ms)	6.5269 (ms)	1.3373 (ms)
5x5	21060	10.411 (ms)	2.1728 (ms)	518.64 (μs)
10x10	5226	2.5234 (ms)	454.59 (μs)	83.696 (μs)
15x15	2340	1.1155 (ms)	181.48 (μs)	65.916 (μs)
20x20	1287	602.47 (μs)	171.19 (μs)	62.085 (μs)

[2] https://lta.cr.usgs.gov/SRTM1Arc"https://lta.cr.usgs.gov/SRTM1Arc

[3] https://blogs.esri.com/esri/arcgis/2015/06/26/terrain-3d-now-with-global-srtm-30-meter-content/""

[4] https://www.ncdc.noaa.gov/data-access/model-data/model-datasets/numerical-weather-prediction

[5] https://content.meteoblue.com/en/research-development/data-sources/nmm-modelling/model-domain"

[6] https://pdfs.semanticscholar.org/55be/a487827c28ªaaf713017c499e4f33ªed62fd.pdf

[7] http://www.sciencedirect.com/science/article/pii/S0377042798002465

[8] https://www.ngdc.noaa.gov/geomag/emag2.html

[9] Peter Wille. Chapter 5: The Sea Floor - Natural Formations. In *Sound Images of the Ocean: in Research and Monitoring*. Springer-Verlag Berlin Heidelberg, 2005. ISBN 978-3-540-27910-5

3
Distributed Algorithms on Cellular System

(also in [1] pages 12-15)

This chapter presents the development of cellular distributed algorithms for physical phenomena simulations based on cellular automata approach.

Section 3.1 provides an overview of cellular automata technology. The classification of distributed algorithm computations on a cellular system and how to map these algorithms on different architectures are described.

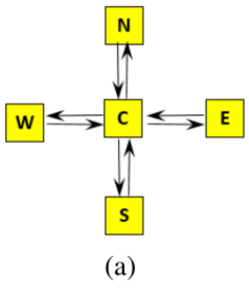
Section 3.2 presents the cellular simulation of flash flooding taking into account geographic topology. A distributed parallel algorithm is developed to compute rainwater distribution with respect to the difference of elevations.

Section 3.3 describes two study cases on flooding simulation in Guadeloupe and Morlaix city, France for validation and evaluation the correctness of simulation results.

3.1 Cellular system

A physical system is defined by a portion of the physical universe. For example, the radio signal propagates in space that could be affected by obstacles along its path or environment. Due to sophisticated behaviors and large scale, the model of systems confronts many issues. Cellular automata (CA) has been utilized for modeling the system.

Cell system: First, cell system composes of a set of cells is defined based on the CA model. Each cell holds its local characteristics (as parameters of state) such as longitude, latitude, elevation, and received power. In addition, cell has connections and directions to its neighbors (adjust cells, Figure 3.1. According to the kind of neighbor (Von Neumann or Moore) directions can be organized as pairs of number, as shown in Table 3.1.



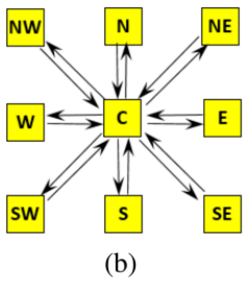


Figure 3.1 – (a) Von Neumann 1 topology consists of a central cell and 4 neighbors. (b) In Moore 1 topology, a central cell connects with all 8 neighbors around.

Von Neumann 1			Moore 1		
Directions	Values	LinkIndex	Directions	Values	LinkIndex
East (E)	(1,0)	0	East (E)	(1,0)	0
North (N)	(0,-1)	1	North (N)	(0,-1)	2
West (W)	(-1,0)	2	West (W)	(-1,0)	4
South (S)	(0,1)	3	South (S)	(0,1)	6
			North-East (NE)	(1,-1)	1
			North-West (NW)	(-1,-1)	3
			South-West (SW)	(-1,1)	5
			South-East (SE)	(1,1)	7

Table 3.1 – Direction encoding for CA Von Neumann and Moore topologies.

Cell behavior: A series of systematic processes to simulate physical phenomenon defines the behavior of a cell. In synchronous distributed systems, at each step cell behavior that consists of several procedures, must send a message, then process incoming message and compute local values for the next state.

3.1.1 Cellular system topologies

From Figure 2.17, we had learn that cell systems are suitable to present zones extracted from geography.

1. **Browsing the geography**: Firstly, a zone planned for a WSN deployment can be chosen by moving on QuickMap browser. This geographical tile browser provides a various kind of map as illustrated in Fig. 3.2). Other functions include sensor placements either manually or randomly. The browser can also display building file descriptions, or mobiles such as drones, Low Earth orbit satellites[4].

2. **Segmentation and classification**: PickCell represents the second step in the investigations, where an image is segmented into rectangles of arbitrary size, counted in pixels. The real dimensions and positions are displayed inside the window header as shown in Figure 3.3.

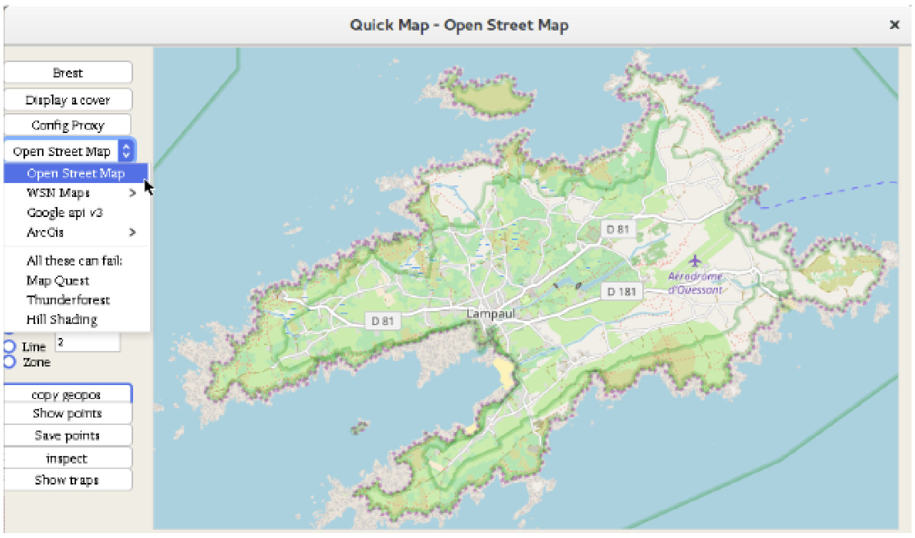
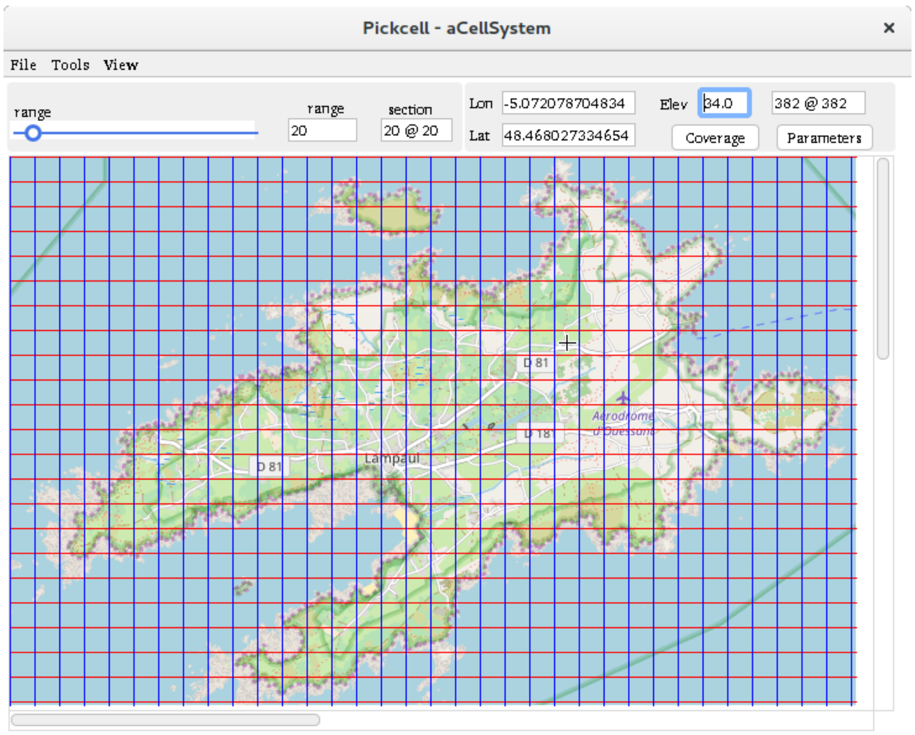


Figure 3.2 – QuickMap: a map browser allows to select various kind of tile server for retrieving tile map for free GIS database such as OpenStreetMap. In this figure, QuickMap is used to select a area at Ouessant island (15.58 km^2) in Brittany, France.



range

range 20

section 20 @ 20

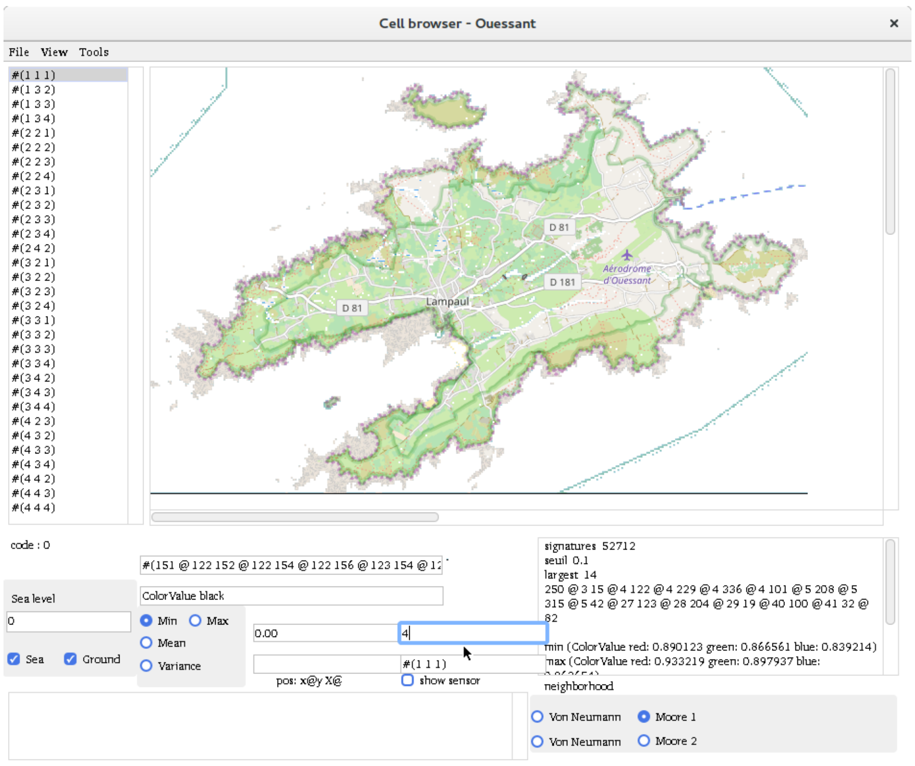
Lon -5.072078704834

Lat 48.468027334654

Elev 34.0

382 @ 382

Figure 3.3 – PickCell cellular presentation with 20×20 pixels cells corresponding to 382 meters. The top fields present geo-position and elevation at mouse point.



```
#(1 1 1)
#(1 3 2)
#(1 3 3)
#(1 3 4)
#(2 2 1)
#(2 2 2)
#(2 2 3)
#(2 2 4)
#(2 3 1)
#(2 3 2)
#(2 3 3)
#(2 3 4)
#(2 4 2)
#(3 2 1)
#(3 2 2)
#(3 2 3)
#(3 2 4)
#(3 3 1)
#(3 3 2)
#(3 3 3)
#(3 3 4)
#(3 4 2)
#(3 4 3)
#(3 4 4)
#(4 2 3)
#(4 3 2)
#(4 3 3)
#(4 3 4)
#(4 4 2)
#(4 4 3)
#(4 4 4)
```

code : 0

#(151 @ 122 152 @ 122 154 @ 122 156 @ 123 154 @ 12

Sea level: 0 — [✓ Sea] [✓ Ground]

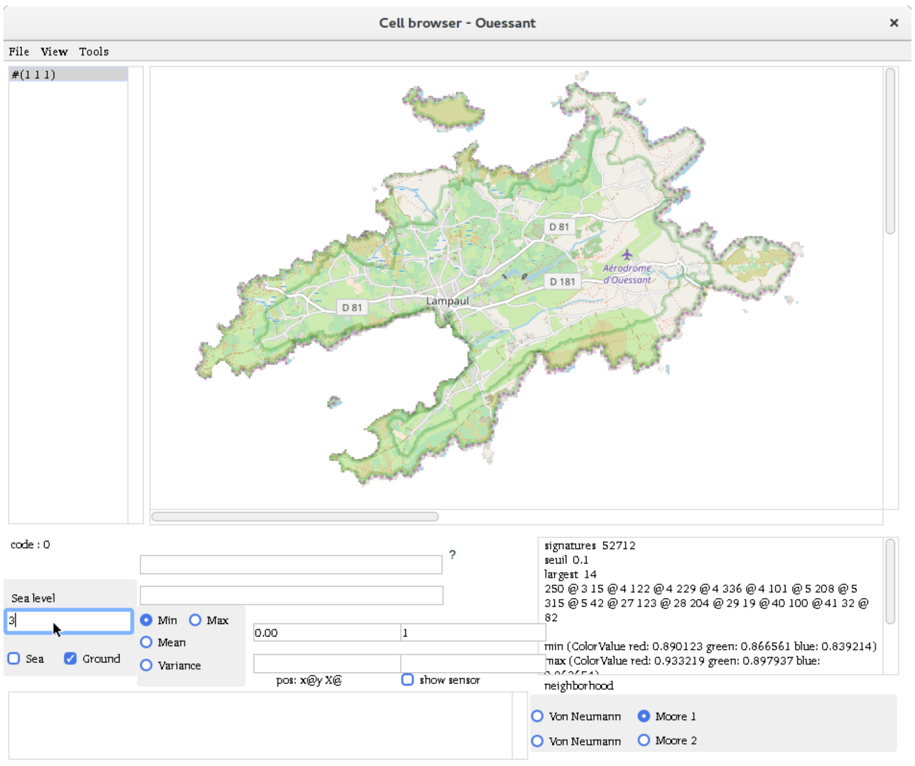
ColorValue black

[✓ Min] Max Mean Variance — 0.00 — 4 — #(1 1 1) — pos: x@y X@ — show sensor

```
signatures 52712
seuil 0.1
largest 14
250 @ 3 15 @ 4 122 @ 4 229 @ 4 336 @ 4 101 @ 5 208 @ 5
315 @ 5 42 @ 27 123 @ 28 204 @ 29 19 @ 40 100 @ 41 32 @
82

min (ColorValue red: 0.890123 green: 0.866561 blue: 0.839214)
max (ColorValue red: 0.933219 green: 0.897937 blue:
```

neighborhood: Von Neumann [✓ Moore 1] Von Neumann Moore 2

Figure 3.4 – This figure demonstrates how to control the classification process depending on Red-Green-Blue color scale for cell system preparation.



```
#(1 1 1)
```

code : 0

?

Sea level: 3 — Sea [✓ Ground]

[✓ Min] Max Mean Variance — 0.00 — 1 — pos: x@y X@ — show sensor

```
signatures 52712
seuil 0.1
largest 14
250 @ 3 15 @ 4 122 @ 4 229 @ 4 336 @ 4 101 @ 5 208 @ 5
315 @ 5 42 @ 27 123 @ 28 204 @ 29 19 @ 40 100 @ 41 32 @
82

min (ColorValue red: 0.890123 green: 0.866561 blue: 0.839214)
max (ColorValue red: 0.933219 green: 0.897937 blue:
```

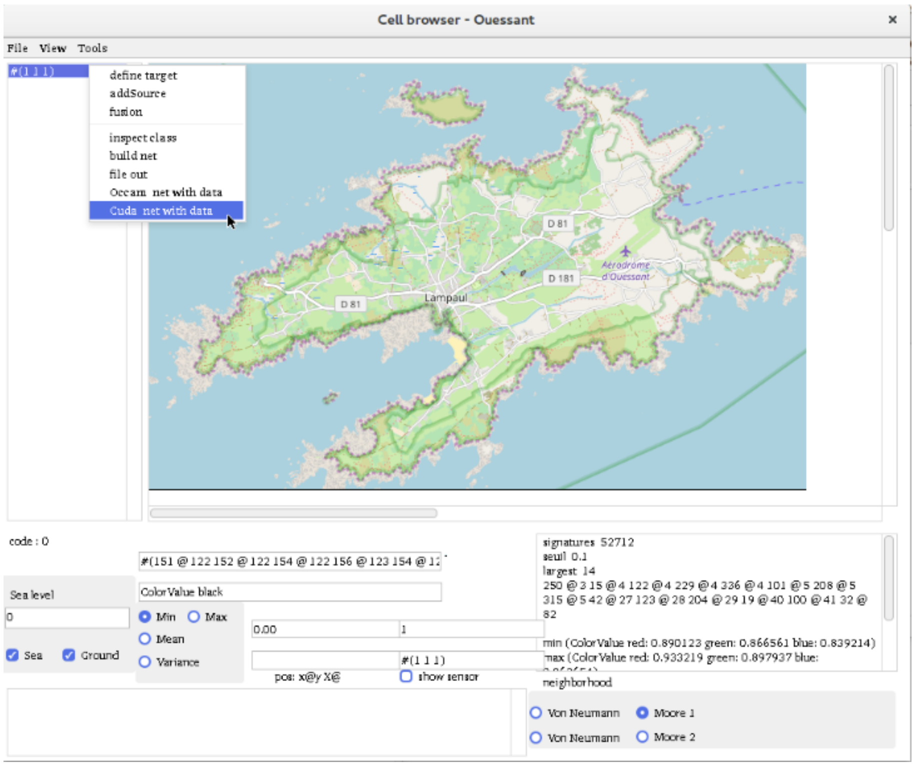
neighborhood: Von Neumann [✓ Moore 1] Von Neumann Moore 2

Figure 3.5 – Instead of using color space, ground elevation parameter is exploited for the classification process based on a critical characteristic of cells, such as 3 meters over sea level.

A window allows to select and control the cell system organization with a number of parameters. One of them is the connectivity between cells according to cellular automata concept. Four kinds of standard neighborhoods are provided such as Moore with eight connections representing physical influences.

Another window allows controlling weaving of physical cells to form complete cell systems or a selection of subsystems. To obtain subsystems, it is necessary to apply a classification algorithm. Standard classifications include Red-Green-Blue color space analysis or/and according to required parameters with repartition into a number of sub-cubes.

Figure 3.5 shows a typical classification based on RGB color scale and Figure 3.4 describes an enhanced data preparation of relying on elevation parameter. The cubes are displayed in the left column of the tool. By selecting a complete system, or subsystem, it becomes possible to write out its organization, according to a concurrent execution syntax.



code : 0

#(151 @ 122 152 @ 122 154 @ 122 156 @ 123 154 @ 1:

ColorValue black

Sea level: 0

0.00 | 1

#(1 1 1)

pos: x@y X@

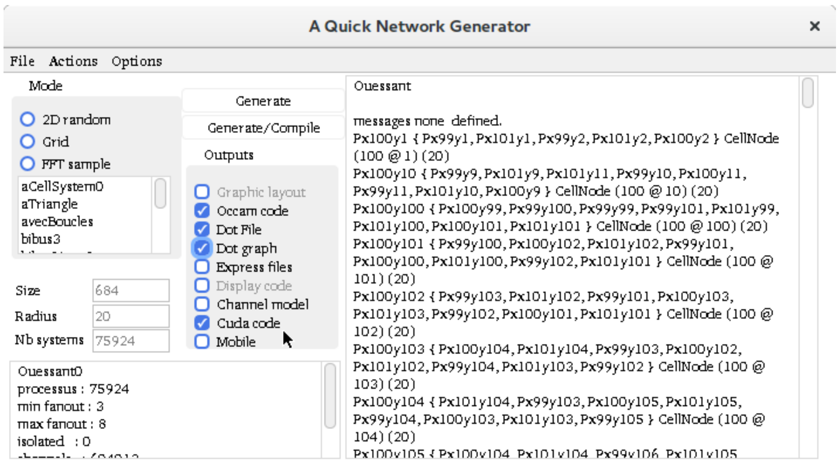
```
signatures 52712
seuil 0.1
largest 14
250 @ 3 15 @ 4 122 @ 4 229 @ 4 336 @ 4 101 @ 5 208 @ 5
315 @ 5 42 @ 27 123 @ 28 204 @ 29 19 @ 40 100 @ 41 32 @
82

min (ColorValue red: 0.890123 green: 0.866561 blue: 0.839214)
max (ColorValue red: 0.933219 green: 0.897937 blue:
```

Figure 3.6 – Cell weaving and simulation synthesis: back-end selection (menu) and cellular topologies (bottom right).

Cell systems are produced by retrieving data from a geography server for geo-locations (latitude, longitude) such as OpenStreetMap. Moreover, the cell system could be integrated into further information sources, for example, ground elevations and/or weather data including wind, rainfall, and so on by retrieving from NASA Shuttle Radar Topography Mission (SRTM) and public weather services respectively. Figure

3.6 will output an organization of 52712 cells in the cellular system.

3. **Cellular systems production** To generate input data for simulators automatically, NetGen [135] is developed as the last facility of tool-set Quickmap/PickCell/NetGen that facilitates implementing simulation processes based on CA methodology. It enables to produce data for simulators both in Occam [97] running on multi-cores CPU and CUDA performing on NVIDIA Graphics Processing Unit (GPUs) [136]. In addition, a visual graph for the abstract network of cell system could be produced by this tool. This graph presents the relationship between cells, considered as processors, and directed links within the cell system to exchange data.



2D random
Grid
FFT sample

aCellSystem0
aTriangle
avecBoucles
bibus3

Graphic layout
[✓ Occam code]
[✓ Dot File]
[✓ Dot graph]
Express files
Display code
Channel model
[✓ Cuda code]
Mobile

Size: 684
Radius: 20
Nb systems: 75924

```
Ouessant0
processus : 75924
min fanout : 3
max fanout : 8
isolated  : 0
```

```
Ouessant

messages none  defined.
Px100y1 { Px99y1,Px101y1,Px99y2,Px101y2,Px100y2 } CellNode
(100 @ 1) (20)
Px100y10 { Px99y9,Px101y9,Px101y11,Px99y10,Px100y11,
Px99y11,Px101y10,Px100y9 } CellNode (100 @ 10) (20)
Px100y100 { Px100y99,Px99y100,Px99y99,Px99y101,Px101y99,
Px101y100,Px100y101,Px101y101 } CellNode (100 @ 100) (20)
Px100y101 { Px99y100,Px100y102,Px101y102,Px99y101,
Px100y100,Px101y100,Px99y102,Px101y101 } CellNode (100 @
101) (20)
Px100y102 { Px99y103,Px101y102,Px99y101,Px100y103,
Px101y103,Px99y102,Px100y101,Px101y101 } CellNode (100 @
102) (20)
Px100y103 { Px100y104,Px101y104,Px99y103,Px100y102,
Px101y102,Px99y104,Px101y103,Px99y102 } CellNode (100 @
103) (20)
Px100y104 { Px101y104,Px99y103,Px100y105,Px101y105,
Px99y104,Px100y103,Px101y103,Px99y105 } CellNode (100 @
104) (20)
```

Figure 3.7 – NetGen tool allows to generate data for massive parallelism processes such as multi-cores in Occam-pi or Graphics Processing Unit accelerators in CUDA.

3.1.2 Algorithmic models

Depending on the behavior of the process, distributed algorithms on a cellular system could be classified into four major types as following:

1. **Pure local computation**: In order to perform distributed algorithms on a cellular system, it is critical that every cell (process)[1] cell must realize about adjacent neighbors. This knowledge is obtained by sharing cells' values together via communication channels. For instance, each cell can at first broadcast its identity (*id*) to all connections which are assigned unique values (*linkIndex*) corresponding to cardinal directions to allow management. After divulgating their own *id*, the cells need to obtain information from neighbors. Based on obtained messages *id* associated with *linkIndex*, each cell can explore all its neighbors around in terms of *(id, linkIndex)*. Another example is ruggedness index computation in which the metric of the topographic roughness of

[1]In the following discussion, depending on context *cell*, *node* and *process* term are used interchangeably to mention about one cell in a cell system corresponding to a distributed process.

each cell is figured out using received elevation values from surrounding neighbors, as mentioned in Section 2.1.

2. **Discuss computation**: In some physical simulations (e.g. radio signal propagation or pollution diffusion in water or air), the processes are required to explore the space step by step. The number of steps, *n*, should be considered as the diameter of an abstract dish being made from all engaged cells.

3. **Symmetric global computation**: To explore the topology of all network properly every node must be executed the process for information exchange relied on message passing at least *MaxNodes* rounds, where *MaxNodes* is a number of nodes. Apart from being obtained knowledge about the attributes of surrounding neighbors such as nodes' latitude, longitude, and elevation, the maximum diameter of all networks may be another target at the convergent time of processes. This diameter is always less than *MaxNodes* and afterward, it becomes a new efficient value of round for other processes.

4. **Dissymmetric explore computation**: Routing messages from one process to the other processes is a mandatory task in WSNs. In this case, the order of processes (cells) in a routing path is important to switch a message towards a definite destination cell. For this reason, the distributed computing processes must be performed in proper order.

Table 3.2 – Summary of algorithmic models for distributed computation on cellular system.

Model	Number of steps	Distance
Pure local computation	1	1
Discuss computation	n	n
Symmetric global computation	MaxNodes	
Dissymmetric explore computation	MaxNodes	

3.1.3 Mapping algorithmic models on different architectures

The CA execution model is a synchronous distributed model (see [130]) that can be formulated to represent physical systems. It can also be simulated on high-performance architectures due to the presence of implicit *barriers* that isolate evolution stages on discrete time boundaries. Time steps produce a high level of parallelism suitable for most of the high-performance computer architectures: thread-level parallelism on multi-core computers, Graphics Processing Units (GPU), FPGA [137] or message passing supercomputers.

In a simulator implementation, cells are naturally associated with *processes*. Barriers isolate communications from state evolution. All communications are achieved on incoming or outgoing buffers, read and fed by the transition program:

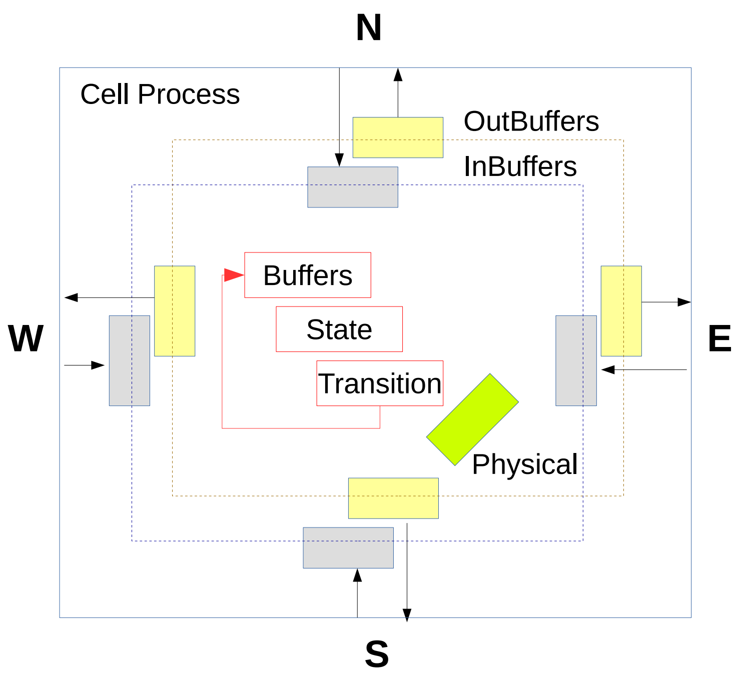


Figure 3.8 – A cell process showing a 3 stage loop that operates on input and output buffers, handle a physical evolution with sensor data, and execute transition rule.

Communicating processes such as Occam [97] or varieties of message passing. Communications are achieved on blocking channels that infers barriers. Figure 3.8 shows the internal organization of processes.

GPUs communication are achieved by a specific code copying buffers to buffers inside shared memory. Each cell has an identity used to execute these copies.

Due to the code generation in the tool framework, it is possible to connect processes by channels, as it is the case for Occam, or by indexes on neighbor buffer arrays (Table 3.3), as it is the case with CUDA. Algorithm 1 and 2 are two fragment code for neighbor exploration in the network in Occam and CUDA respectively that provides a brief comparative explain about concurrent processing on multi-core CPUs and massively parallel processing on GPUs.

3.2 Flash flooding simulation on a complex terrain

To illustrate cell system interactions and behavior, we use the example of a zone *receiving* and *flooding* rain, as discussed in Section 2.1.6.

3.2.1 Managing space

Space is defined by geographical coordinate bounds, and a possible selection operated on cells. Selected cells have common properties, such as an elevation above or under a threshold

Table 3.3 – Routing table

Id	Direction	LinkIndex
17	NW	3
23	N	2
3	NE	1
85	NW	3
.	.	.
.	.	.
.	.	.

Algorithm 1 Cell behavior is defined by calls of synchronous procedures operating on Occam channels with barriers.

```
PROC exploreNeighbors()
    SEQ
        InitLocalVal()                                  ▷ Initialize local variables
        SEQ i=0 FOR (SIZE outChannels)
            outMessages[i] ! localValues
        SEQ turns=0 FOR MaxNodes
            SEQ
                PAR
                    PAR i=0 FOR (SIZE inChannels)
                        inChannels[i] ? inMessages[i]
                    PAR i=0 FOR (SIZE outChannels)
                        outChannels[i] ! outMessages[i]
                SEQ i=0 FOR (SIZE inChannels)
                    UpdateLocalValues(localValues, inMessages[i])
                SEQ i=0 FOR (SIZE outChannels)
                    outMessages[i] ! localValues
```

level (Figure 2.17, step C), some color characteristics, or some signature produced from observed parameters. Thus the cell system can be rectangular or have arbitrary shapes and isolated subsystems.

Practically, cell systems first appear as arrays of objects that carry geometric coordinates, geographic coordinates, and a pixel array extracted from the original image.

Besides the zone structure, designers need to specify neighborhood, representing local physical influences and producing the necessary connectivity between processes. We can just admit that the space animation will be obtained by messages exchanged between neighbors. Execution software and hardware will allow adaptation to the intermediate communicating process model.

The case of rain reveals several interactions:

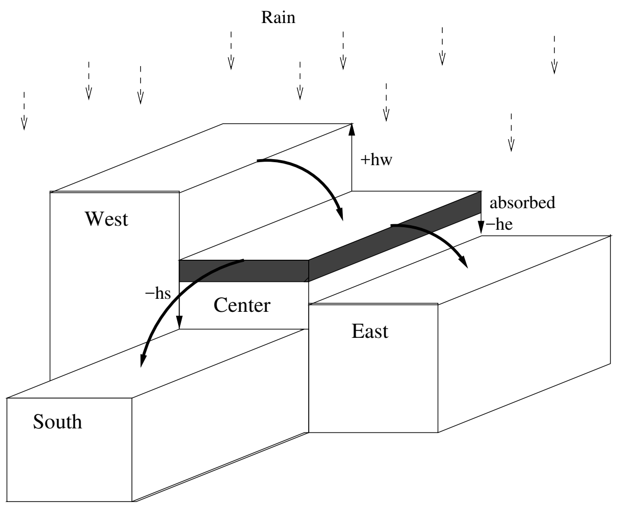
1. Millimeters of water falling on the ground. For this simulation, we admit that the quantity can vary with over time, but will remain uniform.

Algorithm 2 explore Neighbor cells in CUDA

```
 1: procedure EXPLORENEIGHBORS(neighborTab *tabs, Buff *buff, Channel *channel, int numNodes, turn)
 2:     idx = blockIdx.x × blockDim.x + threadIdx.x;
 3:     if (idx < numNodes) then
 4:         nbIn = channel[idx].nbIn;
 5:         ▷ Using the message from last step
 6:         buff[idx].messageTab[0] = buff[idx].messageTab[1];
 7:         ▷ Communicating
 8:         ResetMessage(message);
 9:         for linkIndex=0 to nbIn do                                    ▷ Receiving
10:             nodeIn = channel[idx].read[linkIndex].node;
11:             thisMessage = buff[nodeIn].messageTab[0];
12:             thisLimit = thisMessage.limit;
13:             for entry=0 to thisLimit do                         ▷ Local processing
14:                 anInputId = thisMessage.newFound[entry];
15:                 found = FoundIdInTab(anInputId, tabs[idx]);
16:                 if (found == FALSE) then                              ▷ Updating
17:                     AddIdInMessage(anInputId, message);
18:                     AddIdInNeighborTab(anInputId, turn+1, linkIndex, tabs[idx]);
19:                 end if
20:             end for
21:         end for
22:         buff[idx].messsageTab[1] = message;                               ▷ Sending
23:     end if
24: end procedure
```

2. Water disappearing locally for reasons such as absorption or evaporation.
3. Water passed *locally* from cell to cell according to elevation differences.

Other cases include different specific problems for signal propagation (Chapter 4), sound propagation [138], or insect swarms behavior [127]. For flood modeling, Von Neumann neighborhood is convenient. Figure 3.9 shows such a neighborhood, with the North cell lost, and some elevation differences appearing above and below a center cell.



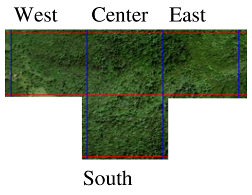


Figure 3.9 – Physical exchange during a rain episode. An incomplete neighborhood from a system shows a center cell with 3 neighbors: west, east, and south. The physical behavior is water flowing downward, represented here by synchronous messages sending water quantity west to center, and center to the east. Refer to [139] for more realistic behaviors.

3.2.2 Transition rule

Each cell will receive rainwater, dispatch part of this water to neighbor cells with lower elevation, and absorb another part. A complete study will take into account the ground specific characteristics, and current weather[2]. While real dependencies are water leaking from cell to cell, the abstract behavior is represented by messages sent and received to and from neighbors. Vertical water behavior is directly encoded in the transition rule that can be specified as follows:

t: time t represented by a step number

Q_t: water quantity in a center cell at the beginning of time step t

α_t: rainfall at time t

[2]ECOCLIMAP [140] supporting French meteorology AROME model provides more than 20 parameters for cells of $1km \times 1km$

β: percentage of water remaining on each cell after each step

$cell_i.elevation$: elevation value of $cell_i$

$cell_c.elevation$: elevation value of center cell

δ_i: the difference of elevation between neighbor $cell_i$ and the center cell

Δ: sum of all elevation differences

$outF_i$: amount of water out coming from center cell to neighbor $cell_i$

inF_i: amount of water in coming to center cell from neighbor $cell_i$

The water quantity of center cell at time $t+1$ after local absorption is

$$remaining = Q_t \times \beta \tag{3.1}$$

But the center cell receives rainfall within its area as following

$$rain = \alpha_t \tag{3.2}$$

The quantity of water coming from neighbors is taken into account

$$received = \sum_{i=1}^{n} inF_i \tag{3.3}$$

where n is a number of neighbors of the center cell.

A center cell also has to distribute water to surrounding neighbors in the proportion of elevation differences. This is calculated as follows

$$\delta_i = cell_c.elevation - cell_i.elevation \tag{3.4}$$

It is obvious that the water of a center cell only flows to neighbor cells with lower elevations so that only positive values of $delta_i$ are used to compute the total proportion of all differences

$$elevationAbove = \Delta = \sum \delta_i, \forall \delta_i > 0 \tag{3.5}$$

Equation 3.6 figures out the amount of water that a cell distributes to its lower neighbors. For higher elevation neighbors, with $\delta_i < 0$, there is no water flowing out, $outF_i = 0$.

$$sent_i = outF_i = Q_t \times (\delta_i/\Delta), \forall \delta_i > 0 \tag{3.6}$$

Eventually, center cell updates its own quantity of water by adding and subtracting. The whole system executes a synchronous transition from t to $t+1$.

$$Q_{t+1} := remaining + rain + received - \sum_{i=1}^{n} sent_i \tag{3.7}$$

Notice that this transition rule needs to know the neighbor relative elevations $cell_i.elevation$.

3.2.3 Algorithm of rainwater distribution

By adopting CSP (Communicating Sequential Processes), rainwater distributed computing process comprises four main steps with a main loop including step 3 and 4:

- *Step 1:* Initializing local variables
- *Step 2:* Preparing for the first communication session
- *Step 3:* Communicating M to N
- *Step 4:* Updating local variables

In reality, physical rules will discover these relations naturally. In the case of simulations, a preliminary procedure called *neighborhood discovery*, as shown in Algorithm 3 is executed to establish initial knowledge such as the number and the elevations of cells around. Note that the following Algorithms borrow some keywords such as *SEQ, PAR, ALT, etc.* from Occam programming language [97].

3.3 Case Studies

The complexity of terrains is one of key factors causing flash flooding during heavy rain episode. To validate simulations, we did two experiments with two different terrains: a volcano mountain area in Guadeloupe island and an urban area in Morlaix, France.

3.3.1 Fist case study in Guadeloupe, France

The West of Guadeloupe has several mountains such as La Grande Soufrière, the highest mountain peak in the Lesser Antilles, with an elevation of 1467 meters or Citerne volcano. Because of the terrain complexity of this area, the high risk of flash flooding and mudslides must be studied intensively.

Maria hurricane hit Guadeloupe, one of five overseas regions of France. This is an island in the Caribbean sea where has suffered many tropical storms and hurricanes every year. This natural disaster caused a lot of serious impact on all aspects of human beings as well as economic and social development. According to Caribbean360 [3] At least two deaths and large areas were destroyed. It is also 80000, or 40 percent, of the homes on the island, were without power in several days. "The total economic cost of Maria was roughly estimated around USD65 billion" as mentioned in an official report of Aon Benfield [4].

[3]http://www.caribbean360.com/news/hurricane-Maria-blamed-two-deaths-guadeloupe

[4]http://thoughtleadership.aonbenfield.com/Documents/20180328-ab-if-hurricane-maria-recap.pdf

Algorithm 3 Water distribution and flooding computation.

```
PROC FLOODING()
    VAL REAL64 outFactor = 0.8                          ▷ Water dispatched a proportion
    [MaxFanOut] REAL64 inTab, outTab                    ▷ Incoming and outgoing water
    REAL64 elevation                                    ▷ From geographic data
    CellPosition cellData                               ▷ Geolocalized data, image
    [MaxFanOut] cellPosition neighborhood               ▷ Neighbors' information
    [MaxFanOut] REAL64 deltaTab, quotaTab               ▷ What to dispatch on links
    REAL64 waterQuantity, rainFalling, waterOut
    [MaxTurn] REAL64 rainTab                            ▷ Flooding history
    SEQ
        waterQuantity := 0.0                            ▷ No water
        SEQ turns=0 FOR MaxTurn
            rainTab[turns] := 0.0                       ▷ Historical data
        SEQ turns=0 FOR MaxTurn
            SEQ
                rainFalling := rainFall[turns]          ▷ Water for this turn
                waterQuantity := waterQuantity × outFactor   ▷ Lost water locally
                waterQuantity := waterQuantity + rainFalling ▷ Rainfall
                waterOut := 0.0
                SEQ i=0 FOR (SIZE out)
                    SEQ
                        outTab[i] ! waterQuantity × quotaTab[i]
                        waterOut := waterOut + outTab[i]
                SEQ i=0 FOR (SIZE in)
                    inTab[i] := 0.0
                PAR
                    PAR   FOR FOR i=0 SIZE out
                        out[i] ! outTab [i]
                    PAR i=0 FOR (SIZE in)
                        in[i] ? inTab [i]
                SEQ i=0 FOR (SIZE in)
                    waterQuantity := waterQuantity + inTab[i]
                waterQuantity := waterQuantity - waterOut
                IF cellData[elevation] > 0.0
                    rainTab[turns] := waterQuantity
                ELSE
                    SEQ                                 ▷ Absorbs water
                        waterQuantity := 0.0
                        rainTab[turns] := waterQuantity
```

Figure 3.10 – Heavy flooding in Goudeloupe occurred during Maria hurricane (photo: BBC weather)

3.3.2 Second case study in Morlaix, France

Another case study is to predict flash flooding due to heavy rain during a storm in Monday, June 3, 2018 in the town of Morlaix, France [5]. The objective is to validate the accuracy of our proposed tool for flash flooding, especially for complex terrain areas.

Apart from data grid produced by QuickMap/NetGen/PickCell, to execute flooding simulation, it is necessary the rainfall values measured in every hour during a storm, hurricane, and so on as input data. The data can retrieve from the database of weather broadcast services such as meteofrance.com, meteoblue.com, etc.

3.3.3 Conclusion

Our proposed simulator is useful to predict risk places where flooding due to heavy rain may occur.

In fact, flood simulation utilized the frame tool flow as what was being used for radio signal propagation simulation and radio coverage prediction as explained in Section 3.2. For this study time step was one hour, we keep β at 20%, α_t rainfalls were taken from public websites shortly after the episode.

For backend process in physical simulations, the set of cells having the same characteristics (e.g. the communication range of a LoRa WSN in reality, the boundary of risk zone due to flash flooding) can be highlighted on the map by performing parallel convex hull algorithm as presented in Appendix D.

It leverages further intensive studies for reducing the risk of natural disasters, especially in complex topographical areas. It gives many benefits to the local resident's lives in terms of human aspects, as well as economic activities by informing of the risks of their position regarding exceptional events.

[5]http://www.letelegramme.fr/finistere/morlaix/inondations-a-morlaix-evolution-de-la-situation-en-direct-03-06-2018-11980811.php

Table 3.4 – Rainfall values were recorded for every hour in three days when Maria hurricane hit Guadeloupe in 18-20 September 2017

	Rainfall (mm)		
Time	Sept 18, 2017	Sept 19, 2017	Sept 20, 2017
00:00 to 01:00	0	0	0
01:00 to 02:00	0	0	11.1
02:00 to 03:00	0	0	9.4
03:00 to 04:00	0	10.5	0
04:00 to 05:00	0	7.7	0
05:00 to 06:00	0	5.6	0
06:00 to 07:00	0	6.8	0
07:00 to 08:00	0.7	8.9	0
08:00 to 09:00	0	11.5	0
09:00 to 10:00	0	12.4	0
10:00 to 11:00	0	11	0
11:00 to 12:00	0	8.9	3.2
12:00 to 13:00	0	6.6	3.2
13:00 to 14:00	2.1	5.2	3.4
14:00 to15:00	2.2	2.8	3.2
15:00 to16:00	2	1.9	2.9
16:00 to 17:00	1.7	1.8	0
17:00 to 18:00	1.9	0	0
18:00 to 19:00	2.1	0	0
19:00 to 20:00	2.8	0	0
20:00 to 21:00	0	3.6	0
21:00 to 22:00	0	4.4	0
22:00 to 23:00	0	4.8	0
23:00 to 00:00	0.5	0	0.8

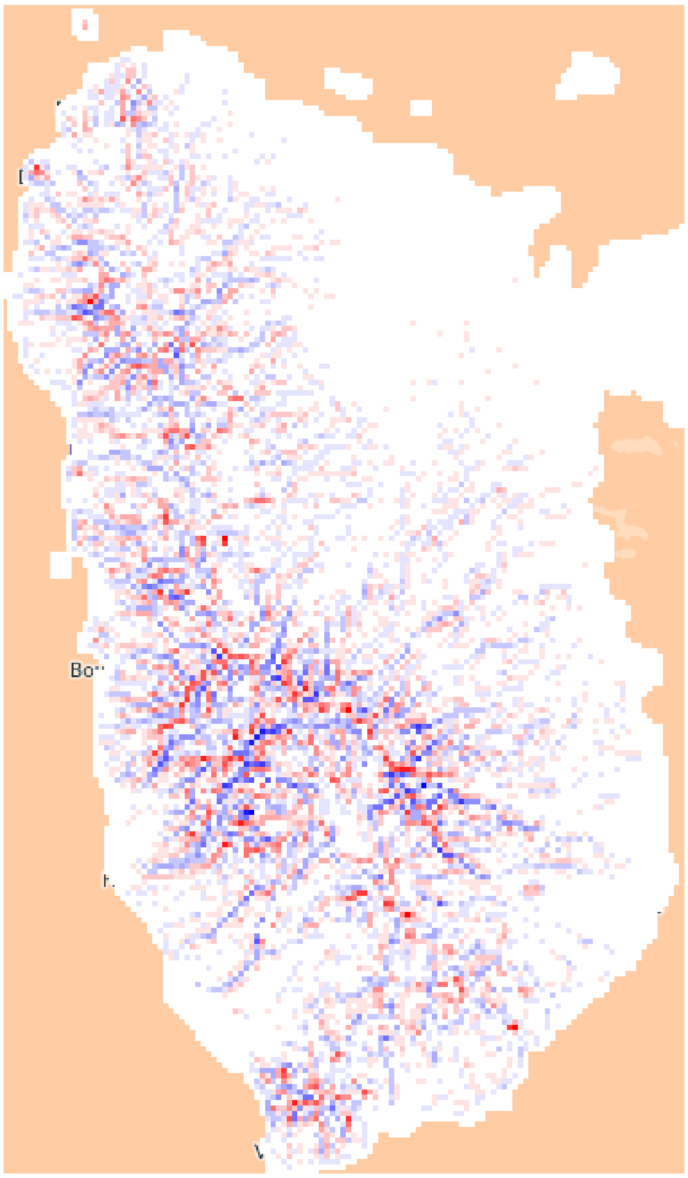


Figure 3.11 – Terrain complexity analysis using Topographic Position Index (TPI) for the island of Basse-Terre (Guadeloupe Archipelago). Blues lines indicate low-elevation zones such as valleys, mountain springs, rivers, etc. and red points associate with high-elevation places, mountains as an example. By the significant slopes due to reluctant of topography and vegetation, a large amount of rainwater rushes down to lower floors in a very short time leading to terrific flash flooding, landfall.

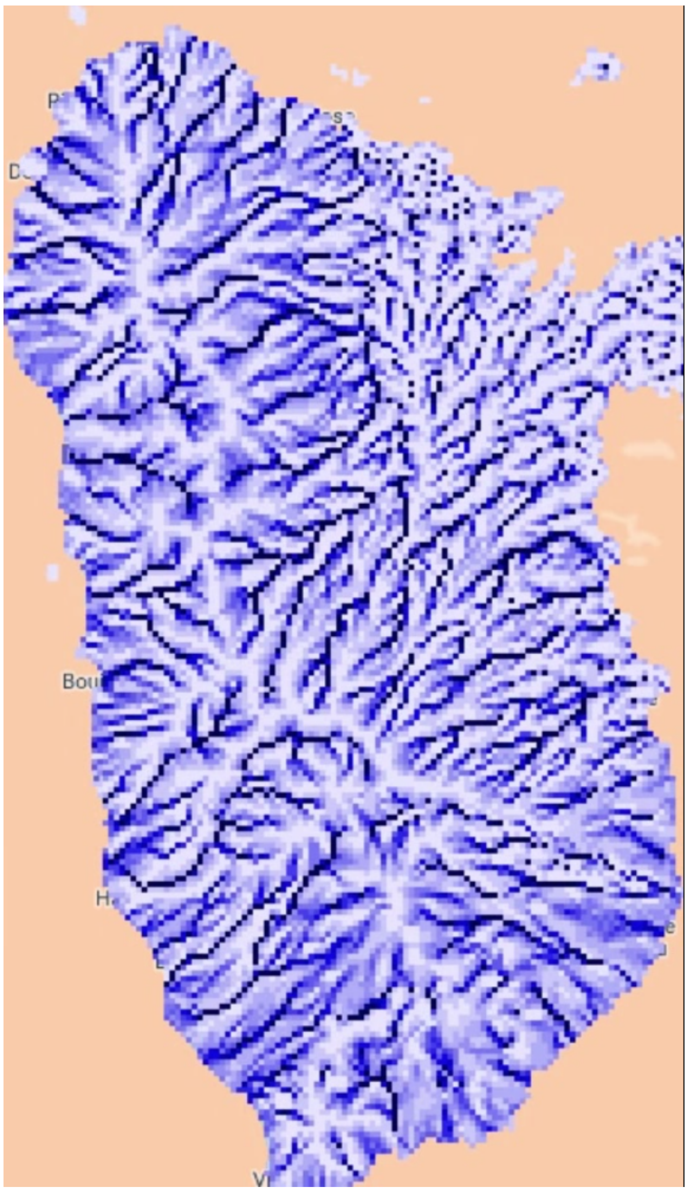


Figure 3.12 – A snapshot of flash flooding simulation for Guadeloupe during Maria hurricane in three days from 18 to 20 September 2017. The result shows the path of water and highlights risk places where suffering heavy flooding (refer to Fig 3.13 for the value of the color in millimeter of water level).

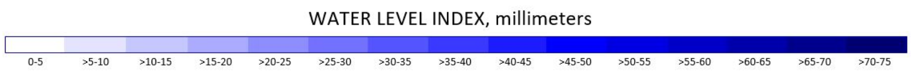


Figure 3.13 – A color scale being used to represent the intensity of flooding on map. The gradient of blue color is proportion with the constitute of water at each cell in uniform grid.

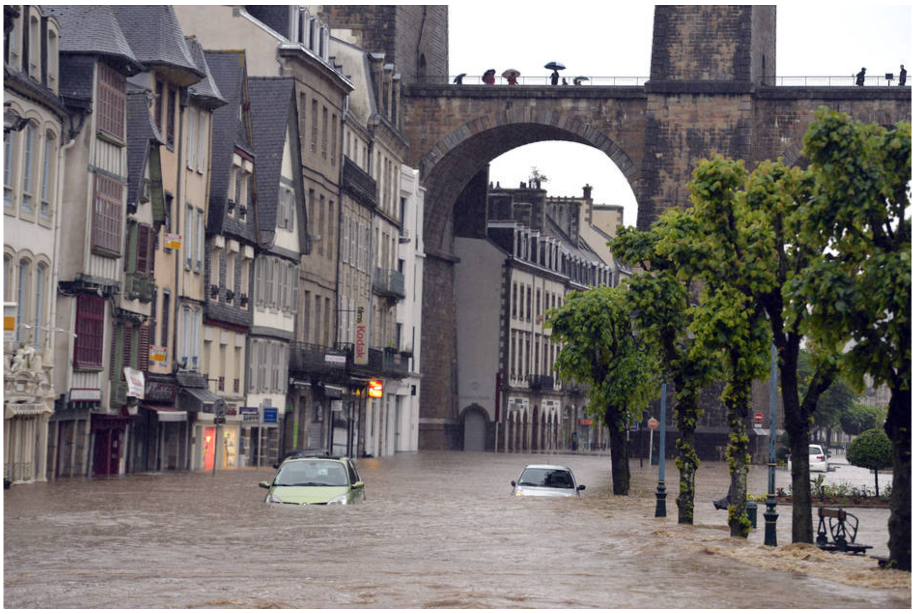
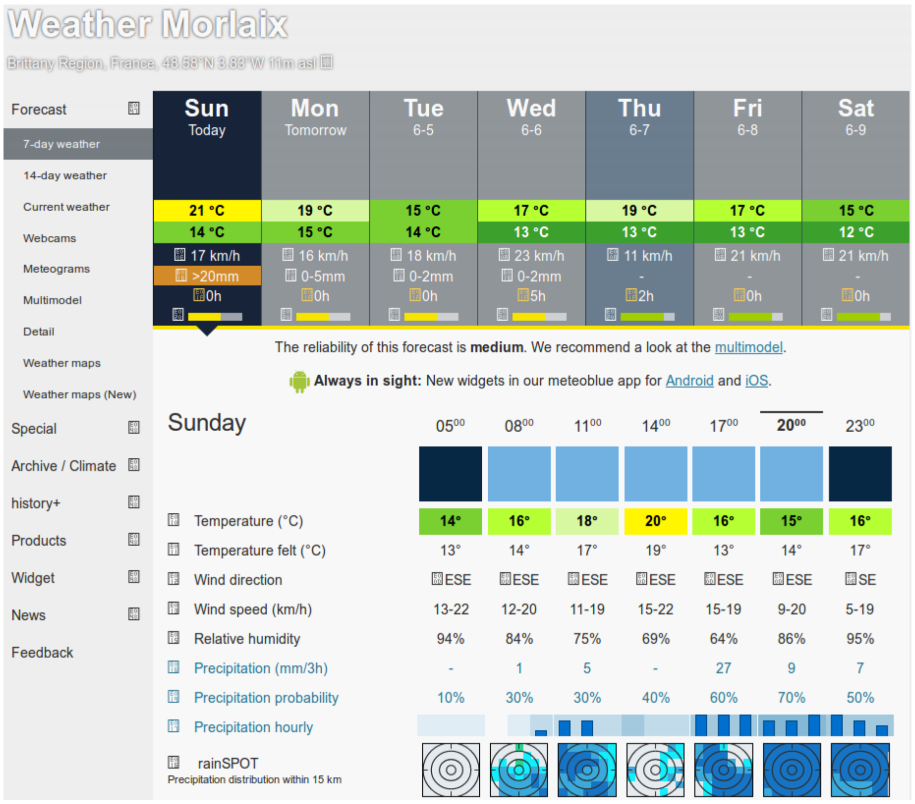


Figure 3.14 – Heavy flooding in the center of Morlaix, France due to a storm in June 3, 2018 (photo: Le Télégramme).



Weather Morlaix

Brittany Region, France, 48.58°N 3.83°W 11m asl

Sun Today	Mon Tomorrow	Tue 6-5	Wed 6-6	Thu 6-7	Fri 6-8	Sat 6-9
21 °C	19 °C	15 °C	17 °C	19 °C	17 °C	15 °C
14 °C	15 °C	14 °C	13 °C	13 °C	13 °C	12 °C
17 km/h	16 km/h	18 km/h	23 km/h	11 km/h	21 km/h	21 km/h
>20mm	0-5mm	0-2mm	0-2mm	-	-	-
0h	0h	0h	5h	2h	0h	0h

The reliability of this forecast is **medium**. We recommend a look at the multimodel.

Always in sight: New widgets in our meteoblue app for Android and iOS.

Sunday	05^{00}	08^{00}	11^{00}	14^{00}	17^{00}	20^{00}	23^{00}
Temperature (°C)	14°	16°	18°	20°	16°	15°	16°
Temperature felt (°C)	13°	14°	17°	19°	13°	14°	17°
Wind direction	ESE	ESE	ESE	ESE	ESE	ESE	SE
Wind speed (km/h)	13-22	12-20	11-19	15-22	15-19	9-20	5-19
Relative humidity	94%	84%	75%	69%	64%	86%	95%
Precipitation (mm/3h)	-	1	5	-	27	9	7
Precipitation probability	10%	30%	30%	40%	60%	70%	50%
Precipitation hourly							
rainSPOT Precipitation distribution within 15 km							

Figure 3.15 – The public weather broadcast service of Meteoblue including rainfalls for every hour in Sunday, June 3, 2018.

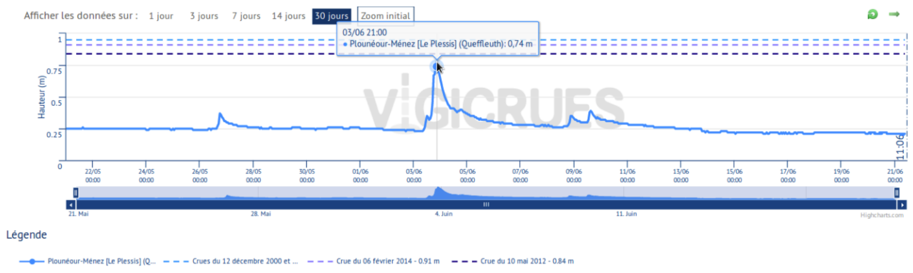


Figure 3.16 – The public weather broadcast service of Meteoblue including rainfalls for every hour in Sunday, June 3, 2018.

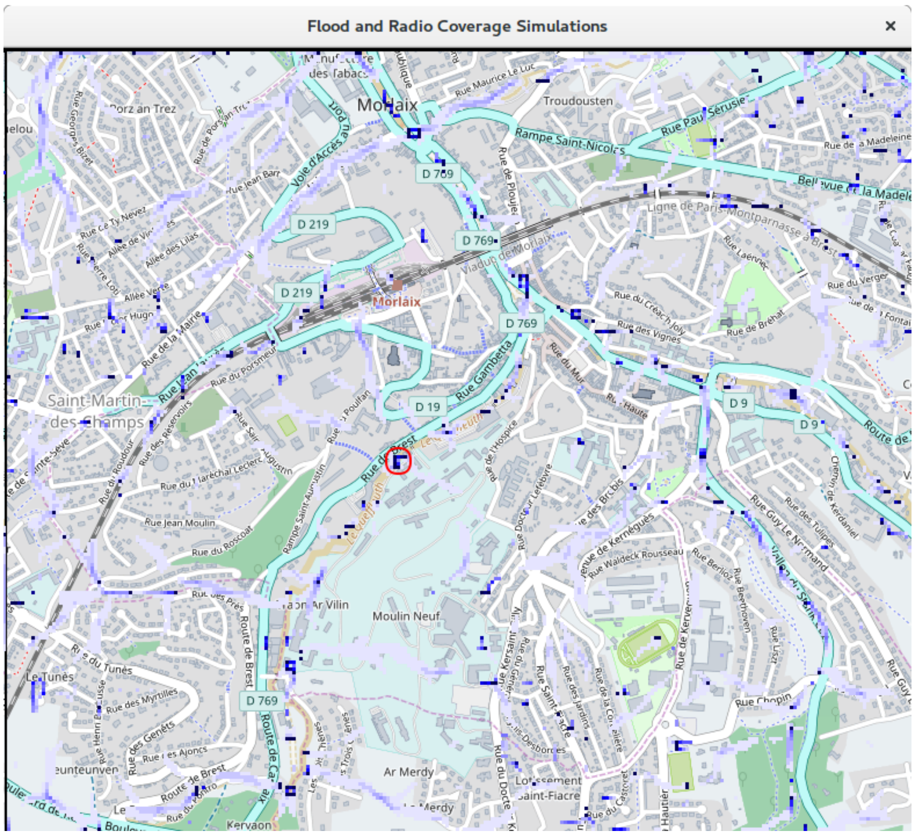


Figure 3.17 – Simulation result for Morlaix, obtained by executing massively parallel processes on GPUs. The experimental zone comprises of 58275 cells corresponding to actual area 3 $\times$ 3 km. A place of concern is indicated by a red circle. This is *rue de Brest* near the river *Queffleuth* (longitude: -3.8329839706421, latitude: 48.573767695437, elevation: 10.1 m) where water level went up to 60 cm according to *Le Télégramme* newspaper. The main flooding directions can be seen in light blue over the maps.

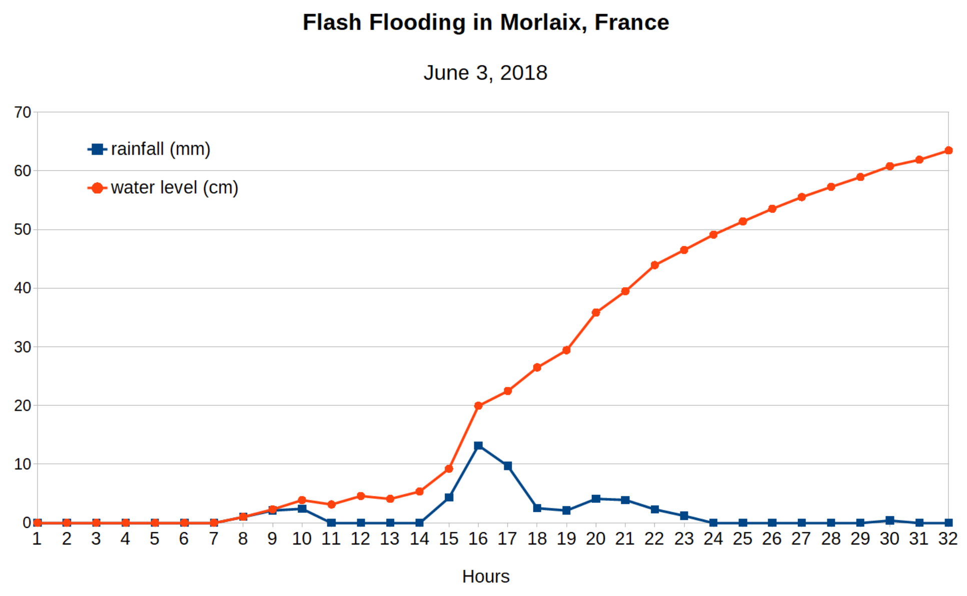


Figure 3.18 – Chart shows rainfall levels during 32 hours from 00:00 June 3, 2018, to 08:00 June 4, 2018 (blue line). The red line presents the level of water at the concerned place (as shown in Figure 3.17). It was noticed that serious flooding often comes several hours after a storm or heavy rain. After the rain stopped, it required a couple of hours or days to drain all water.

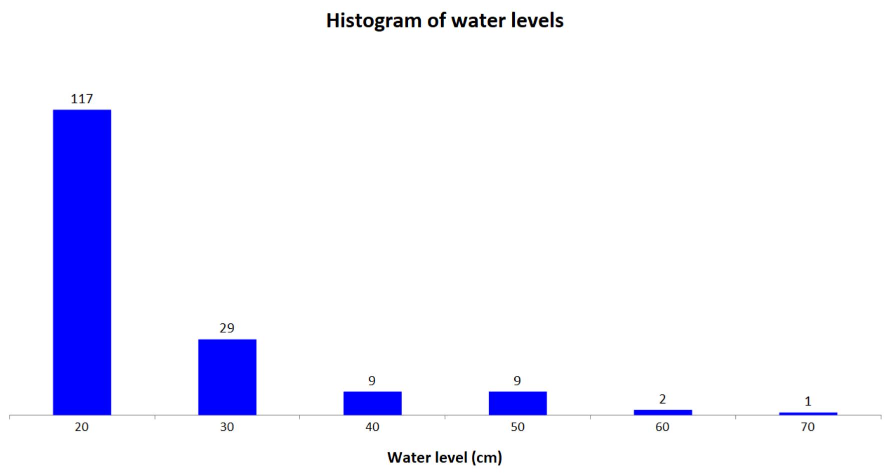


Figure 3.19 – Even though the heavy rain just occurred during a few hours (3:00 - 5:00 PM), simulation result shows that there are 50 points where water levels are from 30 to 70 cm.

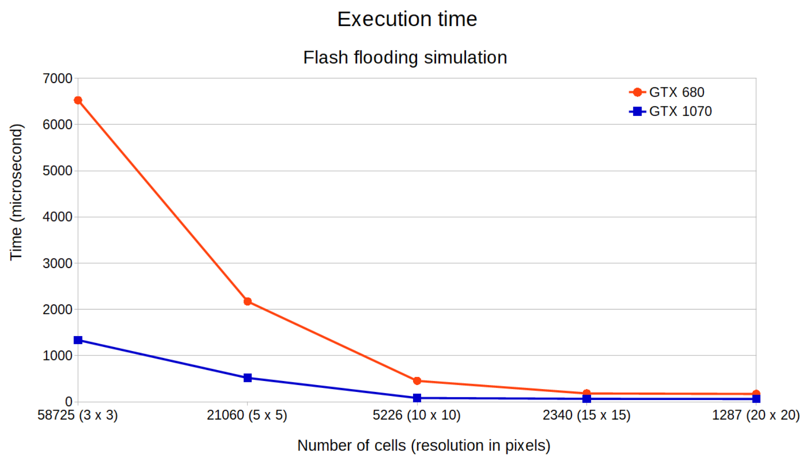


Figure 3.20 – Performances of flooding simulation on different GPU cards. The execution times increase corresponding to the number of cells which need to be processes.

4
Signal Propagation and Communication Modeling

(also in [1] pages 15-20)

The content of this chapter focuses on modeling of radio signal propagation and communication with LoRa as a study case.

Section 4.1 provides an overview of Semtech LoRa technology allowing low-power, long-range radio communications for environmental monitoring applications.

In Section 4.2, parallel algorithms for cellular long-range coverage computation is presented in detail.

Section 4.3 introduces about radio wave propagation models such as free space path loss, single knife-edge diffraction and Okumura-Hata model for signal propagation model taking into account the impact of transmission environment.

Section 4.4 describes the case studies in four typical different terrain areas in Brittany that were done for validation of LoRa coverage prediction based on cellular simulation approach.

4.1 Semtech's LoRa technology: a brief overview

In this section, we will describe how radio signals propagate, and LoRa radio protocol such as range, power, delays, and data rate.

4.1.1 Principles and issues

Radio signal propagation is a physical phenomenon where radio waves spread out from one point (the emitter) to other listening points (the receivers). Because a radio wave is a form of electromagnetic radiation it obeys to principles of physics such as reflection, refraction, diffraction, absorption, polarization, and scattering [142, 22, 143]. As a result, the quality of a received signal is affected noticeably by these physical phenomena corresponding to transmission conditions such as free space, or obstacles such as the shape of the ground, the existence of vegetation, or high buildings. To deploy a long-range wireless sensor network efficiently, it is also necessary to predict the coverage of each emitter, calling a *base station* (BS), and to estimate the *strength of received signal* (RSSI) for each receiving node locations (called *sensors*). Many studies have addressed the modeling of path loss such as free space path loss (FSPL), or Okumura-Hata model [144, 145, 146, 147]. The relation between general purpose radio coverage and geographic modeling by ray tracing was addressed by authors such as [148, 33, 149].

Another important topic is the way signals are modulated to represent information. Chirp Spread Spectrum (CSS) modulation is a key technique used in Semtech's long-range technology. CSS has been widely used, starting from radar applications since the 1940's. LoRa modulation uses the frequency of the carrier, applying continuous linear increases (up-chirp) and decreases (down-chirp) over time, while encoding information. This defines LoRa transmissions as a trade-off between low data rate for long-range communications, and low-power consumption, because CSS provides an ultra-long-range spread spectrum communication with high interference immunity consuming low energy. Consequently, the communication range can be reached up to several tens of kilometers for real sensing networks that can be deployed in areas having a high density of interference sources [150, 151, 152, 153, 154, 155].

These features of LoRa match requirements of remote sensing systems in which a small chunk of data produced by sensing devices is registered, then sent to the base station to feed decision systems. This collection takes place periodically aiming at preserving the lifetime of battery power supply in sensor nodes. According to datasheet specifications for LoRa transceivers, communication range and data rate are inversely proportional. LoRa transceivers work with center frequency (CF) in the range of 137 MHz to 1020 MHz (sub-1GHz), programmable in steps of 61 Hz. It makes LoRa well suited to various regions where the policies on *industrial, scientific and medical* (ISM) radio bands are different. There is a set of LoRa parameters consisting of bandwidth (BW), coding rate (CR) and spreading factor (SF) to define an appropriate trade-off depending on the priority order of applications such as transmission range, communication duration time, and energy consumption [156, 157, 158].

For each SF configuration symbol rate, a number of transmitted chips per symbol, can be computed as in[107, 159, 160] *Chips* are binary elements in CSS transmissions.

$$R_s = \frac{R_c}{2^{SF}} = (symbols/s) \tag{4.1}$$

As a derivative of CSS, LoRa chip rate equals the bandwidth of the signal

$$R_c = BW(chips/s) \tag{4.2}$$

as a result, the symbol rate is

$$R_s = \frac{BW}{2^{SF}}(symbol/s) \tag{4.3}$$

The modulation bit-rate is given by

$$R_m = SF \times \frac{BW}{2^{SF}}(bits/s) \tag{4.4}$$

Due to the inclusion of a forward error correction code that increases the robustness of radio link against interference bursts, the bit rate of LoRa becomes

$$R_b = Rm \times CR(bits/s) \tag{4.5}$$

where CR equals $\frac{4}{4+n}$ with n configurable values from 1 to 4.

4.1.2 LoRa: bandwidth

The values of the data rate in Table 4.1 are calculated by applying Equation 4.5. The possibility to modify the bandwidth enables a trading between *time on air* of messages for communication range (see Table 4.4 and Appendix C). In Table 4.1 LoRa provides ten different communication bandwidths from 7.8 KHz to 500 KHz, even though low bandwidth requires more accurate clock [107]. For this reason, there are only three frequencies 125 kHz, 250 kHz and 500 kHz effectively used in LoRaWAN protocol as proposed by LoRa Alliance [55].

Table 4.1 – LoRa signal bandwidth (BW) and data rate (Rb) with SF=12 and CR = 4/5.

BW (kHz)	Rb (bps)
7.8	18
10.4	24
15.6	37
20.8	49
31.2	73
41.7	98
62.5	146
125	293
250	586
500	1172

Table 4.2 – Coding rate options are supported by LoRa physical layer.

Coding rate	Overhead ratio
4/5	1.25
4/6	1.5
4/7	1.75
4/8	2

4.1.3 LoRa: Coding rate

LoRa employs Forward Error Correction (FEC) to improve the robustness to interference. Four options for coding rate can be selected for each radio link as shown in Table 4.2. Radio communications using smaller CR value offers more robustness, but increase the *time on air* of packets and energy expenditure due to redundant bits. In LoRa packet structure the header, which is encoded at 4/8, contains the CR value used for payload. For this reason, LoRa radios configured in the same central carrier frequency (CF), SF and BW but different CR can operate each other.

Table 4.3 – Spreading factor corresponding to number of chip must be sent for each symbol.

SF	chips/symbol
6	64
7	128
8	256
9	512
10	1024
11	2048
12	4096

Table 4.4 – Packet *time on air* as a function of spreading factor (SF) for a set of typical configuration: BW=500 kHz, CR=4/5, programmed preamble = 6 symbols, explicit header mode, CRC enable, payload length = 8 bytes, and center frequency= 868 MHz . System analysis in detail is discussed in Appendix C

SF	Time on air (ms)
7	8.51
8	17.02
9	28.93
10	57.86
11	115.71
12	231.42

The coding rate CR is used for *cyclic error coding* to detect and possibly correct the disruption of packets with errors. For example, coding rate 4/5 means that coder generates 5 bits of data composed of 4 useful data bits and 1 redundant bit for checking. This parameter is chosen according to the effect on the environment on propagation conditions: weather, the density of interference.

4.1.4 LoRa: Spreading factor

Another vital parameter for LoRa performance tuning is the spreading factor (SF). SF can be defined as a function of received signal-to-noise ratio (SNR), having an impact on data rate and having a significant effect on characteristics: *time on air* of packets, energy consumption as well as receiver's sensibility, communication range (see Table 4.4 for theoretical latencies). The number of *chips per symbol* is calculated as 2^{SF} (see Table 4.3).

As mentioned in the Semtech technical document [107], even though spreading factor can be selected from 6 to 12, SF=6 is a special case for the highest rate transmission. This setting requires special operating conditions such as a specific header. An advantage of using different SF is to establish virtual sub-channels to communication at the same time using the same center frequency without any collision because these modulated signal are orthogonal to each other.

4.2 Parallel algorithms for cellular long range coverage computations

The cellular approach is based on splitting an image (or abstract data representation) into a discrete regular cell system. When the physical space description is obtained, we are interested to compute reachable cells in the line-of-sight from emitting positions.

Each cell is represented by a process either in software (occam *thread* processes) or in hardware (processing elements in a graphics accelerator). Processes are connected by channels that allow to send or receive information from neighbor nodes according to a chosen neighborhood: (W, N, E, S) as an example, or a more complete *Moore* neighborhood with 8 nodes.

Simulation is achieved in a lock-step fashion by cycling on a synchronous parallel program: communicate with neighbors, observe local status (sensing), decide about a new status and prepare next cycle communications. This mechanism defines a physical machine as a distributed product of automata of any shape.

The line-of-sight represents a ray broadcast in any direction from an emitter. The ray propagation can be stopped by ground topology (hills, valley). Cellular simulation mimics the physical behavior by propagating the signal inside a tree rooted at the emitter cell and covering all the space in concentric circles. Each new step in the algorithm covers a new circle, and the computation finishes at $2log(n)$ steps where *n* is the number of cells. During ray propagation, the elevation ground profile is collected into *routes* that are completed progressively based on positions and elevations. Each cell can decide if the emitter is visible or not by comparing its elevation to the received profile.

This algorithm can be identified as a synchronous Breadth First Search [130], while radio propagation estimation follows technical rules described in [141] and [22].

4.2.1 General idea

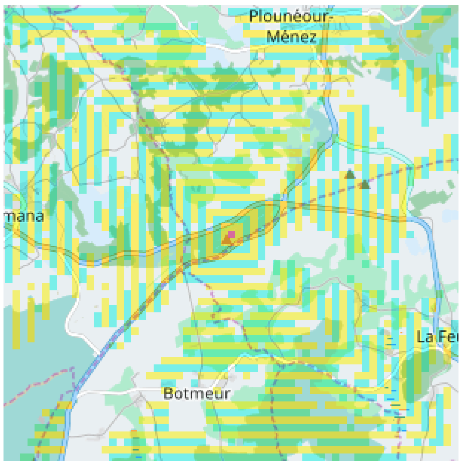


Figure 4.1 – Radio signals propagate in concentric squares step by step. Reachable cells are represented in colored stripes.

Once a geographic space has been selected, a concern is to compute reachable cells in the line-of-sight from an emitting position. According to section 2.2.4, we admit that each cell is represented by a process, and simulation is achieved by cycling on a synchronous parallel program: communication with neighbors, updating the local state, and preparing next cycle communications. The *line-of-sight* (LoS) is a ray broadcast in any direction from a root emitter. Propagation will be stopped or modified by ground obstacles such as hills, valley, etc. The simulation parallel algorithm mimics the physical behavior by propagating the signal inside a spanning tree rooted at the emitter cell, and covering progressively all the space in concentric "*circles*" as shown in Figure 4.2.

Each new step in the algorithm adds a new circle, and the computation finish in $2 \times log(n)$ steps where n is the number of cells. During ray propagation, the ground profile is collected into a route. A route profile is shown in Figure 4.2. Routes are completed progressively based on positions and elevations. Thus each cell can thus decide if the emitter is visible or not by comparing its slope to root with the previous slopes in the received profile, as shown in Algorithm 4.

4.2.2 Vertical model

Signal propagation progresses in steps according to the time. It builds progressively a route from the emitter to any other cell, as an array holding traversed locations and signal parameters (Figure 4.2). As it is the case for most of the cellular simulation algorithms, nodes start with a discovery stage, where geometric coordinates are exchanged to bind communication channels to cardinal directions. This is mandatory to deal with the irregularity of cell system

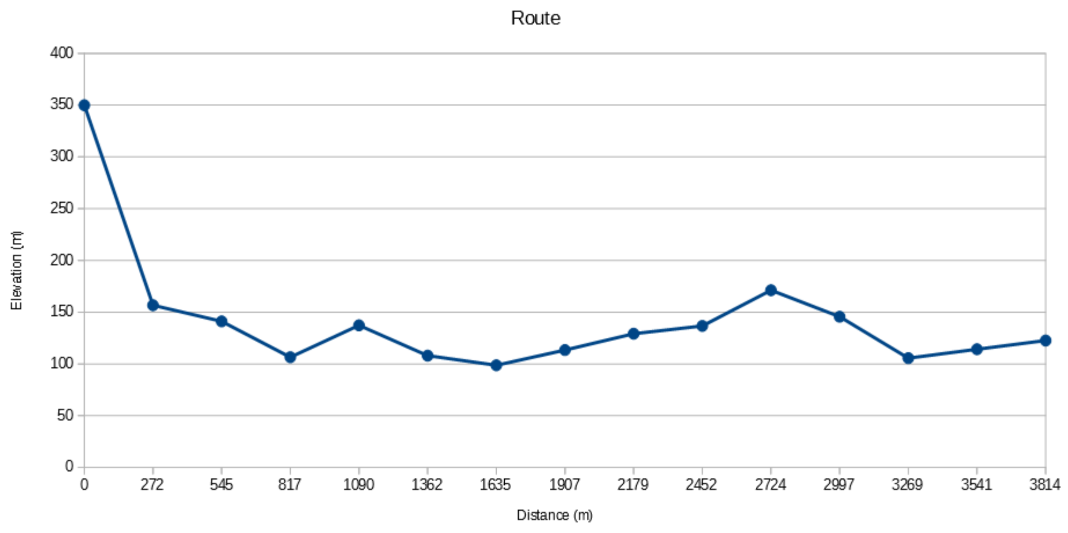


Figure 4.2 – A profile obtained along a route. Let's assume that an emitter located on the furthest left of the chart, with a distance of 0 m, an elevation of 350 m. According to LoS condition three points at distances 2997 m, 3269 m and 3541 m seem unable to receive the signal from the emitter.

shapes as shown in Figure 2.17, step (3), and also to maintain horizontal signal direction, as discussed in Section 4.2.3.

To start the building a propagation, an emitter has to initialize a tracing route with its own identity and geographic location. Other cells start their processes with an empty tracing route. Note that the tracing route, in this system, is the payload of the message for communications within the cell system. The root cell then sends its message to all neighbors via eight directions from the Moore neighborhood. Each cell accepts only one message then assigns the corresponding owner of the accepted message as its parent. This cell must send an acknowledgment message back to the parent cell to confirm the dependency relationship between them. The next step in the transition rule is to insert its own location into the tracing route, then to pack routes and spread out to neighbors. If *size* is the maximum between *width* and *height* then at most, *size* rounds are necessary to cover an entire zone.

4.2.3 Horizontal model and Directed Breadth-First Search

The original Breadth-First Search (BFS) algorithm builds a tree covering a whole network starting from a root position. For radio propagation, this tree would allow reaching any cell in a minimum number of steps, propagating a route to this cell without managing signal horizontal directions. In the case of radio signals, there is the necessity to guide the route horizontally to maintain the best approximation of a straight line. A sequential algorithm that reduces horizontal errors to a minimum was suggested by Bresenham to draw graphic lines on bitmaps [161]. In our case, it was preferred to find a parallel distributed approach based on Directed BFS (DBFS), that could also match particular geographic or electromagnetic dynamic considerations. Figure 4.3 shows DBFS routes resulting from this

Algorithm 4 Setup visible nodes from emitter based on the LoS condition.

```
procedure FSPL
    Input: $cell_i$ and its neighbors $cell_j$
    Output: $cell_j.visible$
    ____________________________________________
    for each $cell_j$ do
        if ($cell_j.slope <= cell_i.slope$) AND (receivedPower > sensibilityPower) then
            $cell_j.visible = TRUE$
        else
            $cell_j.visible = FALSE$
        end if
    end for
end procedure
```

algorithm according to an emitting node and some target receivers.

At the difference of a BFS, the cell transition function makes a decision on receipt of route messages based on the source position and its own source position. It adds its data to the route, and it forwards the route to neighbors of interest. The neighborhood was discovered at initialization stage, associating link indexes and cardinal directions.

4.3 Radio signal propagation on complex terrains

Radio propagation models are used to describe the qualifications and reliability of links using radio frequency. These models based on the physics of the diffusion of electromagnetic waves with respect to both constructive and destructive interference of environment. They can be practically applied to parameters inside routes transmitted from cell to cell, according to Section 4.4.

Free space path loss (FSPL) is the simplest model based on the Friis transmission equation. By assuming unobstructed along the transmission path, FSPL represents the line-of-sight decay of an electromagnetic wave as a function for distance only. This model exposes significant limits for complex terrain areas. Empirical path loss models were proposed to cope with the effects of turbulent terrains. An advantage is that these models attempt to estimate power loss as a function of distance and radio frequency taking into account the terrain heterogeneity as well as the effects of real transmission environment [22, 142].

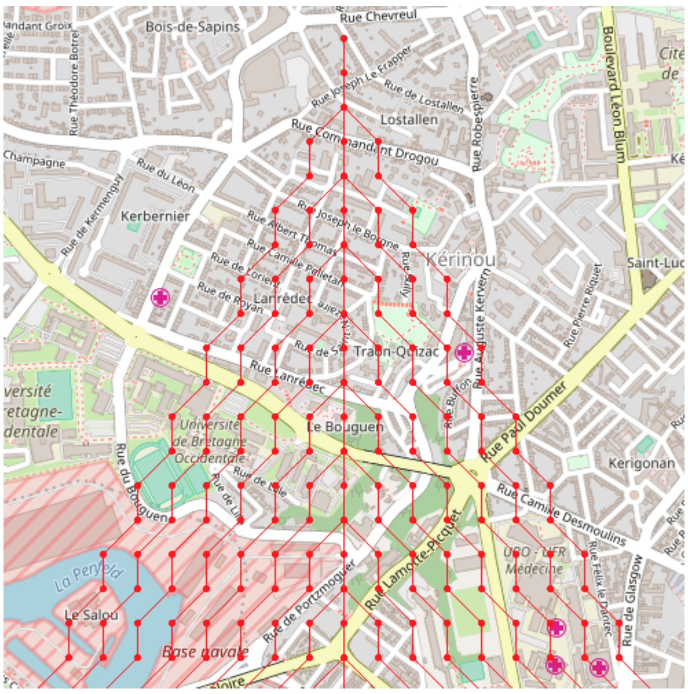


Figure 4.3 – Directed BFS is a distributed parallel algorithm being able to manipulate data on grid cells. Its aim is to mimic point-to-point radio links adopting LoS condition. Segmented lines laid on a map to represent how cells forward incoming signals gradually from a root cell (x=10, y=1) to definite directions.

4.3.1 Free space path loss model

In terms of radio communication, a *free space condition* is a region where there is no obstacle along the propagation path of radio waves. A radio signal is emitted by a transmitter and then it propagates in any direction at the speed of light. In this case, the signal energy can be received by an antenna in inverse proportion of the distance away from the signal source. In addition, the received power depends on the transmitted power, and gains of both transmit and receive antennas. Fading margin and other loss such as system loss, cables, connectors are also considered. The power of the signal is lost on the path is called the *Free Space Loss* represented by

$$FSPL = 20log_{10}(\frac{4\Pi d}{\lambda}) \tag{4.6}$$

Algorithm 5 is derived from Equation 4.6 in order to describe how to calculate the attenuation of signal power due to free space path loss. In consequence, the free space power received is given by equation 4.7, the Friis free space equation [22].

$$P_r(d) = \frac{P_t G_t G_r (\lambda)^2}{(4\pi)^2 d^2 L} \tag{4.7}$$

where

P_t: transmitted power,

$P_r(d)$: received power,

G_t: transmitter antenna gain,

G_r: receiver antenna gain,

d: distance between transmitter and receiver in meters,

L: system loss factor,

λ: wavelength in meters.

Algorithm 5 Received signal power in the Free Space Path Loss Model.

```
1: procedure SignalPower
2:     Input: txPower, waveLength, distance
3:     Output: receivedPower
4:     for each neighbor of $cell_i$ do
5:         if neighbor.visible then
6:             receivedPower ← txPower × (SQR(waveLen /(4.0 ×$valuePi$ ))× SQR(1/distance))
7:         end if
8:     end for
9: end procedure
```

4.3.2 Single knife-edge diffraction model

Diffraction appears around objects such as buildings, vegetation, etc. The losses caused by these obstacles can be described by a *Knife Edge Diffraction* model. This is considered as a function of the path difference around the obstacles and could be explained by Fresnel zones. Diffraction loss of signal after a knife edge is calculated as a function of the Fresnel parameter *v*.

$$v = h(\sqrt{\frac{2}{\lambda}(\frac{1}{t_t} + \frac{1}{d_r})}) \tag{4.8}$$

where *h* is the height of obstacle. d_t, d_r are the distances from the obstacle to the emitter and the receiver, respectively. Algorithm 6 demonstrates the computation of received power within a shadowing area because of a single knife edge obstacle.

$$L(v) = 6.9 + 20log(\sqrt{(v-0.1)^2+1} + v - 0.1) \tag{4.9}$$

Algorithm 6 Received signal power in the Single Knife-Edge Diffraction Model.

```
1: procedure DIFFRACTIONLOSS
2:     Input: fsplPower, h, $d_1$, $d_2$
3:     Output: diffPower
4:     v = h * SQRT((valuePi/2)$\times$((1/$d_1$) + (1/$d_2$)))
5:     if v $<$ 0 then
6:         diffLoss $\leftarrow$ 0
7:     else
8:         if v $<$ 2.4 then
9:             diffLoss $\leftarrow$ 6 + (9 $\times$ v) + (1.27 $\times$ SQR(v))
10:        else
11:            diffLoss $\leftarrow$ 13 + 20$\times$ $log_{10}(v)$
12:        end if
13:    end if
14:    diffPower = fsplPower - diffLoss
15: end procedure
```

In Algorithm 6, *receivedPower* is defines as the signal strength in dBm obtained at a receiver. In addition, *sensibilityPower* is the minimum value of signal strength in which condition a receiver can detect and demodulate a message successfully.

4.3.3 Okumura-Hata model

This is a dedicated radio propagation model to predict the path loss regarding geographic environments such as urban, suburban, and open areas [22, 142]. The formula for the Okumura-Hata model is given by

$$L = 69.55 + 26.16log_{10}(f) - 13.82log_{10}(h_b) - a(h_m) + (44.9 - 6.55log_{10}(h_b))log_{10}(d) - K \tag{4.10}$$

where

f_{MHz}: center carrier frequency of transmission band in MHz,

h_b: antenna height of base station in meter,

h_m: antenna height of mobile node in meter,

d_{km}: distance in kilometers,

$a(h_m)$ for each type of area and K see Table.

The model is valid for radios using carrier frequencies from 150 to 1500 MHz with an effective height of antennas from 30 m to 1000 m and distances ranging from 1 Km to 100 Km.

Table 4.5 – List of $a(h_m)$ and K values for each type of area [22].

Type of area	a(h_m)	K
Open		$4.78[log_{10}(f)]^2 - 18.33log_{10}(f) + 40.94$
Suburban	$[1.1log_{10}(f) - 0.7]h_m - [1.56log_{10}(f) - 0.8]$	$2[log_{10}(f/28)]^2 + 5.4$
Small city		0
Large city	$3.2[log_{10}(11.75h_m)]^2 + 4.97$	0

4.4 Four case studies on LoRa coverage prediction

The following case studies were investigated with the credible installation positions for *base stations* (BS) controlling hundreds of sensors. To evaluate the impact of transmission environment on reliability and robustness of radio links, several places with different terrains in Brittany, in France were selected for our intensive investigations. Okumura-Hata model was used for radio coverage computation in all following case studies.

4.4.1 Case study 1: Pont de Plougastel

This experiment shows how the radio signal propagates in good conditions without obstacles. The *base station* was installed on a bridge dominating Bay of Brest at a height of 30 m. A car was traveled with some stop to sample the emitted signal. In the order of 100 points were analyzed. The experiment shows that the radio is received at a long-distance but blocked by the shore obstacle in the south.

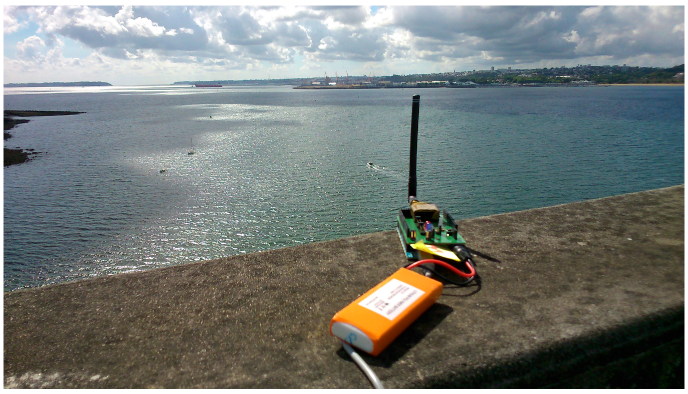


Figure 4.4 – LoRa emitter is placed on the Elorn bridge in our experimental measurement at Plougastel.

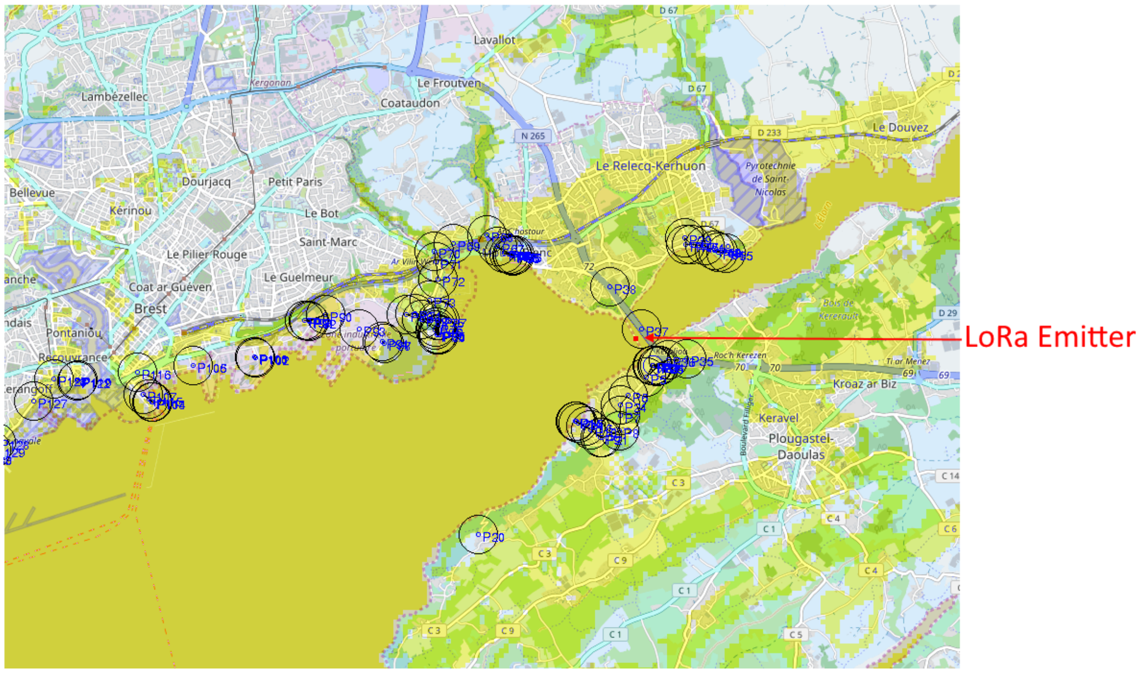


Figure 4.5 – A LoRa emitter was located on Elorn bridge (latitude: 48.386556, longitude: -4.399007, elevation: 30 m). Blue circles indicate points where mobile node (receiver) can receive the signal from the emitter. The coverage prediction by executing simulation are zones highlighted in yellow.

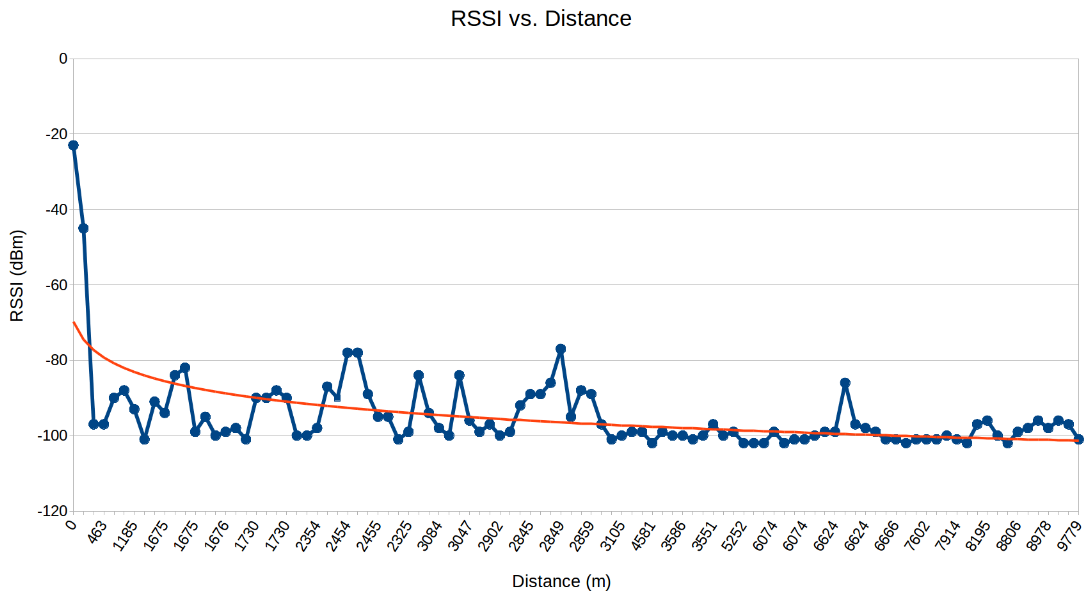


Figure 4.6 – RSSI values decay as a negative exponential function of distance. Obstacles seriously impact on the quality of radio waves.

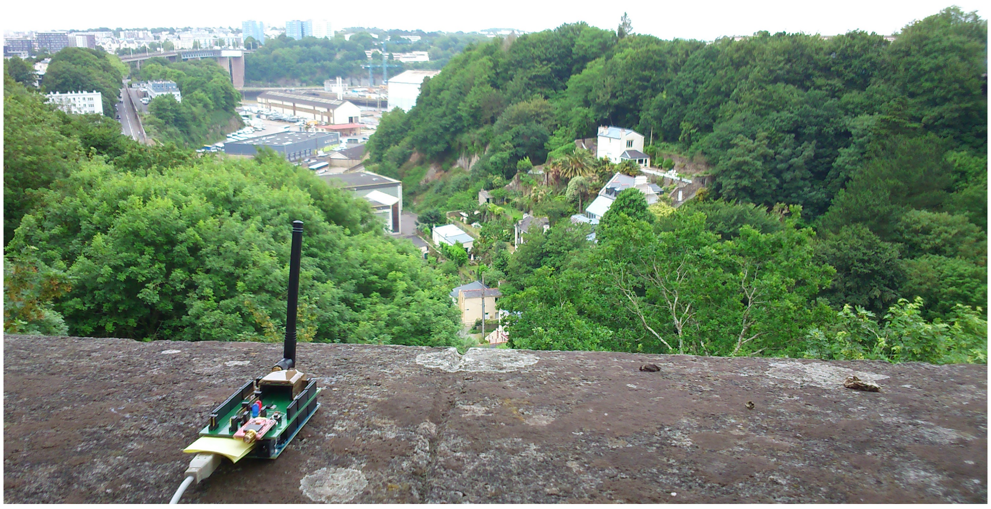
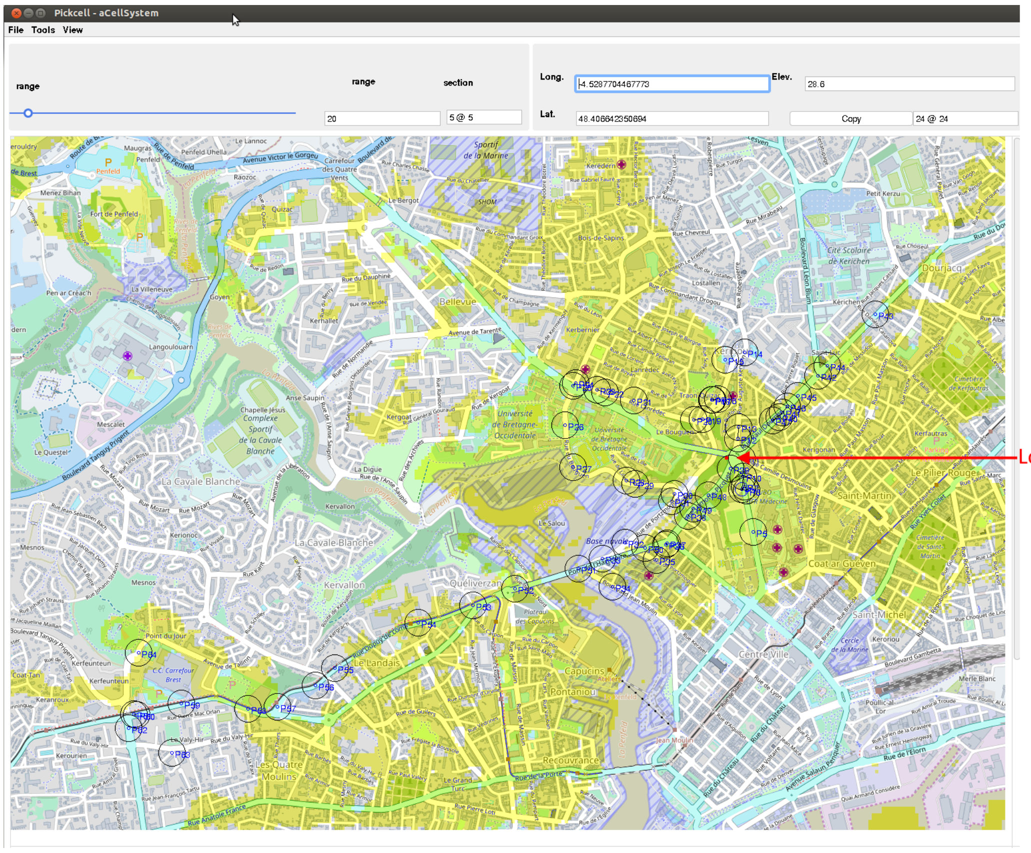


Figure 4.7 – A LoRa emitter was placed on top of a hill at Albert1^{er}, Brest city, France.

4.4.2 Case study 2: Place Albert 1^{er}

Contrasting with the case 4.4, these measures were taken inside the city that has a very complex topology with valleys and heights of 100 m (Figure 4.7). Valleys were created by an ancient river flowing toward the north and were invaded by the sea.



range

range 20

section 5@5

Long. 4.5287704467773

Elev. 28.6

Lat. 48.406642350694

24@24

Figure 4.8 – The set of blue circles locate points where a receiver can establish a radio link with a LoRa emitter (lat:48.397652, lon: -4.489541, elev: 52 m) in our real measurement, plus coverage prediction of the emitter in Albert 1^{er}, Brest city.

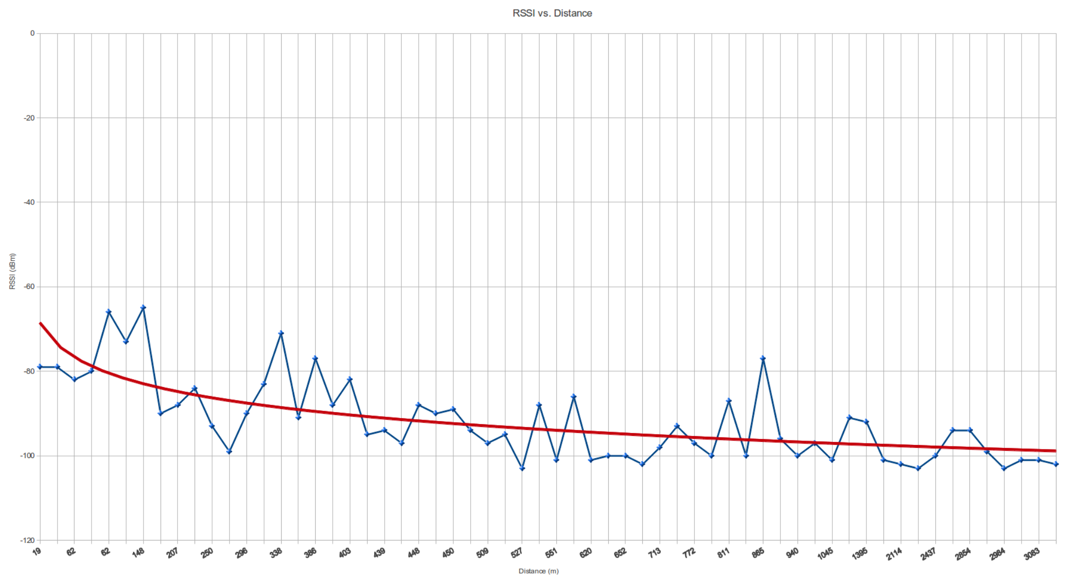


Figure 4.9 – Obtained Received Signal Strength Indicator (RSSI) values for experiment in Albert. This is an urban area with many buildings plus hills, valleys so that radio waves are locked by these obstacles.

A place closed to the Science Faculty was selected that dominates a valley leading to the Penfeld river (Figure 4.8). The car was circulated in the north and south part of the city to evaluate a line-of-sight simulated prediction presented Section 4.3.1.

The simulation was found to be pessimistic with an 80% accuracy on reached points, and some cells reached outside the predicted coverage.

4.4.3 Case study 3: Roc'h Trevezel

This third case have required a visit to *Arrée Mountains* in central Brittany. These mounts culminate at less than 400 m and are the site chosen for TV/Radio antenna.

The car was driven in a loop, up to 10 km showing a good accuracy of simulation prediction.

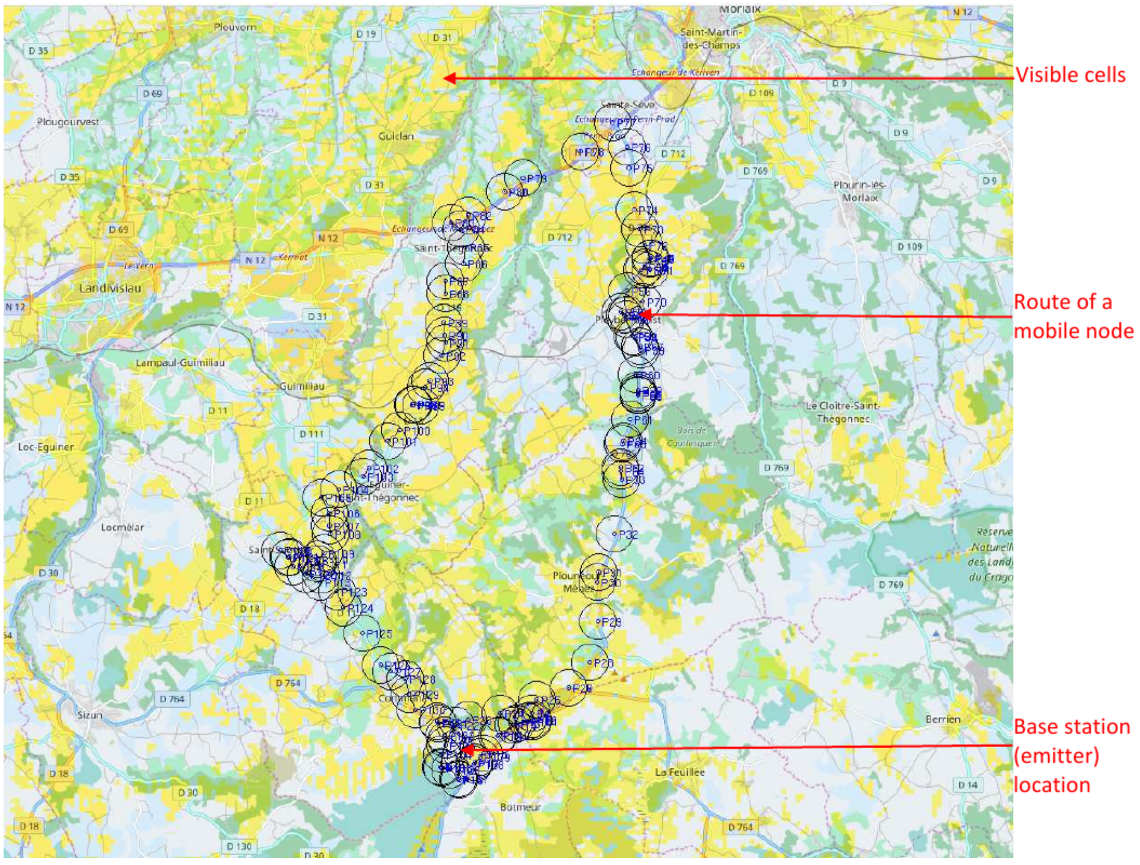


Figure 4.10 – The base station (emitter) is located on top of the mountain (latitude: 48.4051306, longitude: -3.9077762, elevation: 345 m). Another RECoco board is placed on a car (receiver) traveling around the area with many hills, valleys, and big trees. From simulation results, places (cells) are able to receive the radio signal highlighted in pink color. Blue circles show points where the receiver successfully got messages in our real measurement.

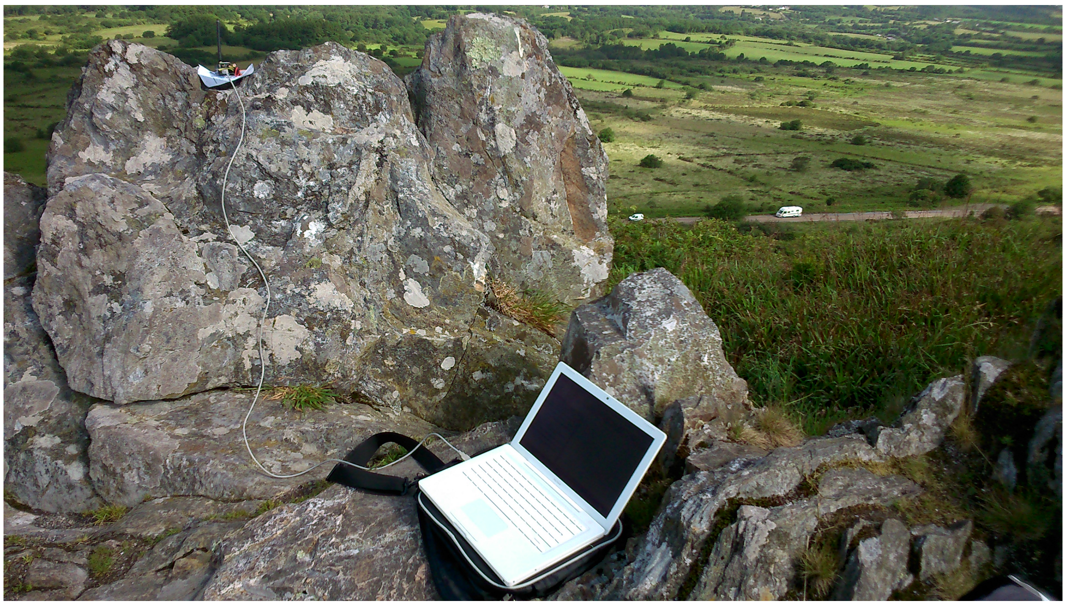


Figure 4.11 – A base station is equipped a RECoco board (emitter) E.0.1 and a MacBook. This base station was deployed on the top of the Roc'h Trevezel mountain, France. The receiver is another RECoco board mounted on a car moving around.

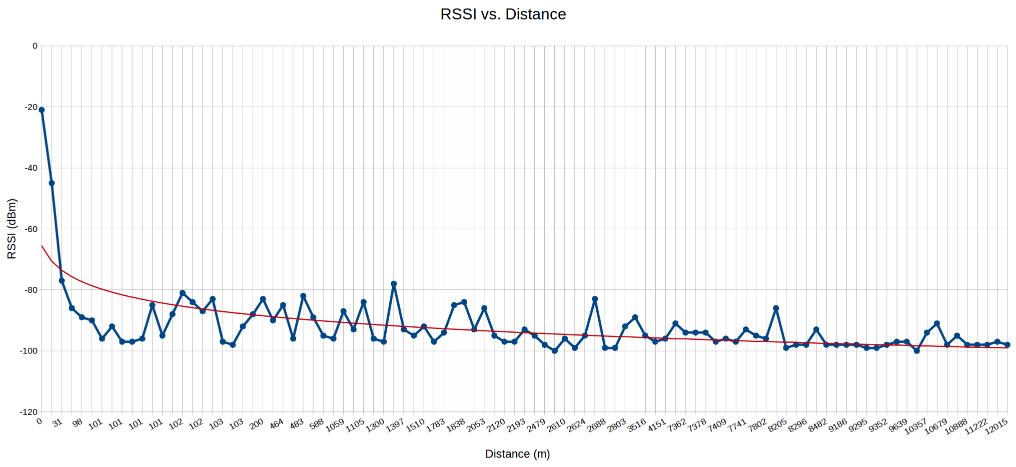


Figure 4.12 – Obtained RSSI values for experiment in the Roc'h Trevezel. The complex terrains in this area strongly impact on quality of radio links.

4.4.4 Case study 4: Brest harbor

This experiment was done at a high point near Brest railway station and harbor. This is a flat zone, but there are some buildings and big cargo containers. Besides, this zone is also crowded with traffic.

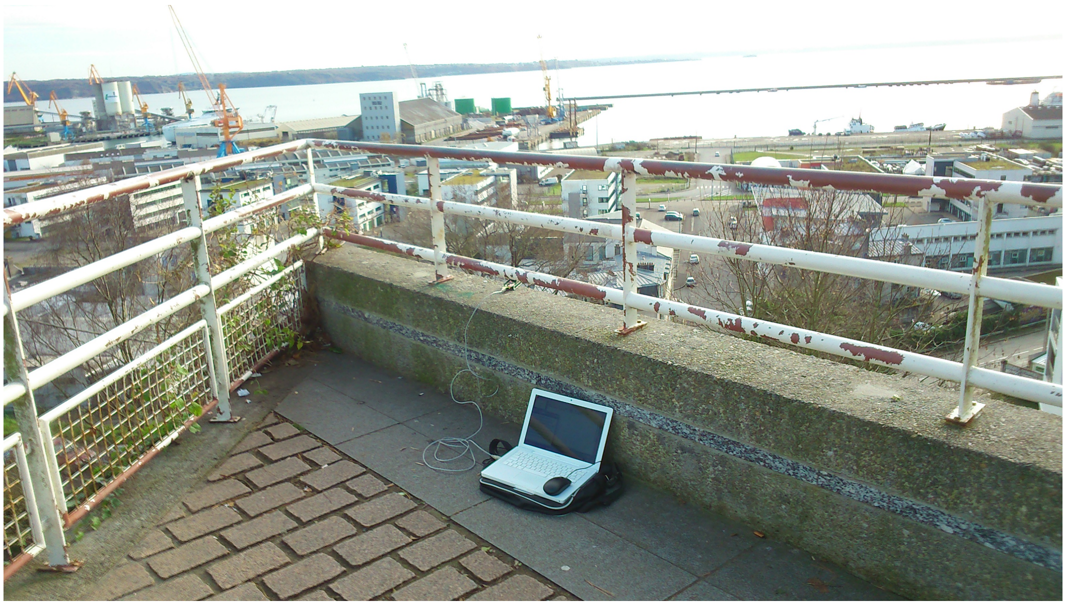
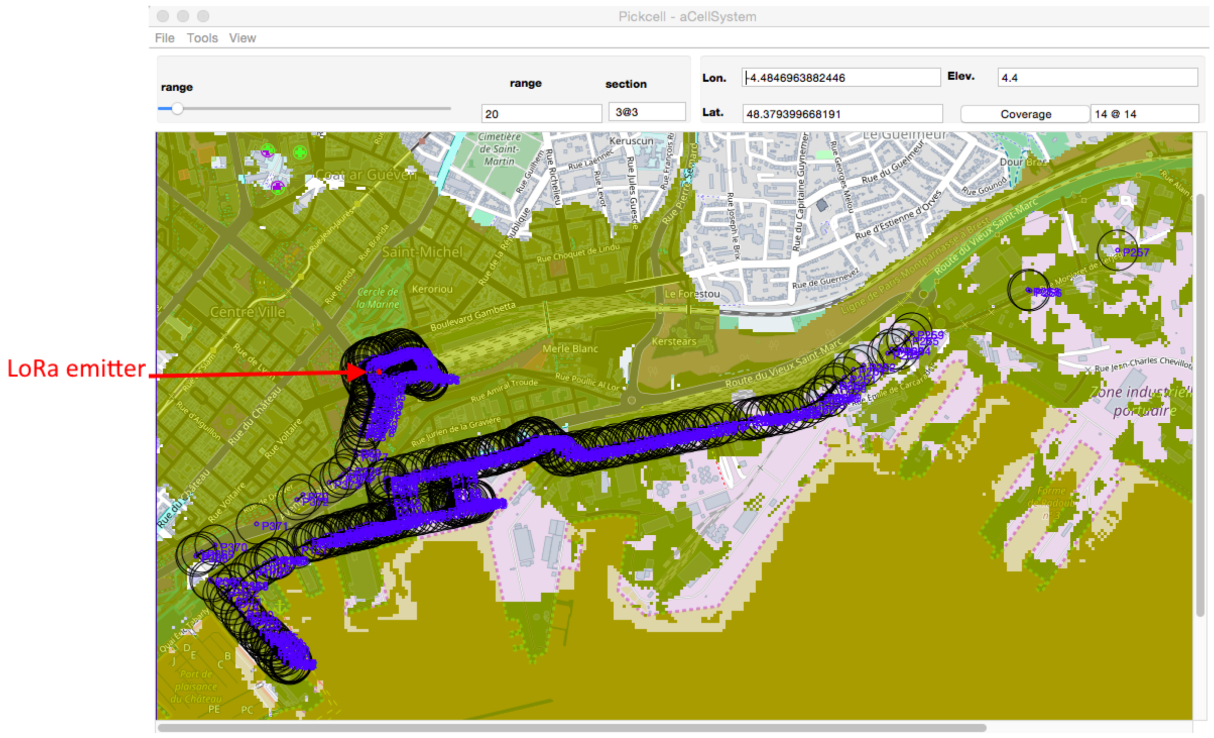


Figure 4.13 – Photo of base station was used in our experiment. It is at a high point (lat: 48.387479, lon: -4.480088, elevation 39 m) near railway station and Brest harbor.



range

range 20

section 3@3

Lon. -4.4846963882446

Elev. 4.4

Lat. 48.379399668191

Coverage 14 @ 14

Figure 4.14 – Simulation result for communication coverage of an emitter at Brest harbor.

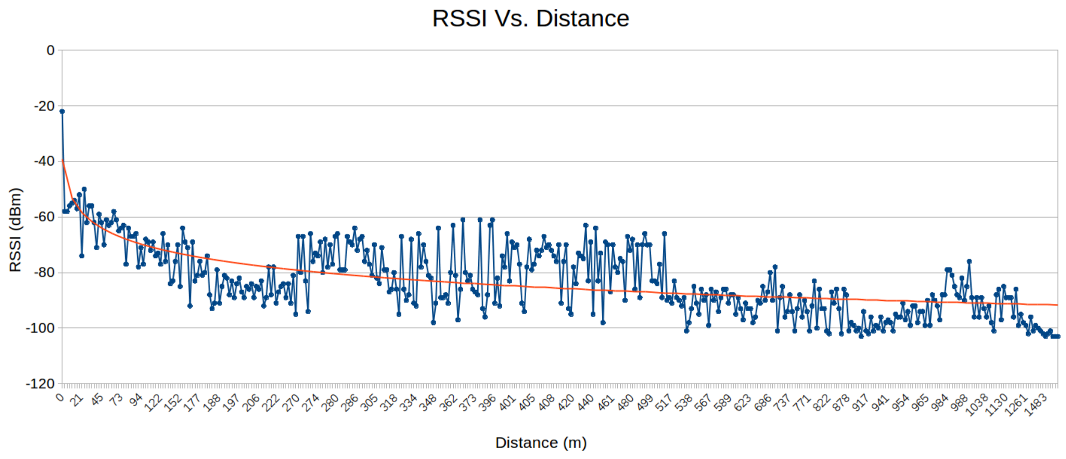


Figure 4.15 – Chart shows captured RSSI values as a function of distance. As can be seen in Figure E.7, some places the strength of received signals went down irregularly due to obstacles such as high buildings, trees, cars, etc.

4.5 Revendications

Appendix F and E present how to develop hardware and software tool supporting for experimental measurements of radio coverage predictions in different complex terrains.

In summary, radio waves blocked by obstacles and propagation loss in proportion with a point-to-point distance need to be concerned to optimize communication ranges. Terrains to be covered can be classified and partitioned as smooth, or rough according to metrics (Section 2.1). In smooth terrain areas, radio transmission can be considered as a near line-of-sight (NLoS). As a result, the issue of topologies can be almost neglected and the communication distance is expressed in relation to power reduction along the path. The free space path loss (FSPL) equation was proposed for this aim (Section 4.3). For example, using LoRa technology for a point-to-point connection in NLoS conditions, the distance for transceiving data is expected to reach hundreds of kilometers [163].

The experiments validate that line-of-sight (LoS) algorithm is more reliable for smooth terrain area. It goes down in proportion to terrain turbulence. Even though most of our measurements in a variety of topographies, the accuracy of simulation results is approximately 80 percent.

4.5.1 Multi-scale data space

The NetGen tool automatically produces data for simulations. Geo-location (latitude, longitude) plus elevation is produced for every cell in a regular grid. Data presented as a grid is fit to manipulate by processes based on cellular technology being speed up by performing on multi-cores CPUs or GPUs. This intrinsic errors of data inherited from SRTM-30 data sources, like other spatial data sets, are exclusive and constitute uncertainly [112].

Gridded SRTM-30 provides a discrete representation of continuous terrain surface. With the resolution 30 $\times$ 30 meters in the grid, the elevation on the ground may vary and based on sampling scheme. Sampling scheme refers either the technique to capture elevation values directly (using handled GPS, photometric as LiDAR, so on) or the algorithms for filtering and interpolating processes. If the sampling scheme is a grid cell is inappropriate, the resulting grid may be prone to serious errors [164, 110].

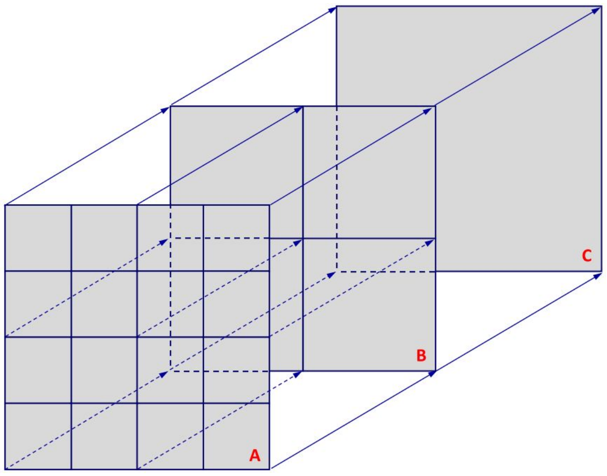


Figure 4.16 – Data spaces with different scales increasing by two are produced by filtering points alternately from the higher resolution data source. For example, data space (A) is in resolution 5 $\times$ 5 pixels. After filtering alternately points in both dimensions data space (B) is produced in 10 $\times$ 10 pixels. By applying the same process on data (B), data space (C) with resolution 20 $\times$ 20 pixels is carried out.

To evaluate the impact of changing resolution on the accuracy of coverage computation, we do statistics on several input data sets corresponding to various resolutions of cell pixels. First, a new data space with half the resolution in both width and height is produced by filtering alternative points of the higher resolution data set (see Figure 4.16). Following the same way, several data sets are prepared for our analyses.

In Figure 4.18, a bar chart for each experimental analysis, using cell resolution 5 and 10 pixels for the Roc'h Trevezel as an example, is comprised different percentage errors from 0% to 100%. In our analyses, there are five percentage error elements because the spaces are divided by two to obtain different cell sizes. 0% means that the computation results are exactly matched for two data grids with different resolutions. Otherwise, 100% refers two computation results are totally mismatched together. The other percentage errors can be analyzed in the same way.

The chart in Figure 4.18 shows the percentage error elements for each experiment. Additionally, using higher fine-grained grid data gives more accurate results. Although, it is necessary to compromise between reliability and workload of processes respect to phenomena which need to be observed.

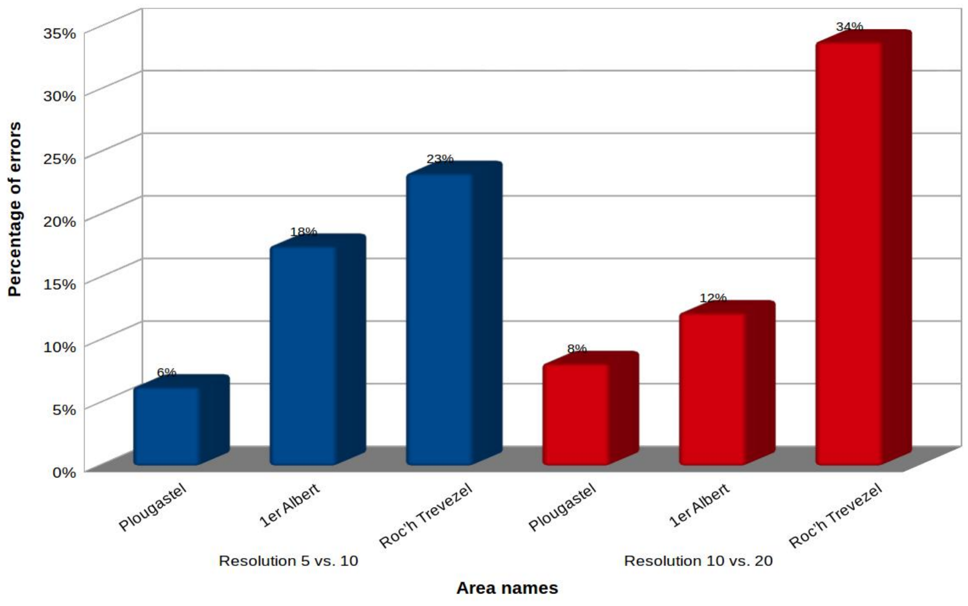


Figure 4.17 – This chart illustrates the mean of potential error ratios in LoS computations for various cell grain taking into account of terrain complexity. Percentage of potential errors are proportion with the resolution in pixels of segmented data. Apart from this, the error values for a given resolution is in proportion with data resolution. Moreover, the correctness of LoS computations is strongly depended on the terrain complexity of experimental zone.

The overall accuracy of the approximation in a segmented system is described by the maximum error causing at any cell. For this reason, the potential error components will be discussed in the previous section, both through theoretical analysis and empirical study. However, in this work such errors of the SRTM-30 source can be neglected, we just focus on the errors could be occurred when changing the resolution of the cell system. The accuracy of LoS algorithm for coverage prediction with different scales are shown in Figure 4.16. With the same resolution, the accuracy of computation is greatest in flat terrain and decreases in steep terrain. From these theoretical analyses, it can be concluded that segmented data should be in higher resolution for complex terrain areas than other areas in smooth areas. For data space generated by PickCell tool, a regular grid system, data size is an exponential function of the resolution of cell in pixels. Thus we must pay for higher accuracy by losing the impressive performance of massively parallel processing. Our suggestion is that the best compromise must be achieved between the accuracy of results and performance of processing. The user must be responsible for knowing what scale is reasonable for their analysis which should be matched with the physical objects and relevant for the phenomenon being analyzed to improving the accuracy of results.

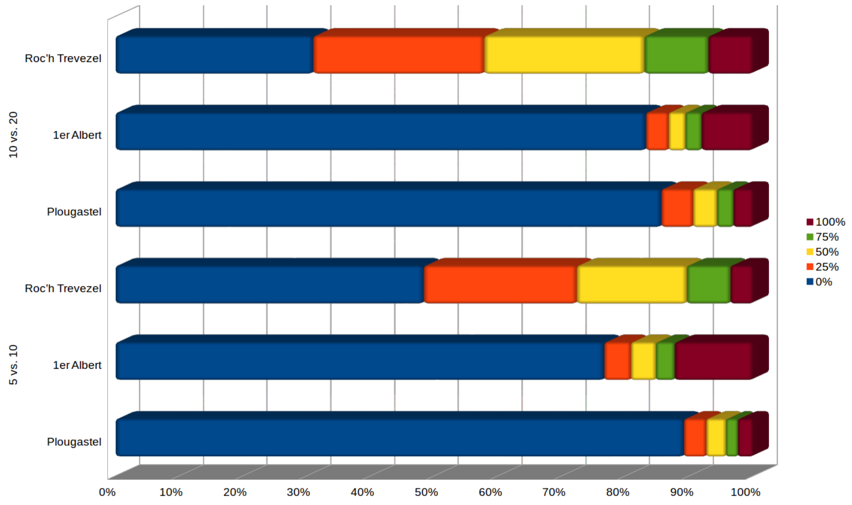


Figure 4.18 – Two statistical experiments with three data space with resolution 5, 10 and 20 in pixels at three different topographic terrains. Plougastel area is a river and its banks, a smooth terrain surface. Albert is an urban area with hills and valleys and Roc'h Trevezel is a mountain area, with turbulent topographic terrains. The mismatch of computation results due to rescaling are evaluated in percentages number of cells.

4.5.2 Latency in long-range communications based on LoRa technology

In a wide area network based on LoRa technology, the communication distance is in inverse proportion to data rate with each configuration of critical parameters (BW, SF, CR) for LoRa chip. Each node in a sensor network is distributed in a large area, probably having a complex geography. Due to this reason, for the effective performance of each node could be done by optimizing parameters regarding both communication and energy issues. Generally, for a node at a short distance, LoRa can be configured to operate in lower output transmission power and a higher data rate. This mode allows reducing the time for sending data and extend battery life. This leads to decrease the waiting for the other nodes and lifetime. Otherwise, with nodes located at a long-distance, it must be increased output power to the antenna and perform sending data at the lower data rate.

Even though, because of operating with very power LoRa modulated signals suffer serious impacts from transmission environment. According to the technical document from Semtech [107], radio signal propagation in LoRa technology is obeyed line-of-sight condition, so that it is critical to concern about relative positions among nodes in comparison within surround area. Adopting LoS constraint, one node can receive a signal if and only if that signal is not locked by a neighbor node right in front of it. In our work, it was done by mean of tracing profiles which were produced by accumulating all points along the routes during processes. At each step, the tracing profile is used to determine a node can be reachable from a root (an emitter).

By regarding the geographic complexity, the performance of LoRa networks is optimized in terms of time delays for communications and power consumption.

4.5.3 Quality of communications in LoRa networks

The quality of radio communication is qualified based on strength of the signal in Radio Signal Strength Indicator (RSSI). These values are measured at the receivers for every message. In the LoRa network, the receiver can receive and demodulation messages down to -149 dBm by utilizing CSS [159]. During operation of the network, if a node received three successive messages with RSSI low nearly -149 dB, it needs to change the parameters of configuration towards improving the quality of transmission (called Adaptive Data Rate, ADR) [107].

5 Parallel Algorithms for Simulations

The motivation of this study is to propose an effective algorithm which is well suited with parallel processing along deterministic routes in a uniform grid of cells, namely Directed Breadth-First Search (DBFS). Our proposed line-of-sight (LoS) algorithm, which attempts to mimic how to propagate radio waves in the rule of Friis model, is a particular application case of DBFS. The major benefit of this algorithm is to enhance the performance of routing computation by utilizing parallel distributed processing. The comparison in terms of execution is that DBFS needs $n \times n$ processes to explore a space with n cells while Bresenham's straight line algorithm requires $n \times n \times n$ for the same space.

First, Section 5.1 introduces distributed Breadth-First Search (BFS) algorithm. Next, Section 5.2 explains in detail about distributed DBFS algorithm, and then in Section 5.3 three distributed algorithms to optimize the communication coverage of wireless sensor networks over complex terrain areas are presented.

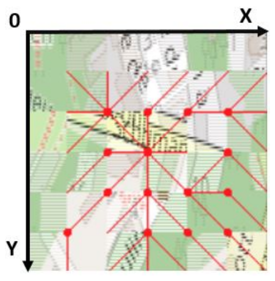
5.1 Breadth-First Search and Directed Breadth-First Search

5.1.1 Basic Breadth-First Search

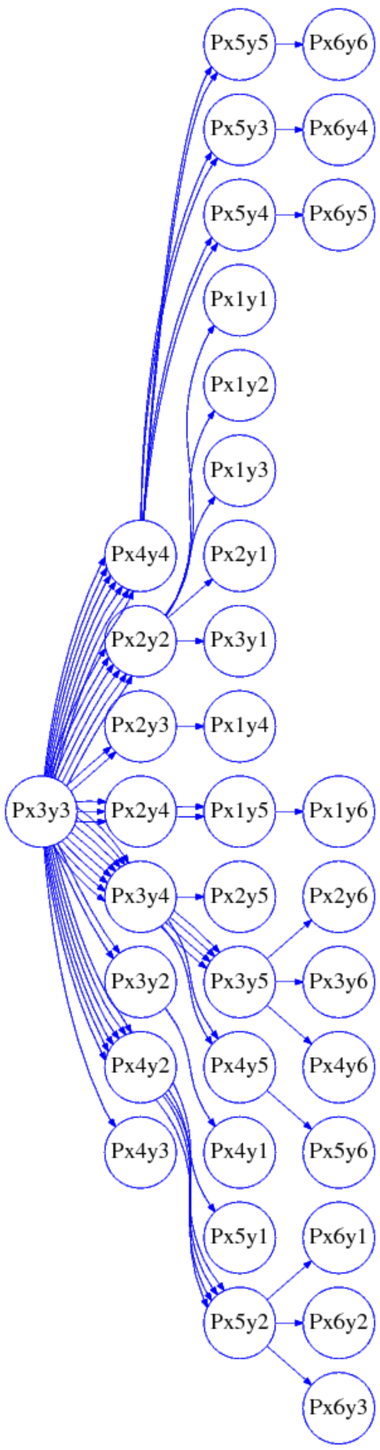
A nondeterministic routing algorithm taking into account geographic topology which is a derived from distributed Breadth-First Search (BFS) algorithm [130]. In this case, *nondeterministic* term refers to the unpredictable behavior of the cell in accepting only one value of simultaneous oncoming messages. Nondeterministic routing is a suitable model to describe intrinsic characteristics and evolution of physical phenomenon in nature such as gas or pollution spreading out in air or water.

Algorithm 7 Breadth-First Search Algorithm

```
PROC BFS
    IF isRoot                                           ▷ Initialize local values
        SEQ
            state := gotSearch
            initRoute (myTracePoints)
            addInRoute (mytracePoints, thisPosition)
    ELSE
        SEQ
            state := gotSearch
            initRoute (myTracePoints)
    SEQ turns=0 FOR MaxNodes - 1                        ▷ Main loop
        SEQ
            PAR                                         ▷ Communicate M to N
                IF state = gotSearch
                    PAR i=0 FOR SIZE out
                        out[i] ! points; outMsg[i]
                ELSE
                    PAR i=0 FOR SIZE out
                        out[i] ! null; nullByte
                PAR i=0 FOR SIZE in
                    in [i] ? CASE
                      null ; nullByte
                        SKIP
                      beParent ; nullByte
                        addInTabChildren (aChild, i)
                      points ; inMsg[i]
                        SKIP
                IF state = gotSearch                    ▷ Update local values
                    state := forwardedSearch
                ELSE
                    SKIP
```



(a)



(b)

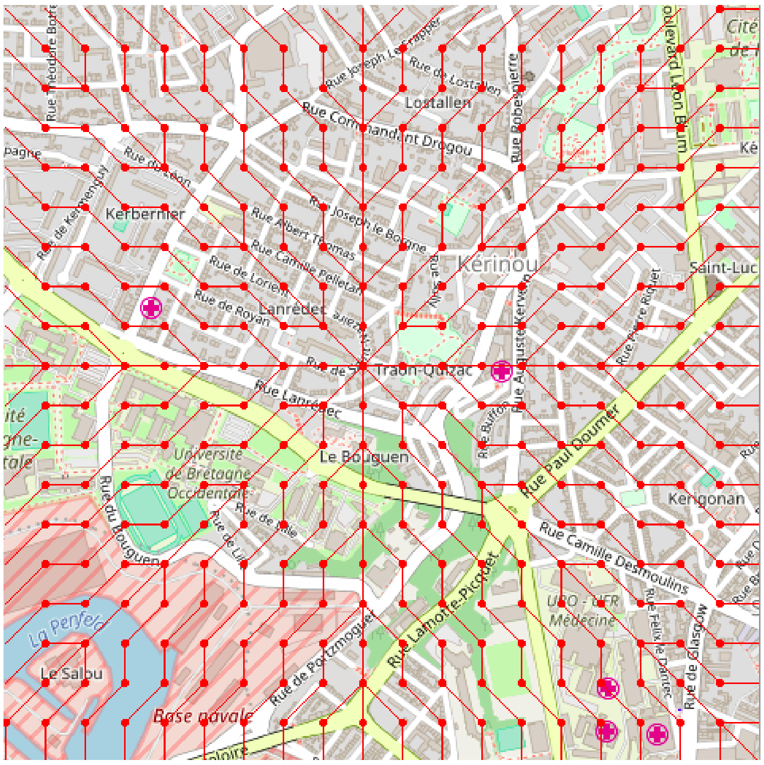
Figure 5.1 – (a) Segmented lines are laid over map of zone to depict the routes for circulating value of root cell (x=3, y=3) to all the other cells in space. The space composes of 36 cells in regular square grid with resolution 6 $\times$ 6 in pixel. Note that, every cell can obtain incoming values at any point of its four corners.
(b) For instance, a directed graph shows a typical routing scheme for communication by executing nondeterministic BFS in which each cell accepts one and only one value of oncoming values from surrounding neighbor cells in random order.

Figure 5.1a shows a small zone is segmented to product data space for nondeterministic routing computation. Because the simulator developed in occam-pi programing language, so that it is inherited the nondeterministic of run-time processes from KRoC compiler [98][1].

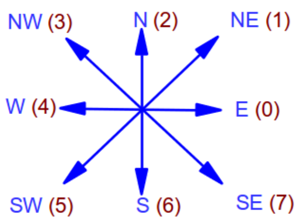
5.2 Directed Breadth-First Search

Directed Breadth-First Search (DBFS) is derived from Breadth-First Search (BFS). The main motivation for this work is to propose a deterministic routing strategy instead of nondeterministic one providing by the original BFS.

Algorithm 8 *outDirects* presents two distinguish processing to circulate messages associated with either odd or even step of execution. As shown in Figure 5.2a, each cell (process) forwards received messages to three adjacent neighbors according to its relative position with the root cell in the system. Notice that in Algorithm 8, *changeDirect* variable is used to indicate a change of direction, if any, in comparison with an original direction from the root.



(a)



(b)

Figure 5.2 – (a) Routes from a root node to the other nodes by executing deterministic BFS. (b) Direction computation.

Figure 5.2b depicts the assigned values of directions. From this definition, the appreciate output direction is computed from the direction of coming message as explained by

[1]Noticed that Occam-pi adopting zero-based numbering in which the initial element of a sequence is assigned the index 0, rather than the index 1 as C programming language.

Algorithm 8.

Algorithm 8 Select directions for DBFS

```
PROC compDestDirects
    IF step REM 2 = 0                                        ▷ Even steps
        IF orgDirect IN (N, W, S, E, NILL)
            IF changeDirect = NILL
                switchDirectForEvenSteps(inDirect)
            ELSE
                calcOutDirects(changeDirect, outDirects)
        ELSE
            calcOutDirects(orgDirect, outDirects)
    ELSE                                                     ▷ Odd steps
        IF orgDirect = NILL
            IF inDirect IN (N, W, S, E)
                calcOutDirects(inDirect, outDirects)
            ELSE
                switchDirectForOddSteps(inDirect)
        ELSE
            IF orgDirect IN (N, W, S, E)
                calcOutDirects(orgDirect, outDirects)
            ELSE
                IF changeDirect = NILL
                    switchDirectForOddSteps(changeDirect)
                ELSE
                    calcOutDirects(changeDirect, outDirects)
```

Algorithm 9 shows how to compute output direction values (*outputDirects*) corresponding to a certain input direction (*inDirect*) to spread out messages from the root cell in DBFS approach.

In order to explain the DBFS algorithm, first it needs to explain the meaning of variables being used in Algorithm 8, 10.

step is a variable to trace number of steps during processing.

orgDirect keeps the value of the original direction for each route.

changeDirect stores the value of direction in case of it being changed in comparison with *orgDirect* as traversing along a branch.

inDirect is direction of received message and *outDirects* are directions to send messages out.

Figure 5.3 provides a visual view of helping to easily realize how to determine the directions for sending messages in the DBFS algorithm. Algorithm 8 describes the process steps

Algorithm 9 Computing the directions of outcoming messages based on a coming message direction.

```
1: PROC calcOutDirects(inputDirect, outputDirects)
2:     IF inputDirect = NILL
3:         outputDirects := NILL
4:     ELSE IF (inputDirect >= 4)
5:         outputDirects := inputDirect - 4
6:     ELSE
7:         outputDirects := inputDirect + 4
```

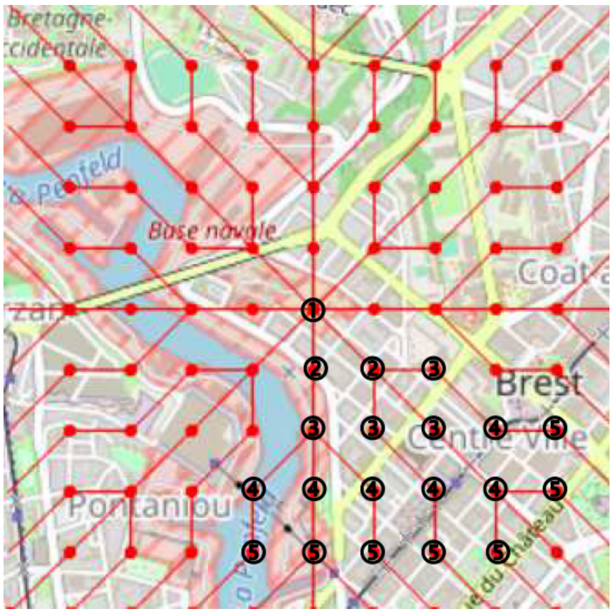


Figure 5.3 – Routing strategy for circulating information by applying DBFS algorithm. The numbers inside circles indicate the step of processes.

to compute target directions to forward an incoming message. Notice that at starting point, *parentId* variables are NILL for all cells.

In general, the rule of selections depend on the process step, associated with the location of cells and routing history of previous steps. In this strategy, each cell accepts just one income message for a neighbor but it can send out to one or to all three neighbors. The routing paths in a system are deterministic at compile time. The reference conditions to switch messages corresponding to whether odd or even step as presented in Algorithm 10. This process also regards to *orgDirection* and *changeDirection* as mentioned by Algorithm 8. For a cellular system with Moore 1 neighborhood, space is divided into 8 sub-spaces, corresponding to eight different directions. Every direction for circulating information is assigned an integer value from 0 to 7 as illustrated in Figure 5.3. The aim of these numeric assignments is to facilitate the computation of outgoing directions from incoming messages. Algorithm 9 mentions the trick providing an easy way to compute directions for sending out messages for each income message.

The key difference between DBFS and BFS is how to circulate messages. With BFS, each

Algorithm 10 A strategy to switch messages for even steps in process. For the odd steps, the process is computed in the similar way.

```
procedure SWITCHDIRECTFOREVENSTEPS
    Input: inputDirect                          ▷ Direction of coming message
    Output: outputDirects                       ▷ Directions of outcoming messages
    SWITCH inputDirect
     CASE (N)
        outputDirects := (S, SW, SE)
     CASE (W)
        outputDirects := (E, NE, SE)
     CASE (S)
        outputDirects := (N, NW, NE)
     DEFAULT
        outputDirects := (W, NW, SW)
end procedure
```

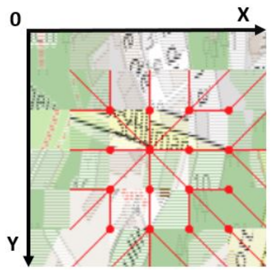
cell receives and send messages from/to all other processes in a cell system. In the context of the grid system, it could be done by using the broadcast information mechanism. Due to this way of communications, at a moment each cell can receive several messages from its neighbors. How they choose which message for updating it local variables is unpredictable. Otherwise, DBFS suggested with the main objective is to give a uniform and deterministic rule in communication for all cells. The strategy as explained in Algorithm ***compDestAddr*** plays this role. The advantages are easy formalization the behaviors of cells in a definite condition. Another important routine is ***compNodeVis*** which is employed to determine the visibility of cells from a root cell based on ground elevation profiles producing throughout the process.

Figure 4.2 shows the ground elevation profile of a route at the Roc'h Trevezel in our experimental measurement. Let's consider that an emitter located at furthest left of the chart, with a distance of 0 m, and elevation of 350 m. Along with this route, there is a remarkable place with a ground elevation around 170 m at a distance of 2774 m. Moving further beyond this high place, a radio signal emitted by the emitter is unable to reach the three successive places at distances of 2997 m, 3269 m, and 3541 m according to LoS condition.

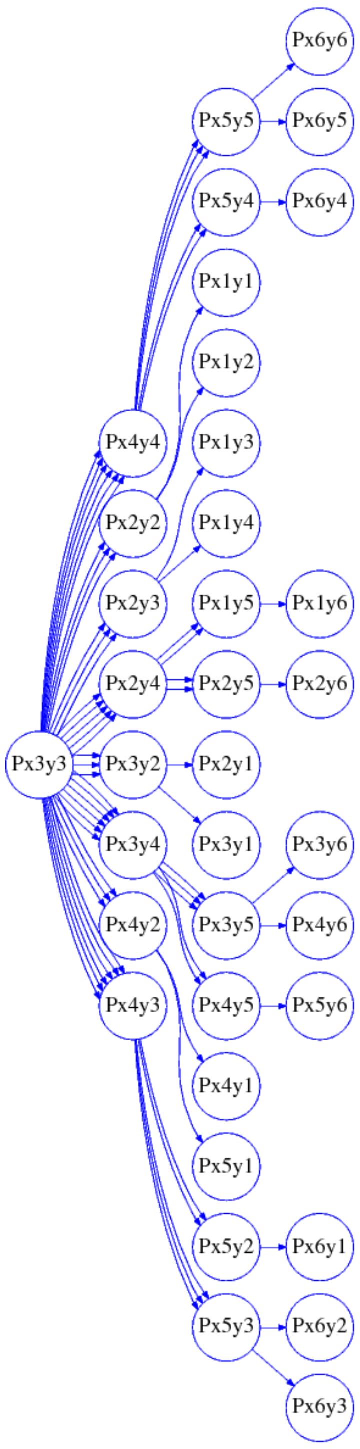
It should take into account the constraint about the strength of the received signal as a condition to establish radio links. If characteristics of the cell do not satisfy either LoS or the signal strength condition, cells stop forwarding messages to neighbors. It means that the signals are blocked by obstacles or bad transmission environment.

Algorithm 11 Directed Breadth-First Search Algorithm

```
▷ Step 1: Initializing local values (myValues, traceRoute, and state)
initValues(myValues)                ▷ Initialize node's geo-location and other local values
IF isRoot
    SEQ
        traceRoute := myValues
        state := gotSearch
ELSE
    state := waitSearch
▷ Step 2: Preparing for the fist communication session
IF state=gotSearch
    PAR i=0 FOR numOutMsg
        outMsg[i] := traceRoute
▷ Step 3: Communicating M to N
SEQ i=0 FOR numNodes - 1
    IF state=gotSearch
        PAR i=0 FOR SIZE outChan
            outChan[i] ! points ; outMsg[i]
    ELSE
        PAR i=0 FOR SIZE outChan
            outChan[i] ! null ; nullMsg
    PAR i=0 FOR SIZE inChan
        IF inChan[i] ? CASE
              null ; nullMsg
                SKIP
              points ; inMsg[i]             ▷ Receiving a new point from the neighbors
            SEQ
                newState := gotSearch
                IF newPoint.parent = myValues.Id
                    addIntabChildren(newPoint,i)
                    TRUE
                        SKIP
▷ Step 4: Updating local values
IF state = gotSearch
    state := forwardSearch
ELSE
    IF (state = waitSearch) AND (newState=gotSearch)
        PAR i=0 FOR SIZE inChan
            IF inMsg[i]= myValues.Id
                SEQ
                    traceRoute := inMsg[i]
                    outMsg.addresses := compDestAddr          ▷ Compute addresses
                    visible := compNodeVis        ▷ Determine current cell is visible
                IF visible
                    addInRoute (myValues)
                state := newState
        PAR i=0 FOR numOutChan
            outChan[i] := traceRoute
```



(a)



(b)

Figure 5.4 – (a) Deterministic routing algorithm for exploring space. (b) A graph for deterministic routing strategy.

5.2.1 Evaluation the accuracy of LoS computation

Bresenham's line algorithm in a cell system

In computer graphics, Bresenham's line algorithm is a well-known method to draw a graphic line by generating successive pixels with a minimum relative error along the corresponding straight line connected by two endpoints. Because of its simplicity and lightweight execution [161], this algorithm is suitable to perform cellular automata for LoS connection modeling and simulation. The cellular method operates in a cell system which comprises generated cells using NetGen tool-set as described in Appendix F. Figure 5.5 illustrates a comparative study between Bressenham and DBFS algorithm for drawing lines.

Accuracy comparison between Bresenham and DBFS algorithm

In this work, we use *deviation* as a metric to evaluate the reliability of Bresenham's straight line and DBFS algorithm. The term *deviation* refers to the absolute difference between a point of a segmented line and its correspondent reference line. Three typical cases of routes are shown in Figure 5.6.

Both cases of lines produced by Bresenham and DBFS (black lines in Figure 5.6) are exactly matched with a straight line for reference from the root cell to cell Px1y10. The *deviation* equals to zero in this case.

The routes from the root cell to cell Px9y12 (blue lines), Bresenham and DBFS solution give different routes. However, they suffer the same mean of deviations, 0.224. This mean value is figured by comparing, in turn, each point along the routes with the reference points of dotted blue straight lines.

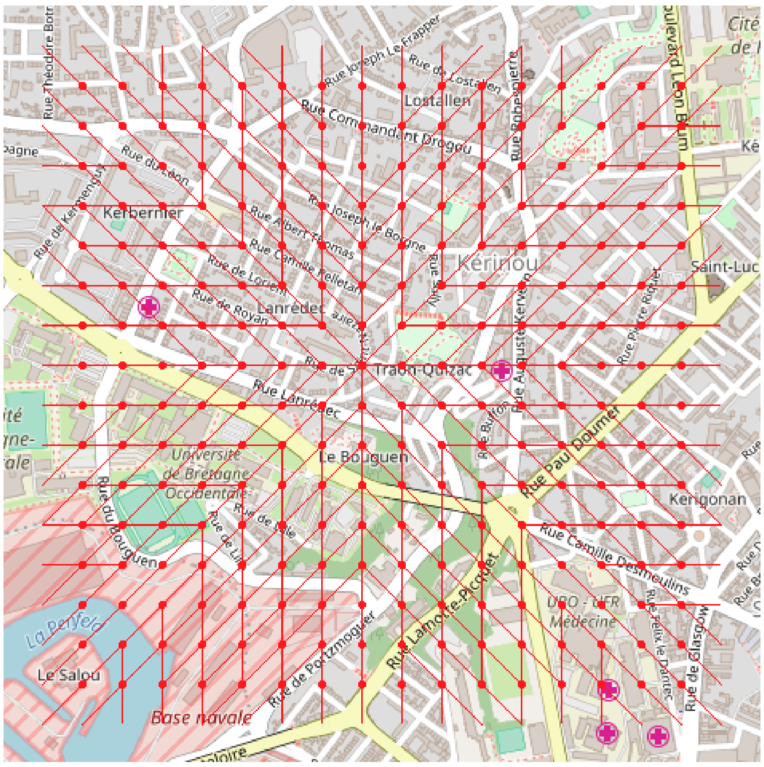
For cell Px17y4, a segmented rad line created by the Bresenham's algorithm in Figure 5.6[a] has a maximum deviation of 0.447 and mean of all deviations is 0.268. This result is better than DBFS's segment line (segmented red line in Figure 5.6b). DBFS result suffers maximum deviation 1.431 and means of deviations 0.716.

With the objective to compute the routes for transferring values adopting LoS constraint in regular grid space, Bresenham's algorithm is more accuracy than DBFS one in terms of *deviation*.

5.2.2 Influence of multi-scale data on simulation accuracy

As shown in Figure 5.5b, DBFS routes are comprised of eight sub-routes with the same topology associated with eight different directions to explore all area. For this reason, only one sub-route (see Figure 5.7 is used in the next analysis.

In this case, the maximum deviation suffering by Bresenham's line algorithm is $\frac{1}{\sqrt{10}}U \simeq 0.32U$. Note that U is the size of the cell in pixels. In theory, the maximum error for Bresenham's line algorithm is less than 0.5.


(a)

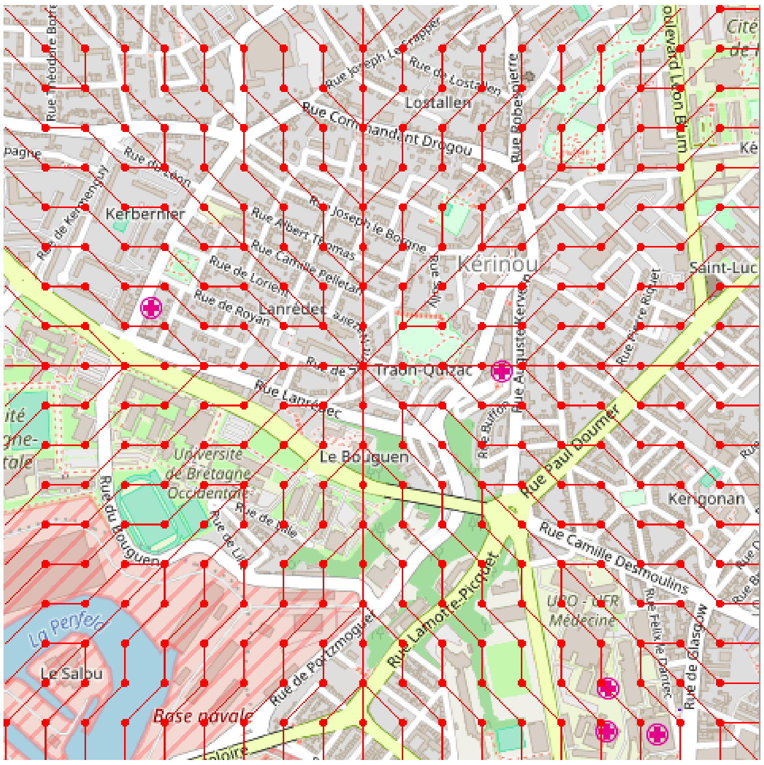
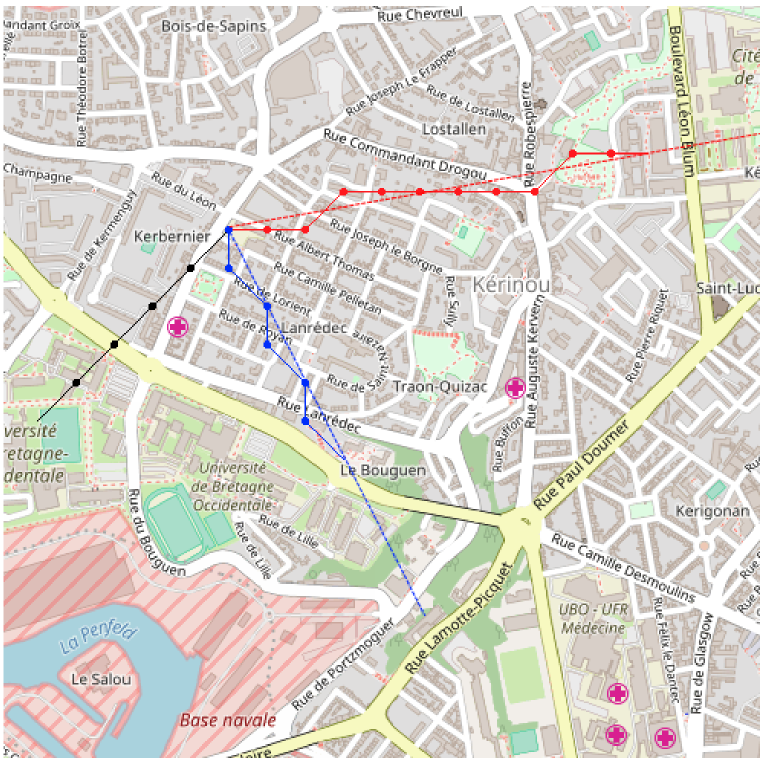
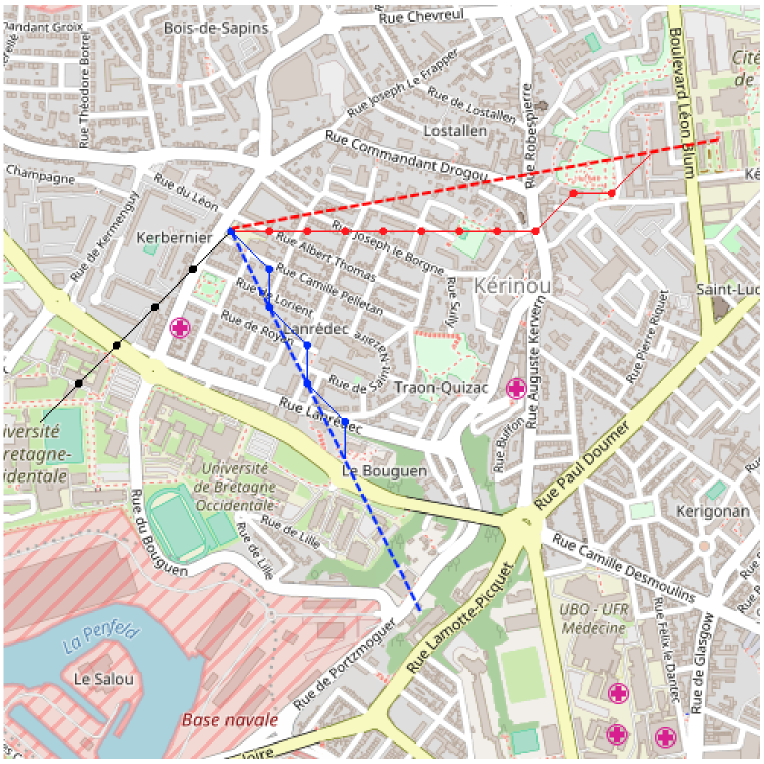

(b)

Figure 5.5 – (a) Bresenham's routes. (b) DBFS routes.

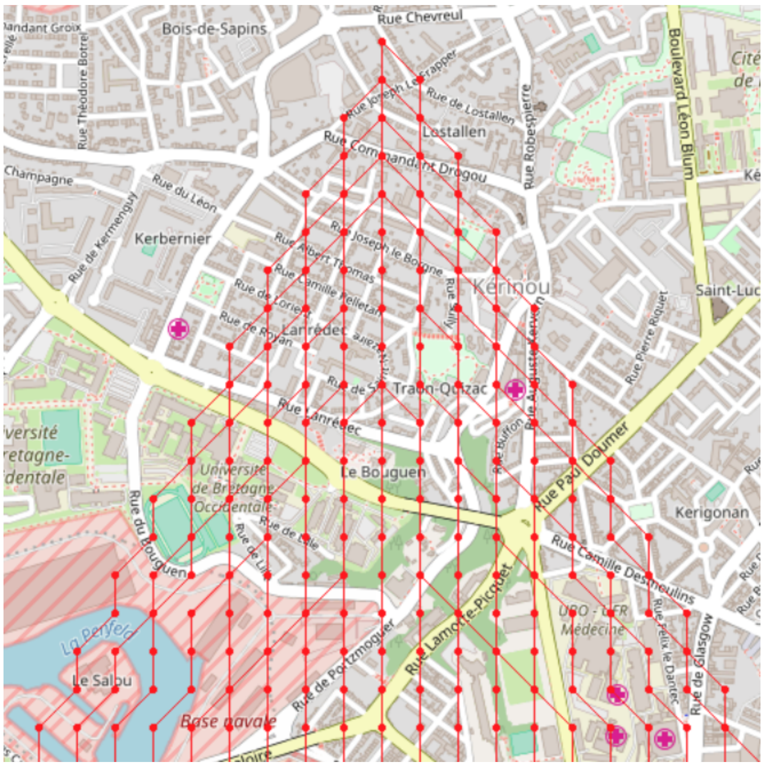


(a)



(b)

Figure 5.6 – Routes from root cell (x=6, y=6) to three point (x=1, y=11), (x=9, y=12, (x=17, y=4) indicated by black, blue and red color respectively.
(a) Segmented lines are produced by implementing distributed parallel version of Bresenham's straight line. (b) Executing DBFS.


(a)

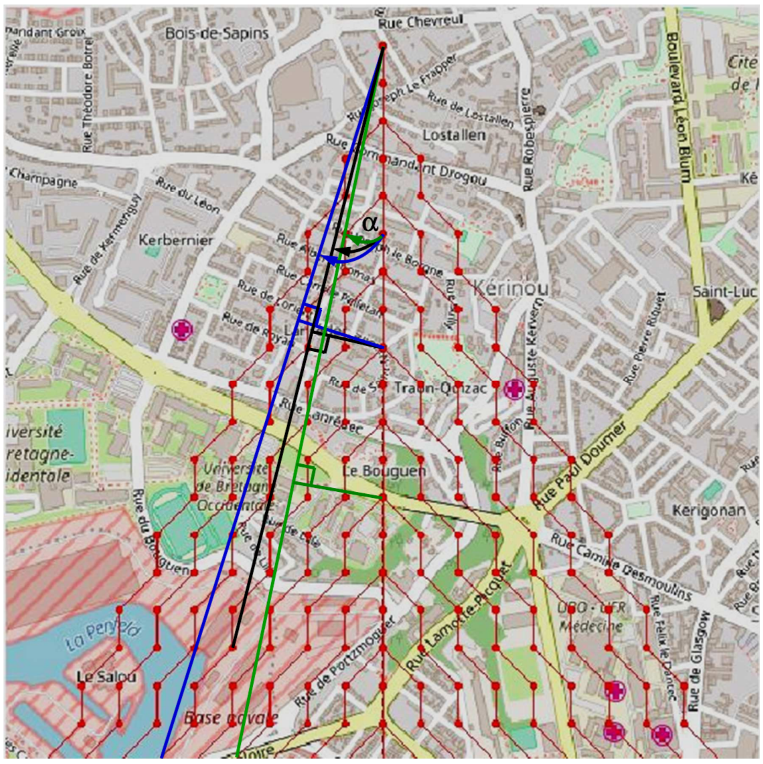

(b)

Figure 5.7 – (a) Bresenham's routes from a root cell for a certain direction. (b) DBFS routes from a root cell for a certain direction.

Figure 5.7 (b) demonstrates three routes applying DBFS algorithm. Comparing these routes, the maximum values of deviations along the route compared with the straight line are varied depending on the relative position of the cell in comparison with the root cell. Because of α angle is always less than $\frac{\pi}{8} \simeq 22.5^o$, so that maximum deviation is approximately $\frac{\pi}{8} \times steps \simeq 0.39 \times steps$.

Our algorithms produce routes from a root cell, where putting an emitter, to the others cells in a zone. In some cases, they match exactly with routes generating by Bresenham's line algorithm. In other cases, they suffer deterministic deviations in comparison with the Bresenham ones.

Now turning to discuss about the error of our approach, the maximum error must be suffered is approximately $1U$. For instance, in Figure 5.7, considering a route from A(10,1) to B(18,18) the function of this line is $\frac{17}{8}x - y + \frac{-162}{8} = 0$, so that the distance from concerning point, M(10,3), is $\frac{16}{15} \simeq 1.0667$.

From these results, it can conclude that our approach suffering maximum deviation of points along the route can be computed as $sin(\frac{\pi}{8}) \times steps \simeq \frac{\pi}{8} \times steps$. Notice that $step$ in this the equation is the distance in cells from the root cell to a considering cell.

It is critical to remind the algorithmic complexities, let suppose that a zone can be divided into n cells in a cell system. By applying Bresenham's line algorithm, it requires n^3 processes to produce the routes from every cell to all other cells. In more lightweight performance, DBFS needs running just n^2 processes to complete the same task.

Table 5.1 – Evaluation errors associated with resolution (in pixels) for Roc'h Trevezel

			Non deterministic		Deterministic	
Resolution	Size (m)	Cells	Errors	%	Errors	%
5@5	191	48618	8	17.78%	5	11.11%
10@10	382	12099	13	28.89%	5	11.11%
15@15	573	5402	16	35.56%	7	15.56%
20@20	764	2970	17	37.78%	8	17.78%
25@25	955	1892	19	42.22%	10	22.22%
30@30	1147	1332	23	51.11%	18	40.00%

5.2.3 Comparison of accuracy, performance between BFS and DBFS

BFS and DBFS strategy are shown in Figure 5.1b and Figure 5.7b respectively. Both of them can sweep all cells in space with the same order. The difference is the routes on which the values from root cell can reach to the other cells. In the context of physical simulation based on CA, directed routing is critical in case of transition rules obeying a certain constraint to mimic physical phenomena. For example, LoS constraint in communication, distribution of rainwater.

Figure 5.9 shows calculating percentages of errors associated with different resolutions of the cell. In the context of routing the message to mimic LoS, the results of the DBFS

Table 5.2 – Performance comparison of DBFS on three different NVidia Graphic cards (execution time in millisecond).

Resolution (pixels)	820M	GTX680	GTX1070
5	636.036	95.423	69.743
10	164.793	26.162	18.991
15	76.757	15.405	7.608
20	40.655	11.855	3.709
25	27.091	11.455	4.295
30	20.108	11.568	2.811

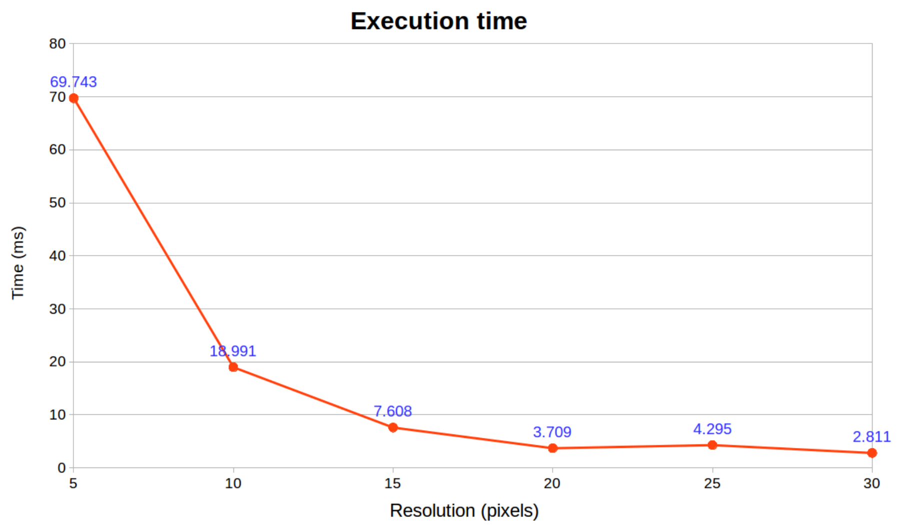


Figure 5.8 – DBFS computation times for different resolutions of cell from 5 to 30 pixels. This statistics was obtained with DBFS executing on PC equipped Core i7-7700K CPU@4.20GHz × 8, 16 GiB DDRAM, card NVIDIA GeForce GTX 1070 (1920 CUDA cores).

algorithm is more accurate than BFS one. The reliability of computations is in proportion to the resolution of cell in pixels.

5.3 Communication coverage optimization

Communication overage refers to a zone consisting of places where radio signals can reach. To optimize ranges, two major issues need to be concerned. These issues are radio waves locked by obstacles and propagation loss proportioning with the distance due to the conditions of the transmission environment. In smooth terrain areas, without significant changes of topographic surface, it can be considered as near line-of-sight for radio transmission in which distance for communicating data could be reached up to hundreds of kilometers over the Earth plane. Otherwise, in complex terrain areas, terrain heterogeneity causes serious impact on the quality of radio links, especially for low power and long-range communication

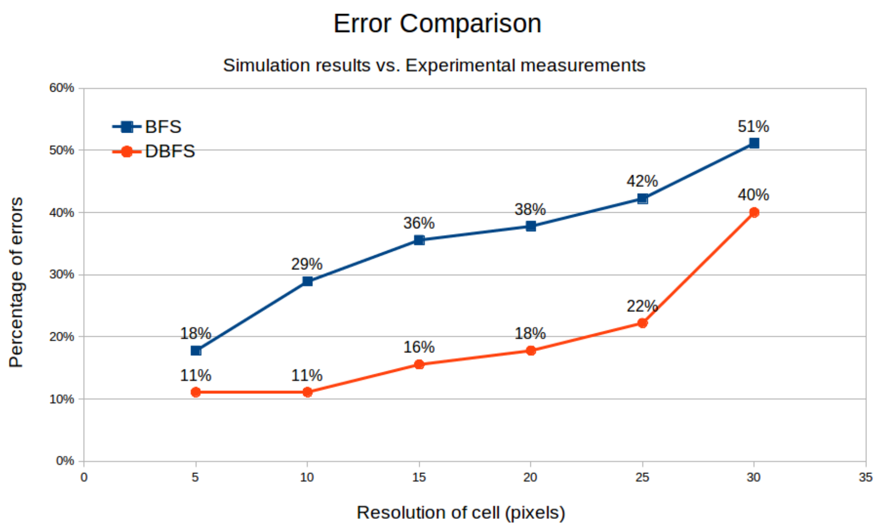


Figure 5.9 – The comparison of accuracy between simulation results and experimental measurements for Roc'h Trevezel, a mountain area. Blue line and red line are percentage errors of BFS routing and DBFS routing computation in comparison to actual values obtained from measurements respectively.

networks. In theory, an ideal coverage is usually indicated by a circle with a certain radius considering as maximum communication range. Due to the complexity of geography and obstacles such as building, vegetation, etc. the actual coverages in the real world are seldom as uniform dishes. Consequently, unpredictable gaps need to be solved when deploying sensor nodes to establish robust radio links in wireless sensor network considering geographic topology. For these reasons, we proposed a CAD tool flow which helps to predict rapidly the coverage of a LoRa emitter based on cellular automata and massive parallel execution.

These algorithms aim at optimizing a number of cells that could be selected as root (place to put the emitters) for covering all complex geographic area. At first, the LoS algorithm, as mentioned in Section 4.2, is executed to explore its neighbor's nodes which could be visible from the first selected node as emitter/root [2], R_0. The communication coverage of R_0 based on the line-of-sight (LoS) condition is determined. In order to deploy a wireless sensor network aim at covering all area, it is necessary to select a visible cell from R_0 to be the next emitter. Three different algorithms were proposed to determine the root node namely maximum distance (*maxDistance*), maximum visible neighbor cells (*maxNeighbors*) and maximum elevation (*maxElevation*). In the next sections, we will describe in turn them.

5.3.1 Coverage optimization algorithms

maxDistance algorithm

To select base station based on maximum distance from previous nodes, distances from R_0 to each cell illuminated by it are calculated by applying Euclidean formula corresponding

[2]For convenience, the terms *emitter* and *root* are used interchangeably throughout this thesis.

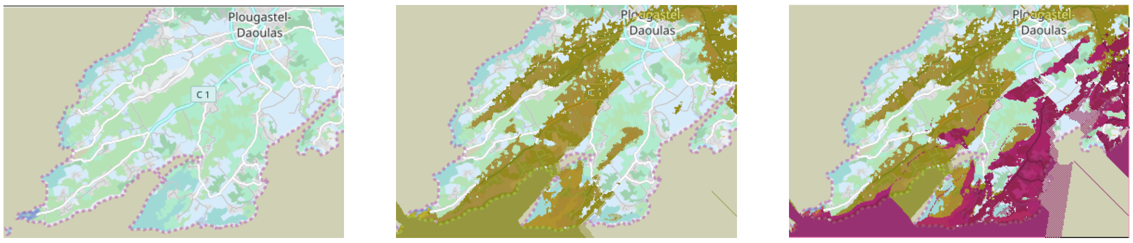
Algorithm 12 Select a root cell, where an emitter is located, using a Euclidean distance metric.

```
1: function LatLonToMeter(lat, lon)    ▷ Calculate $(x, y)$ in meter using Mercator projection
2:     originalShift := $\Pi \times 6378137$
3:     x := $(lon \times originalShift)/180$
4:     y := $(log_{10}(tan(((lat + 90) \times \Pi)/360)) \times originalShift)/\Pi$
5:     return (x,y)
6: end function

7: procedure maxDistance    ▷ Select root cell based on the maximum of distances
8:     Input: root cell ($R_0$) and all cell in cell system $cell_i$
9:     Output: set of cell selected as root.
10:    Initialization $maxDistance$, $rootID$
11:    ▷ Calculate $R_0(x, y)$ from its geo-location
12:    $(R_0.x, R_0.y) := LatLonToMeter(R_0.latitude, R_0.longitude)$
13:    for each $cell_j$ do
14:        ▷ Calculate $cell_i(x, y)$ from its geo-location
15:        $(cell_i.x, cell_i.y) := LatLonToMeter(cell_i.latitude, cell_i.longitude)$
16:        ▷ Calculate Euclidean distance between $R_0$ and $cell_i$
17:        $cell_i.d := SQRT(SQR(cell_i.x - R_0.x) + SQR(cell_i.y - R_0.y))$
18:        if $cell_i.d > maxDistance$ then
19:            $maxDistance := cell_i.d$
20:            $rootID := cell_i.ID$
21:        end if
22:    end for
23: end procedure
```

to theirs coordinate (longitude, latitude) which are fetched from the map services such as OpenStreetMap, Google map, etc. Consequently, the cell with maximum distance value will become the next root node. An advantage of this strategy is to allow reducing the interference between base stations which normally produce high output power for larger communication coverage. The remaining steps are in the same way. Notice that the cells were chosen as root node that cannot be selected again in the following turns. After each step, a number of the cell not be illuminated by the roots are a statistic. The procedure performs until all cells in the cell system (map/image) are visible. It means that the result of the program is a wireless network (in hierarchical/mesh topology) which can cover all area in term of communication and networking.

maxNeighbors algorithm

With the *maxNeighbor* algorithm, selecting next root node based on a maximum number of visible neighbor cells, every cell under the coverage of R_0 are asked for number cells that it could be visible by applying LoS algorithm as well. To enhance the speed execution, parallel computing on GPUs are utilized for all of the algorithm processes. The cell in which the maximum visible neighbor cells was chosen to be the next root, R_1. The successive processes are as the same until all cells are reachable by a set of root nodes. For instance, the coverage of the network with one and then two emitters are illustrated in Figure 5.10 (b), and (c) respectively.



(a) Display map (b) One emitter (c) Two emitters

Figure 5.10 – (a) Map in Plougastel peninsula, France. (b) First, a cell is select as root node indicated in red color and its coverage in yellow color. (c) Second, Two cells with maximum visible nodes is chosen automatically for root associated with cells under its coverage (indicated in violet color). In this experiment, map of zone is segmented into more than 90000 cells with resolution 2×2 pixels.

maxElevation algorithm

Another way is to select the node for based station role to rely on the highest elevation of cells. Because elevation is a key factor in the line-of-sight communications, a cell within the coverage of the current root corresponding to the maximum elevation of cells can be chosen as the coming root. This strategy seems to be good but the experimental simulation results show that it is worse. For this reason, the further evaluation was carried out for two former algorithms.

5.3.2 Performance comparison of coverage optimization algorithms

To evaluate the execution time of three proposed select base station algorithms as described above, we selected a zone around 60 km^2 at Plougastel peninsula, France on QuickMap (see) and transfer it to Pickcell/NetGen to generate CUDA code for simulations. Table 5.3 is produced by changing the cell sizes of data and then evaluating the simulation results on GPUs with $nvprof$ command [165]. The program performs to select root nodes automatically and compute its corresponding coverage until all cells are under their coverage. Execution times for each algorithm are presented in Table 5.3.

Table 5.3 – Performance comparison between maxDistance and maxNeighbor. (NVidia GTX 480, 480 CUDA cores)

		Execution time (s)	
Size of cell (pixels)	Number of cells	maxDistance	maxNeighbor
10	7217	30.11800	18.26566
15	3158	8.51220	5.82960
20	1763	2.26680	2.14987
25	1155	1.88520	1.10281
30	783	0.69103	0.55861
35	575	0.63420	0.36853
40	420	0.26833	0.15095
45	342	0.11679	0.09424
50	272	0.07916	0.06073

This statistical result shows that *maxNeighbors* algorithm gives better performance than *maxDistance*. With *maxNeighbors*, a root is selected at each step based on its capability to provide the maximum of visible cells, except for the first root R_0, so that this is an appreciate method to optimize the set of roots for communication coverage under LoS constraint. Considering to the *maxDistance* algorithm, it seems not suitable for coverage objective due to the maximum distance from the previous root is not a key factor to ensure the best coverage, probably reducing interference effect between the roots.

6 A Development Method for Sensing Systems

The objective of this chapter is to propose a complete development flow for sensor networks based on concurrent process expression.

Section 6.1 proposes methodological approach for development of application on network in a portable way.

Section 6.2 are programming considerations with in mind a complete flow including simulation, and using result for code generation.

Section 6.3 is a sample application on LoRa with round-robin control over a distributed application.

Section 6.4 exposes validation work with an extension of TVMs working on nodes interfacing LoRa transceivers using a SPI bus. This technique was also used to validate range computation from simulators described in Section 4.4.

6.1 Organization structure of distributed sensing systems

Distributed sensing can be defined in a general way as a composition of local observations, operated under a time schedule. Various context can match this definition, such as synchronous distributed sampling, or round-robin monitoring of distributed clients.

Example applications monitoring for of the traffic, river flows, pollution, in the first case, while the second case embed data collections from several locations, perhaps without any physical relations.

In such applications, data collection points are characterized logically, by the type of service produced, and the geographic position, rather than by any interconnection constraints. Mesh networks and star organizations could match the same application objectives. The methodology is to bind a distributed set of services to a particular network. This is useful in architecture synthesis as a choice for resource allocation, and operation scheduling, these choices defining how a computation is realized [166], with a component cost and performance factor.

6.1.1 Top-down approach for development distributed sensing systems

In Chapter 3, we have presented the local representation of a wireless node, having radio periodic communications with neighbors on radio links. Given a network of nodes with sensing and routing capabilities, the problem is to define where these activities take place, to deduce the program structure for each node, with timing and resources for the distributed program.

Inside a top down flow, tool designers have typically two elements:

- A program abstract representation, with program nodes and edges representing internal constructs,
- An architecture description with computing resources and data paths.

CAD tools embed algorithms to allocate computing resources and time slots for operations. These algorithms will propose solutions to share resources, reducing hardware costs, or at the opposite will allocate excessive number of resources to develop parallelism, and reduce latency. This technique can be applied to distributed sensing environment, with resources being sensor nodes, and data path being radio links. Furthermore, taking into account a multiplicity of sensors in each node, the different frequency of measures, data conversions and communications, each node is itself a system carrying parallel operations.

Figure 6.1 and 6.2 represents two practical versions of the same application, using respectively a star architecture such as LoRa, and a mesh connected architecture such as 802.15.4. The application could be the same, as example control of creeks during heavy rain, with a control sink somewhere. Communication delays are variable in the first case, due to the radio signal propagation latency, as explain in section 4.5.2, but delays are known due to the geographic work. In the second case, the network is synchronous, with super-frames that structure communications at a rate compatible with the application. Respectively sequencing and local tasks are different, with round robin collection in one case, and concurrent routing in the second case. Other critical differences are scheduling of operations, buffer sizes for communication, quality of services, complexity of the system.

To reduce application development time and to improve reliability and solution costs, it is interesting to look the feasibility of a complete synthesis flow. We did a partial contribution in this direction, with an executable specification with support components for the LoRa case.

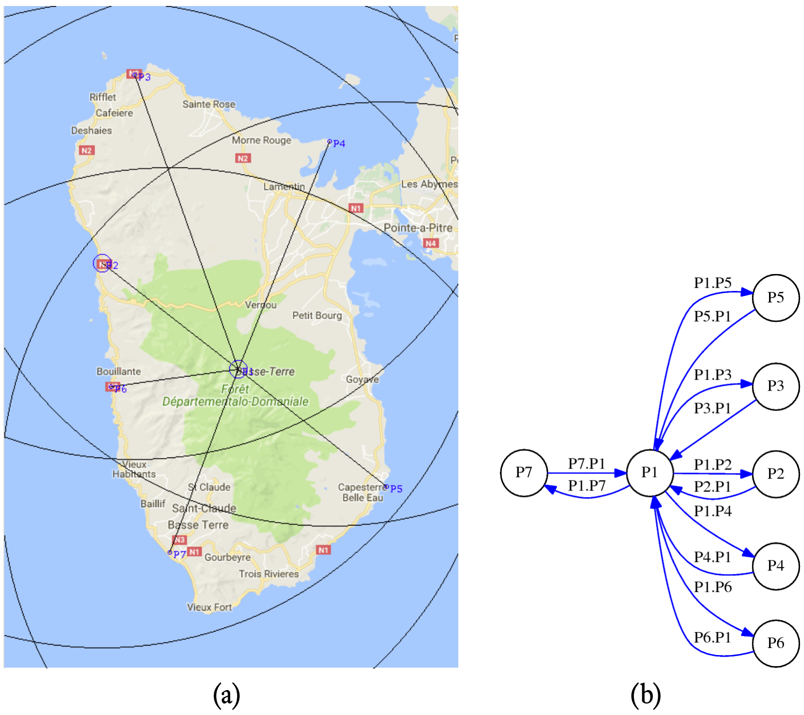


Figure 6.1 – (a) A star network (case of LoRa). (b) Directed graph of the star network with a sink node P1.

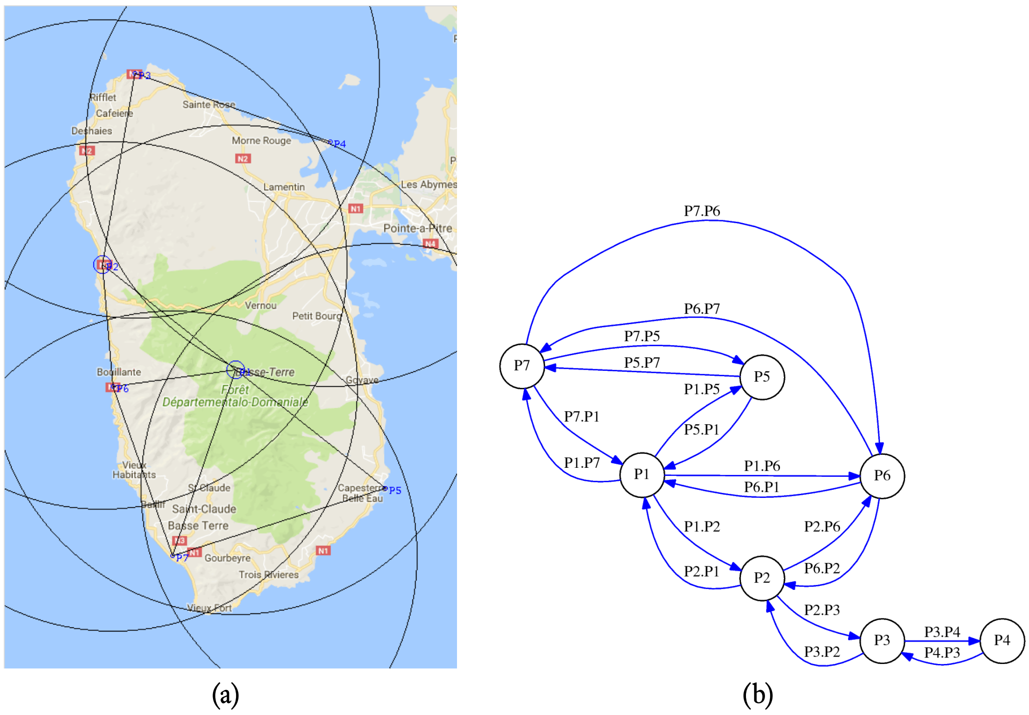


Figure 6.2 – (a) A mesh network (case of 802.15.4). (b) Directed graph of the mesh network. The connections between nodes are not complete so that it is necessary routing processes to send message from the sink node P1 to the other nodes.

These considerations lead to the following propositions:

- Use of a specification language requirements: possibility to represent network architectures, and node operations as well. Occam-pi was proposed in the NetGen tool bench, with a code generation system from geographic front-end tools.
- Specifying a general behavior at application level, with notion of concurrency, timing constraints with abstraction of the execution layer.

  We build samples of application at this level, and demonstrate by hand the translations to be operated, with demonstration on a small set of LoRa nodes.
- Production of a low level node behavior, providing the capability to produce Occam-pi program that communicate with neighbors, sense and compute locally.

6.1.2 A complete process oriented system organization

This system has a set of application services suitable for wireless sensing. Figure 6.4 shows the organization of the system. Top-level is a useful part with computing process which behavior is the sensing application. They can make arbitrary number and sensing activity can vary a lot depending on the application. They are portable between different architectures.

The second level is a description of system support implementing a real application. Each process is connected to a central controller (red arrows), and can interact with neighbors to access their states (blue arrows). This second level of communication is an abstraction of virtual neighborhood since the central node is effectively in charge of neighborhood management. Naming and binding channels could be done by the synthesis software based on geographic considerations or service names. The blue system allows nodes to share status, to make decision, and they represent a context similar to neighborhood in mesh network. In this way an application can have version on both network architectures, the red process being a sink.

We can distinguish the red sink, the blue nodes and the green services. Figure 6.1 shows a possible application context with the red sink in the middle and LoRa connections with blue sensors on the coast of the island. This architecture can be produced automatically with high-level tool, in view of physical application behaviors, for example water levels. It is also not noticeable that signal propagation simulation provides the information about communication delays.

6.1.3 Programming considerations

Following Concurrent Sequential Processes (CSP) paradigm [167], the system specifications are supported by programming languages such as compilers Occam, and Occam-pi for mobile system [101]. Occam-pi is derived from Occam enabling Milner's Pi-calculus [168], dynamic parallelism, mobile processes, mobile channels, extended rendezvous [169]. CSP propose a set of constructs allowing to specify concurrency as a hierarchical organization of processes communicating through blocking channels. The organization shown in Figure 6.4

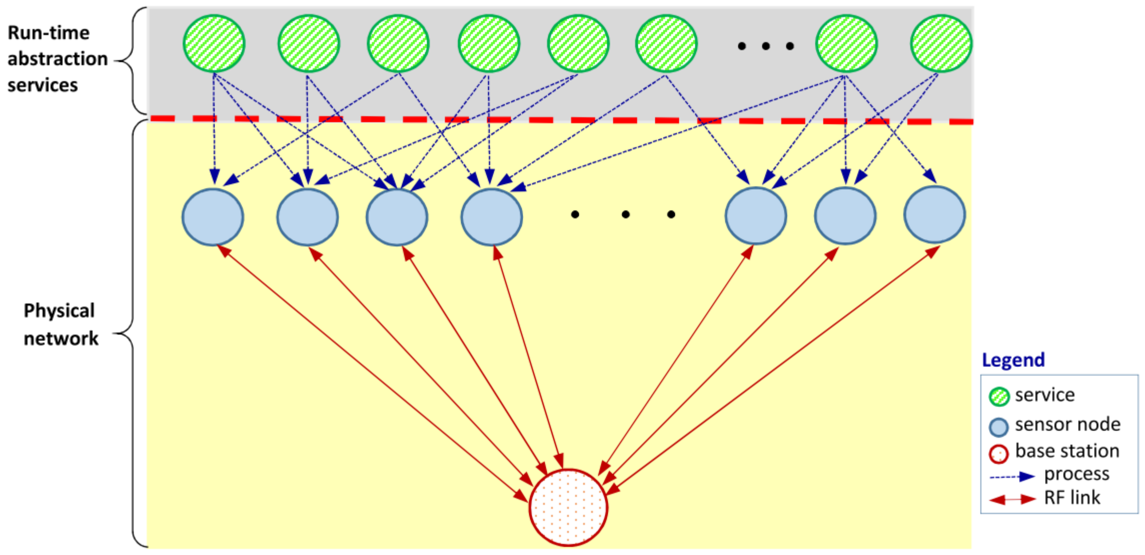


Figure 6.3 – Conceptual organization showing a star network of sensors, with run-time abstraction services at top and physical networks at the bottom. This application could also be deployed on mesh connected architecture as Figure 6.2

.

implies channel declaration, construction of process systems, and procedure behaviors for these processes. The control structures being used in Occam-pi language provide adequate information to produce program trees including parallel operations, channel communications, delays, nondeterministic communications. This kind of program provides support for process level simulation, and after code transformation, mapping to real network.

Use of Occam-pi for simulation is yet implemented by code generator from the NetGen software [170]. A complete program has local activities for sensing and distributed activities for data collection and distributed control. The existence of global program is critical to associate a deployment description to application requirements such as data specification appearing in messages and local activities, or timing from sampling requirements.

Although we are still far from having a practical compiler for sensor networks, we can describe the flow allowing to produce network simulators and local programs corresponding to these simulators based on language constructs, local supports on current system on chip for sensing, and communication protocols for common sensor networks.

All the practical background about Occam-pi execution was taken from "Transputer instruction set: A compiler writer's guide" [171, 172] that describes processes representation inside a kernel, channel representation, and operation implemented for each instruction. This machine exists as real hardware (Transputers) and software implementation for several micro-controllers (TVMs)[1].

[1] We used a 8-bit TVM version for the Arduino and studied hybrid of software and hardware implementation for Cypress coarse gain reconfigurable PSoC architectures (appendix J).

Distributed sensing modeling using Occam-pi

Occam-pi provides a complete set of primitives for the expression of concurrency illustrated in reference of Figure 6.4.

- PAR constructs for the development of structural parallelism controlled by barriers. Service processes (green) and node processes (blue) can be specified as PAR constructs with arbitrary number of components and arbitrary connectivities. PAR construct terminates with end of barrier detection inside the kernel primitives. The PAR can denote local or distributed activities.

- CHANnels as a single primitive for synchronization and communication. Channels (red) are implemented as physical radio links. These links are operated by low-level software binding circuits access to the local run-time Occam-pi kernels (CCSP) [173]. Kernels exist on each system and handle relation between process synchronization and termination of radio communications. In practice, we have implemented support to extend an existing virtual machine for the Arduino with a primitive software operating LoRa communications on a transceiver (in practical by a SPI bus, see also appendix E).

- ALTernatives specifying non-deterministic behaviors. Alternatives appear when a set of channels is activated without enforcing a particular order in synchronization. This is the case for processors with several peripheral circuits working concurrently to acquire or process information.

- PROTOCOLs specify data type for communications that circulate on channels. Compiler will verify conformity between local data structures and network messages. Protocols will appear on the red channels to denote neighborhood communications and control operations. These also provide a way to evaluate communication cost and delays.

- TIMERs allow to produce delays based on real-time clocks existing in the hardware of node. An obvious interest for timers is to sequence operations according to a general scheduling such as TDMA, or scheduling table produced for LoRa star system.

6.1.4 Details on: radio communications

Communications are LoRa messages exchanged between the red and blue nodes. Therefore the local programs need to control communication circuits to send and receive these messages. Detail about target systems and operating system support are given in appendix E and I: specific boards with LoRa transceivers.

In a main loop, the red sink in turns send messages to blue nodes as requests for services or data and then wait for responding messages from the blue nodes. When a message comes, the sink node receives that message if the destination address match with its own address. Otherwise the sink node helps to forward the message to the target blue node. Once receiving a message, the red sink also measure corresponding RSSI (Radio Signal Strength Indicator) and SNR (Signal to Noise Ratio) value to evaluate the radio link quality. For instance, Listing I.2 in Appendix I is a Occam-pi program running on a sink node.

6.1.5 Details on: services

Run-time services (green nodes) can be achieved by spreading distributed processes over a set of blue nodes through abstract links (blue arrows). This process are procedure calls.

After initializing the system, the program running on a blue node executes a main loop in order to wait for a request from a sink node. Once receiving the request, it performs routines for services associated with the request such as collecting data from sensors, control the actuators. After finishing the required services, the blue node must response to the sink node in form of either confirmation or results such as collected data. In this case, the red sink node play a core role to manage all the operation of the network. Listing I.2 in Appendix I is a Occam-pi program running on a sensor node (blue node).

6.2 Code organization

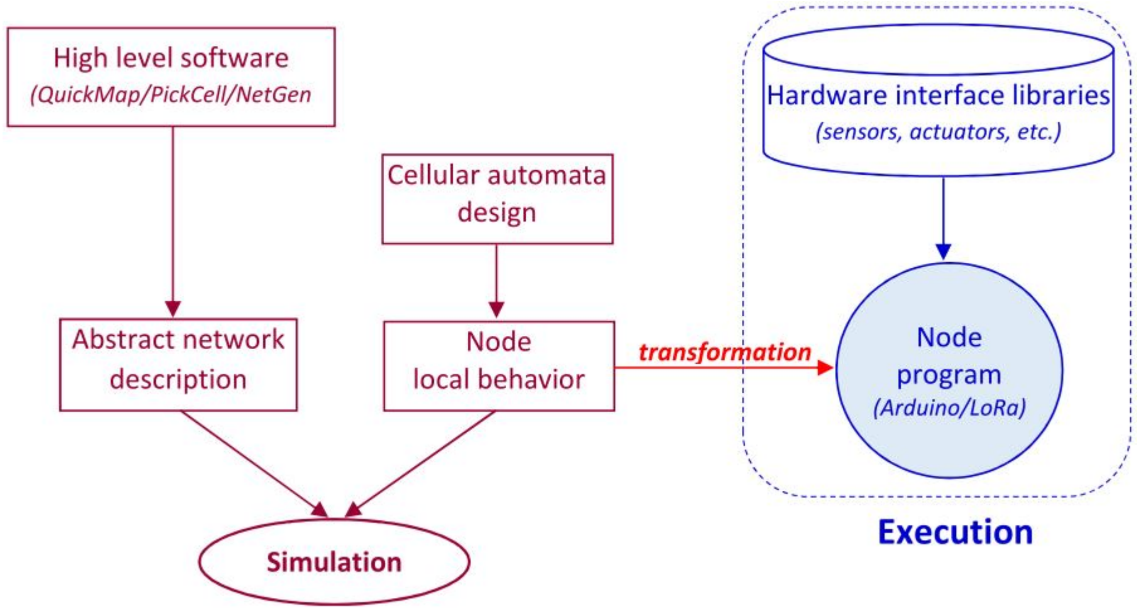


Figure 6.4 – Code organization comprises of *Simulation* and *Execution* block. *Simulation* helps to develop distributed algorithms for sensor networks. *Execution* is a process to port the local behavior of node on real hardware, for example Arduino running TVM.

The meta-simulator environment is a higher level modeling allowing to produce and manage networks instances. Models are described and produced from QuickMap/Pick-Cell/NetGen tools [135], and the simulations are produced in either Occam-pi or CUDA language.

- *The abstract network description* is an output from higher level tools. This is a static topology description which could be reusable for different kind of simulations. In case of the abstract description in Occam-pi for executing on multi-core CPUs, this abstraction will translate into a parallel construct with guarded channels allocated for communication links. Systems as large as hundreds of nodes, if not thousands, have been produced in this way.

- *The node local behaviors* are currently assembled by hand. They consist of the distributed parallel algorithm with message passing aiming at neighbor exploration, the building of routing scheme, base station voting, or many specific activities [174, 175]. For Occam-pi programming, local behaviors appear as an inclusion of developer's routines into the main simulation program. By following this way, several behaviors can co-exist and interact with each other in the same system.

6.3 Round-robin scheduling in LoRa star networks

Long-range communications have been proposed along with star network topology as a promising technology to satisfy the mentioned requirements [176, 55]. Sensing data at every sensor nodes can be inserted directly to a base station so that this trick facilitates setting system up and preservation. A typical network in a star topology as depicted in Figure 6.5, which is employed to collect environmental data to a base station from many sensor nodes distributed in a wide area based on time slices mechanism, namely round-robin scheduling.

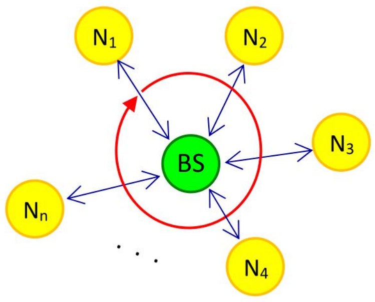


Figure 6.5 – A long-range star network comprising of a base station (sink node) and several sensor nodes.

It is necessary to building up an appropriate timetable to handle operations of the star network. Firstly, base station (BS) needs to emit a broadcast *hello* message to explore other nodes. All sensor nodes within the base station's communication coverage after receiving a "hello" message must response by sending back itself identification. Assume that the size of data needs to be sent are the same for all nodes. Consequently, the time slot for each node based on the duration time required to complete a transaction between the corresponding node and the base station. In fact, the time value for each sensor node is computed from the strength of received signal (RSSI).

It strongly depends on the node's distance and topology of geography. Figure 6.7 describes the principal of star topology and scheduling communication for the star network.

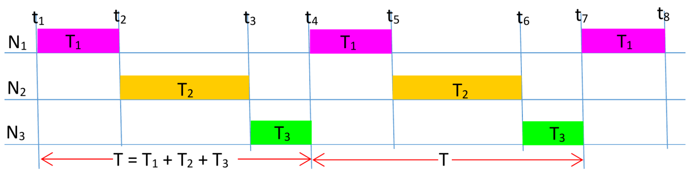


Figure 6.6 – A flow time describes a scheduling for division multiple access between a base station and three sensor nodes. Each sensor node requires different periods of time to complete its transaction successfully.

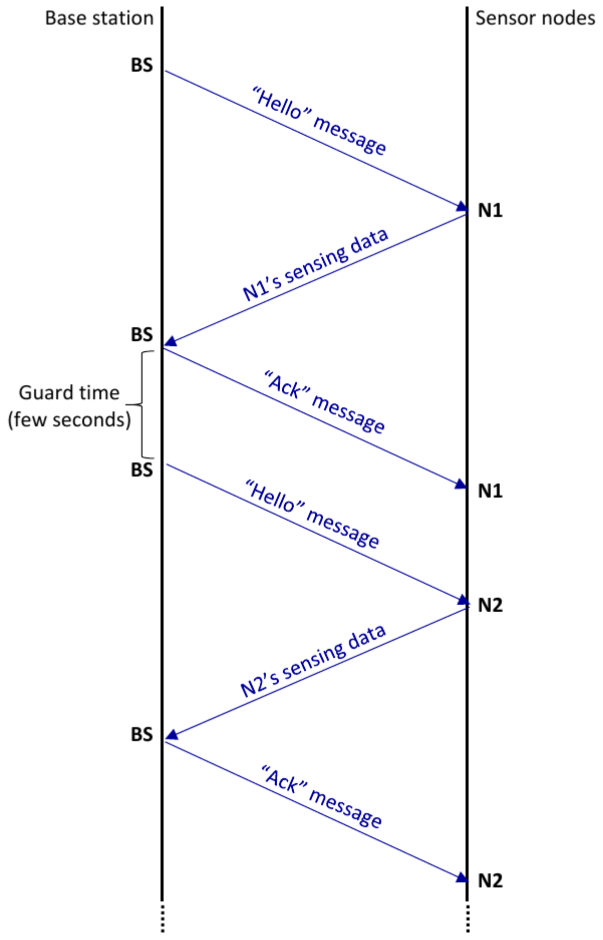


Figure 6.7 – A diagram describes a typical transaction flow between a base station and several sensor nodes based on scheduling approach.

In the next stage, a scheduling table is built up at the BS based on all time values. BS then circulates this scheduling table to all connected nodes. As a result, all nodes must obey this strict time flow in communications. Note that the global time extracted from GPS message is utilized as a time reference to synchronize all activities of the system. Figure 6.6 depicts an example of the round-robin scheduling for three nodes which require a different length of time for each transaction with a base station. These communications probably need a guard time in a few seconds between two successive transactions to ensure that the base station finishes completely the former transaction before starting the next one.

As can be seen in Figure 6.6, the cycle, T, value, called time-frame alignment, is the total sum of duration times that is necessary to communicate with all sensor nodes, including guarantee periods.

The operation of the experimental network can be described as following. By obeying strictly time scheduling, the base station starts a transaction by sending a *"hello"* message to an identified sensor node. When receiving a *"hello"* message for it, the sensor node knows that it can send own sensing data, which is captured from connected sensors, back to the base station. Channel Activity Detection (CAD) feature of LoRa protocol can help to solve the problem in case of occurring a collision between several communications [176]. If the base station gets the data from the sensor node successfully, it will confirm by responding back an *"ack"* message. In case of occurring a sudden failure connection or corruption of data during a transaction, BS misses the data of that node.

There are a lot of benefits from applying this scheduling strategy in wireless sensor networks. This offers system can reduce energy consumption to extend battery life based on optimizing the sensing activities, especially in communications for data dissemination. For example, at a scheduling time, a node can wake up, read data from connected sensors, pack it into a message in a certain format, and then send to the data center. In another period of time, it is able either to do other tasks or turn into sleep mode for saving energy.

6.4 Validation and Revendications

We arranged an experiment on a simple LoRa star network consisting of a base station and two client nodes as shown in Figure 6.8. Three LoRa shields for Arduino manufactured by Froggy Factory [177] were used in this experiment. This Froggy's LoRa shield with LoRa chip SX1272 [178] can interface with the micro-controller of Arduino Uno via SPI bus. Distributed algorithms were developed to describe the local behavior of sensor node and then evaluated by simulators. The final code synthesis can port on executing hardware for testing in a real network.

Figure 6.9 illustrates the operation of the network. Appendix G and H introduce two Occam-pi libraries for LoRa which are developed during this work and Appendix I provides Occam-pi programs being used.

Distributed algorithms for sensing activities are developed and validated by simulators. Node's local behavior (Occam-pi program) can be interpreted on different target hardware platforms that is feasible thanks to TVM.

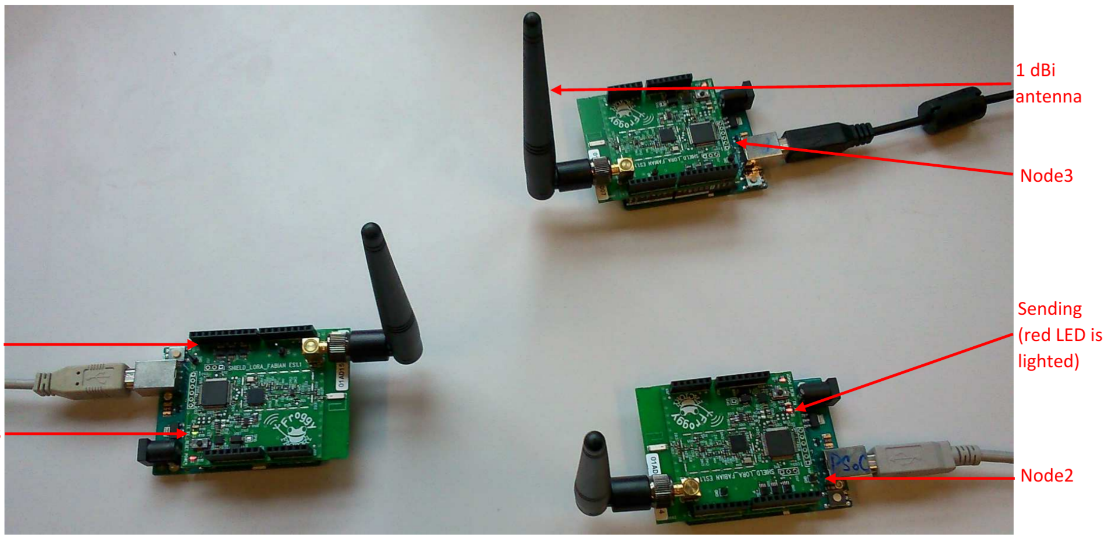
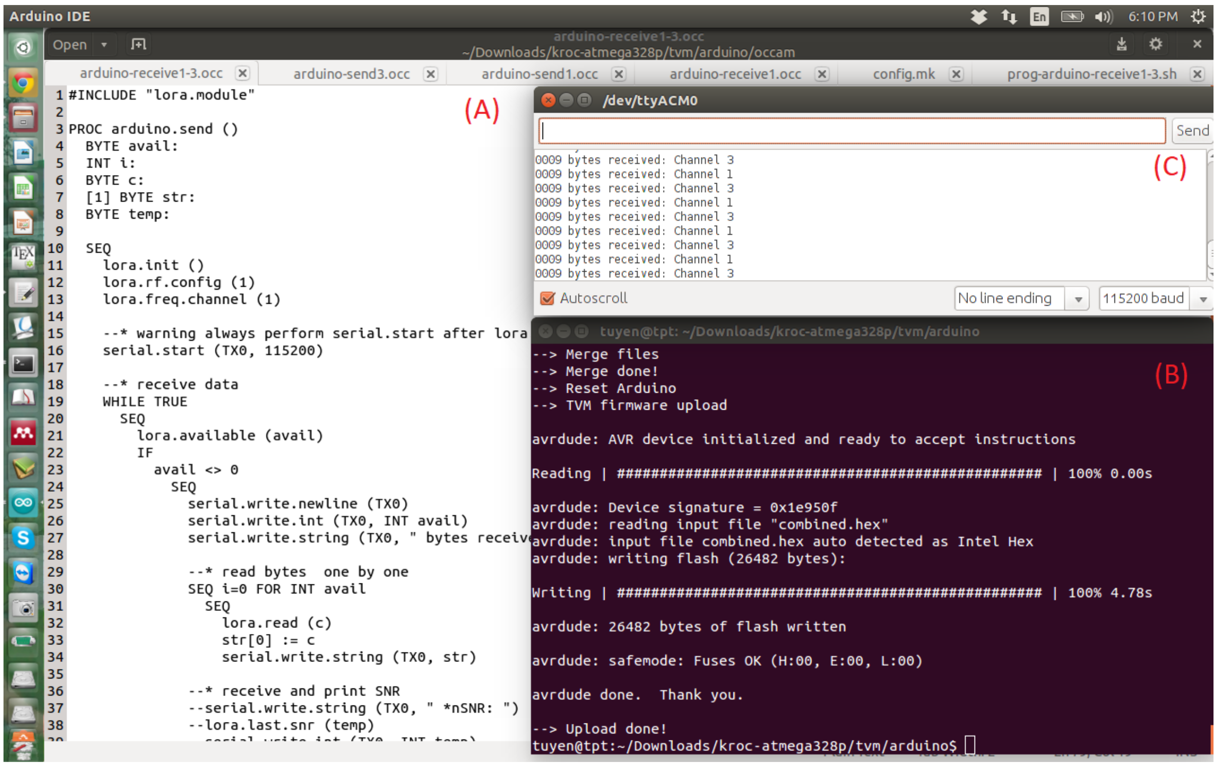

1 dBi antenna
Node3
Node1
Sending (red LED is lighted)
Receiving (green LED is lighted)
Node2

Figure 6.8 – Experiment on star network including a base station and two client nodes. Froggy Lora Fabian Shield (ES1.1) with SX1272 LoRa chip and 1 dBi antennas were used in this experiment.



```
#INCLUDE "lora.module"

PROC arduino.send ()
  BYTE avail:
  INT i:
  BYTE c:
  [1] BYTE str:
  BYTE temp:

  SEQ
    lora.init ()
    lora.rf.config (1)
    lora.freq.channel (1)

    --* warning always perform serial.start after lora
    serial.start (TX0, 115200)

    --* receive data
    WHILE TRUE
      SEQ
        lora.available (avail)
        IF
          avail <> 0
            SEQ
              serial.write.newline (TX0)
              serial.write.int (TX0, INT avail)
              serial.write.string (TX0, " bytes receive

              --* read bytes  one by one
              SEQ i=0 FOR INT avail
                SEQ
                  lora.read (c)
                  str[0] := c
                  serial.write.string (TX0, str)

              --* receive and print SNR
              --serial.write.string (TX0, " *nSNR: ")
              --lora.last.snr (temp)
```

```
0009 bytes received: Channel 3
0009 bytes received: Channel 1
0009 bytes received: Channel 3
0009 bytes received: Channel 1
0009 bytes received: Channel 3
0009 bytes received: Channel 1
0009 bytes received: Channel 3
0009 bytes received: Channel 1
0009 bytes received: Channel 3
```

```
--> Merge files
--> Merge done!
--> Reset Arduino
--> TVM firmware upload

avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.00s

avrdude: Device signature = 0x1e950f
avrdude: reading input file "combined.hex"
avrdude: input file combined.hex auto detected as Intel Hex
avrdude: writing flash (26482 bytes):

Writing | ################################################## | 100% 4.78s

avrdude: 26482 bytes of flash written

avrdude: safemode: Fuses OK (H:00, E:00, L:00)

avrdude done.  Thank you.

--> Upload done!
tuyen@tpt:~/Downloads/kroc-atmega328p/tvm/arduino$
```

Figure 6.9 – (A) A fragment of Occam-pi program developed for remote sensing. (B) Compile, upload and evoke Occam-pi program. (C) Terminal window shows received messages from two client nodes in round-robin scheduling.

Our proposed application model is suitable for both mesh and star networks. It is possible to extract statically at simulation time a table with all distances and expected communication times between radio transceivers in the system. This information could help to avoid collisions during dynamic network discovery. At this time, we were not able to valid this assumption for practical reasons.

Two Occam-pi libraries, which allow handling LoRa communications from Arduino boards, are developed. Some modification of TVM is done to mount LoRa chip as a peripheral. We did some experiments on a simple LoRa network to execute and evaluate the results of simulators. Time slices for communications of the round-robin scheduling can be optimized taking into account geographic complexity as mentioned in Appendix C.

7 Modeling and Optimization of Mobile Connections of Dynamic Sensor Field

Wireless Sensor Network (WSN) [79] is known as a network of sensors cooperatively operating in order to surveillance or collect the environmental parameters. In addition, a sensor field is included in a number of devices that can interact with one another and also to the environment. Most of existing wireless technique aims at short range application in the aspect of smart cities such as parking allocation, home services, and so on [4] [81]. Although the short transmission range can be compensated by applying a mesh topology, it would be economically infeasible to deploy in large geographic areas or behind obstacles (mountains, oceans, etc.). To overcome these disadvantages, long-range wireless sensor wireless sensor networks based on satellite communications have emerged as a promising technology with various applications.

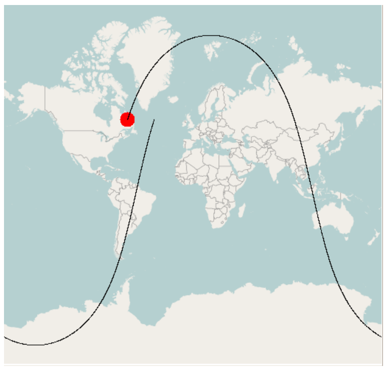


Figure 7.1 – LEO satellite's trajectory westward shifts due to the Earth's rotation [4].

In order to improve the connection time, it is necessary to choose proper gateways for the longest length of connection time. For this purpose, the connections between an LEO satellite and a dynamic sensor field should be presented by a graph-based model because it is convenient to determine the number of neighbors in the satellite's communication range and then apply the optimization algorithms [82] [86]. Moreover, BT-Graph model [179] based on ball-tree structure is efficiently support not only for searching the range nearest neighbors (RNN) but also finding the shortest path to a given target node [180] [181] [84] [182]. In this article, we propose a new approach, namely dynamic sensor field optimization model based on BT-Graph (BT-DYNSEN), to model and optimize the DSF in communication with LEO satellite.

In addition, with the development of Compute Unified Device Architecture (CUDA), general purpose Graphics Processing Unit (GPGPU) is widely used to implement parallel algorithms, which can speed up the large and complex processing tasks [136]. Nonetheless, serial graph algorithms are hard to be parallelized on GPUs due to irregular memory access and huge memory-intensive operations. For this reason, we also propose parallel algorithms to improve search speed based on BT-Graph model using NVIDIA CUDA GPUs. In our work, the experimental results were deployed on parallelization of RNN algorithm and Dijkstra's shortest path search algorithm.

The rest of this chapter is organized as follows. Section 7.1 presents the graph-based model for a DSF based on BT-Graph (BT-DYNSEN). How to optimize the DSF for a satellite connection is presented in Section 7.3. Implementation of BT-DYNSEN is described in Section 7.4. Section 7.5gives the simulation experiments on a forest fire surveillance network in Vietnam before a conclusion is drawn.

7.1 BT-DYNSEN

7.1.1 BT-DYNSEN model of DSF

BT-DYNSEN model of a dynamic sensor field (DSF) [79] based on BT-Graph [179] is a graph G(V,E). In this graph, set of vertices $V = \{v_i\}, i = 1..n$ corresponds to sensor nodes and set of edges, $E = \{e_j\}, j = 1..m$ are connections between the nodes with associated weight functions $W = \{w_j\}, j = 1..m$. The value of each w_j is given by Euclidean distance $d(v_i, v_j), i \neq j$. Additionally, $R = \{r_i\}, i = 1..n$ are the radii of the communication ranges of nodes [179] [180] [183]. An edge is established if and only if the distance between two nodes is less or equal to the minimum value of their communication radii, $d(v_i, v_j) \leq \min(r_i, r_j)$, $i \neq j$. Note that the terms *node* and *vertex* are used interchangeably in this article as a matter of convenience.

In Figure 7.2, for an example, a pair of vertices (v_5, v_6) has communication ranges r_5 and r_6 respectively. Because the distance between v_5 and v_6 is less than *r*, $d(v_5, v_6) < r$, so there exists an edge e_5 connecting them as can be seen in Figure 7.3. Similarly, the others edges of this graph namely $e_1 = e(v_1, v_2), e_2 = e(v_1, v_3), e_3 = e(v_2, v_3), e_4 = e(v_3, v_4)$ could be established. There are $2^n - 5$ edges between a pair of nodes of this graph that are not

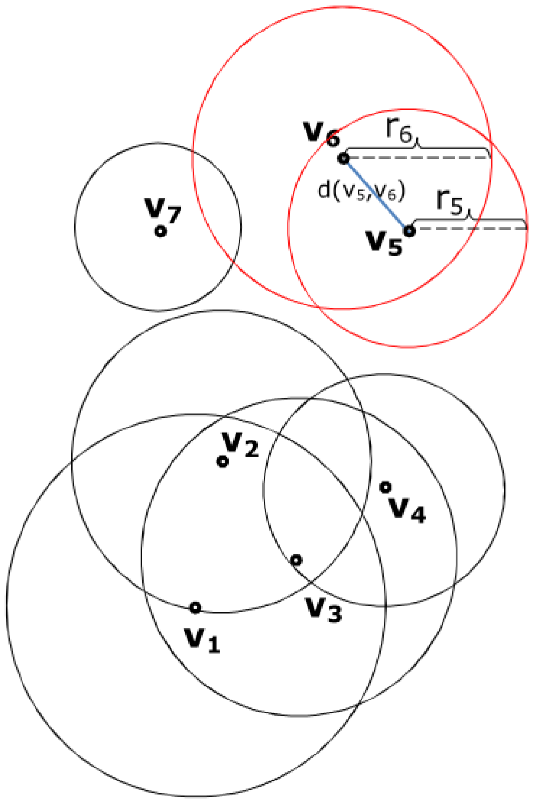


Figure 7.2 – A dynamic sensor field in which the communication ranges of sensor nodes are indicated by the radii of balls.

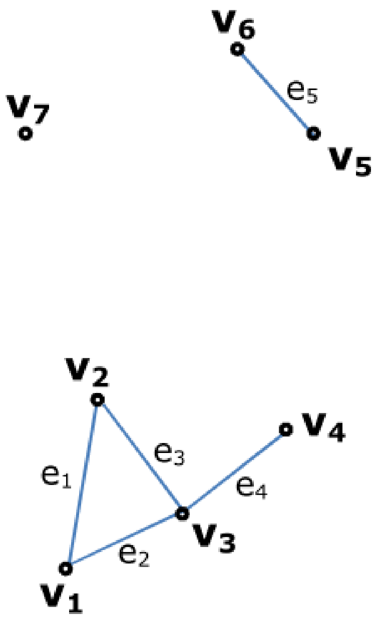


Figure 7.3 – A BT-Graph of the dynamic sensor field with 07 vertices $V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$ and 5 edges $E = \{e_1 = e(v_1, v_2), e_2 = e(v_1, v_3), e_3 = e(v_2, v_3), e_4 = e(v_3, v_4), e_5 = e(v_5, v_6)\}$.

existed due to inadequacy of the condition. The vertex v_7 is isolated because all distance values between it and the other nodes are inadequate to the condition.

7.1.2 BT-DYNSEN model of LEO satellite connection

The connections between an LEO satellite and a dynamic sensor field are also described in a BT-Graph. Let $V = \{v_i\}$, i=1..n, is a set of sensor node coordinates in a DSF. In this scenario, connection time is defined when any sensorset [79] of the DSF under the satellite coverage. The LEO satellite communication range is considered as a circle whose center is sub-point on the ground (sub-satellite point), s. It is noted that sub-satellite point, s, is where on the ground the straight line connecting the center of the Earth and the satellite meets the Earth's surface [105]. The associated BT-Graph consists of $n + 1$ vertices $P = \{V, s\} = \{v_1, v_2, ..., v_n, s\}$. If a node is within the communication range of the satellite, there exists an edge with weight given by the Euclidean distance between them. Otherwise edge weight is set to infinity. Consequently, the number edges of the graph are $m + n$ by adding n new edges $C = \{c_1, c_2, ..., c_n\}$ with $c_i = c(s, v_i), i = 1..n$. The weights of the n new edges are denoted by $Z = \{z_1, z_2, ..., z_n\}$. In this case the set of edges is $R = \{E, C\} = \{e_1, e_2, ..., e_m, c_1, c_2, ..., c_n\}$ and the set of corresponding weights is $Q = \{W, Z\} = \{\{w_k\}, \{z_i\}\}$ with $k = 1..m$, $i = 1..n$.

Table 7.1 – An example of satellite connection data with three rows: *Connection number*, *Connection vector* and *Time vector*.

Connection number	1	2	3	...
Connection vector	v_1	v_3	v_4	...
Time vector	t_1	t_2	t_3	...

Hence the BT-DYNSEN model for LEO satellite connections is presented by a graph G(P,R) with weight functions Q.

Furthermore, during the connection time only one sensor node (vertex) of a dynamic sensor field is chosen to connect with the sub-satellite point (center vertex) at one time [79]. A number of different nodes could be chosen based on the value of edge weights at different times. To manage the connections, the name of chosen node is kept in *Connection vector* [79] and the corresponding time is saved in *Time vector* [79] (as in Table 7.1).

Table 7.1 shows that in *connection 1*, center vertex s connects with v_1 at time t_1. In a similar way, in *connection 2* at time t_2 and *connection 3* at time t_3, v_4 and v_2 are chosen to connect with s respectively.

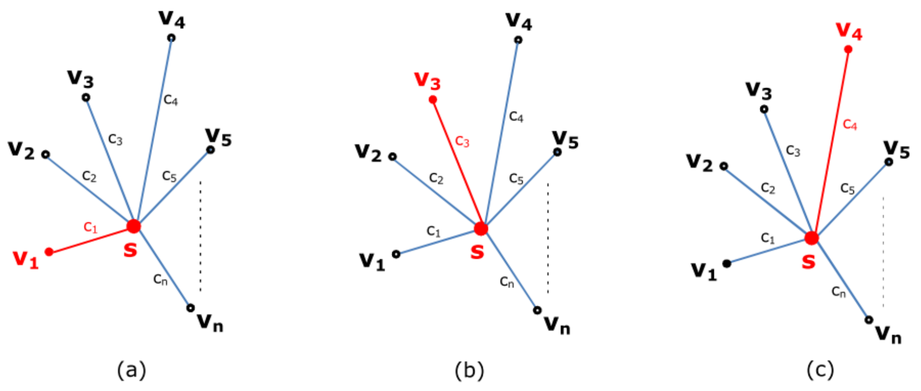


Figure 7.4 – The BT-Graph of a dynamic sensor field in three different connections (solid red lines) at times t_1, t_2 and t_3.

For instance, Figure 7.4 shows three graphs of the dynamic sensor field in three different connections with the center vertex s (a sub-satellite point) at different times t_1, t_2 and t_3. At time t_1 (Figure 7.4(a)), vertex v_1 is chosen and edge c_1 is established. Similarly, in Figure 7.4(b) and Figure 7.4(c) vertex v_3, v_4 are chosen for connections that leads to corresponding edges c_3, c_4 are established at time t_2 and t_3 respectively.

7.1.3 Computation of connection time

In this section, we describe the way to calculate connection time, T_{ij}, between an LEO satellite and a gateway of DSF, v_j [79] [105]. Similarly, the calculation could be applied to all other nodes. Note that every node of the sensor field is assumed as a gateway for the connection with the satellite in calculating the values of connection time.

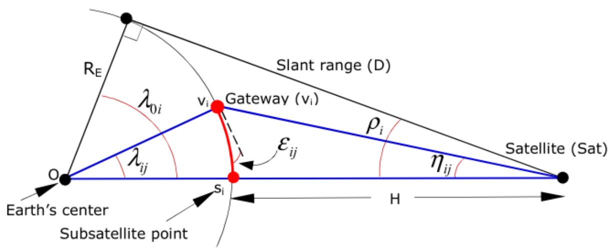


Figure 7.5 – Gateway of sensor field geometry [105].

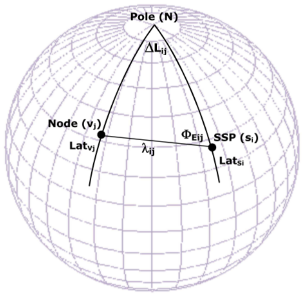


Figure 7.6 – Relationship between sensor node of a DSF and sub-satellite point [105].

First, it is necessary to define the angles and related distances between satellite, a gateway on the ground and the Earth's center. The parameters are indicated in Figure 7.5. For an angular radius of the spherical Earth, ρ_i, and the angular radius λ_{0_i} can be found from relations

$$\sin(\rho_i) = cos(\lambda_{0_i}) = \frac{R_E}{R_E + H} \tag{7.1}$$

$$\rho_i + \lambda_{0_i} = 90\ deg \tag{7.2}$$

where R_E = 6378.14 km is the Earth's radius and H is the altitude of the satellite above the Earth's surface.

With the coordinates of a sub-satellite point, s_i $(Long_{s_i}, Lat_{s_i})$ along each satellite's ground track and a sensor node, nodes v_j of the DSF $(Long_{v_j}, Lat_{v_j})$, and defining $\Delta L_{ij} = |Long_{s_i} - Long_{v_j}|$, the azimuth, $\Phi_{E_{ij}}$, measured eastward from north, and angular distance, λ_{ij}, from the sub-satellite point to the sensor node (see Figure 7.6) are given by

$$\begin{aligned} cos(\lambda_{ij}) =& sin(Lat_{s_i})sin(Lat_{v_j}) + \\ & cos(Lat_{s_i})cos(Lat_{v_j})cos(\Delta L_{ij}) \end{aligned} \tag{7.3}$$

$$\begin{aligned} cos(\Phi_{E_{ij}}) =& ((sin(Lat_{v_i}) - \\ & cos(\lambda_{ij})sin(Lat_{s_i}))/(sin(\lambda_{ij})cos(Lat_{s_i})) \end{aligned} \tag{7.4}$$

where $\Phi_{E_{ij}} < 180\ deg$ if v_i is east of s_i and $\Phi_{E_{ij}} > 180\ deg$ if v_i is west of s_i Consider triangle Os_iv_j (in Figure 7.5), the distance, d_{ij}, between s_i and v_j can be found using the law of cosines:

$$d_{ij}^2 = R_E^2(1 - cos(\lambda_{ij})) \tag{7.5}$$

The *connection time*, T_{ij}, is given by

$$T_{ij} = (\frac{P}{180\ deg}) \arccos(\frac{\cos(\lambda_{ij_{max}})}{\cos(\lambda_{ij_{min}})}) \tag{7.6}$$

where P is the *orbit period* in minutes. From Equations (7.5) and (7.6), the communication duration, T_{ij}, strongly depends on how close the nodes v_j is to the sub-satellite points s_i, the distances d_{ij}, along the ground track on any given orbit pass [105].

7.1.4 Definition the operation modes of DSF

In order to describe the behaviors of a dynamic sensor field, we propose two definitions as follows:

Definition 1 (Passive mode). Passive mode of a dynamic sensor field is established when a dynamic sensor field automatically collects and prepares the environmental data to sent before visiting an LEO satellite. Each time the satellite is perceived, the gateway of the sensor field will establish the connection and then send the data. All processes of a dynamic sensor field are programmed and repeated systematically.

Definition 2 (Active mode). Active mode of a dynamic sensor field is established when a dynamic sensor field has the ability to response satellite's commands before the end of a contact. In this mode, an LEO satellite visits a dynamic sensor field and establishes a connection with a gateway. The satellite then sends a command to the gateway for control purposes and/or collecting environmental data from DSF. After a certain period of time, it expects to receive the feedback data also via a gateway. Hence, there is a pair of gateways: *Input gateway* for receiving satellite's command at the starting time and *Output gateway* for sending data to satellite before the satellite leaving.

7.1.5 Constraint

The altitude of an LEO satellite must be in the range from 275 km to 1400 km due to atmospheric drag and Van Allen radiation effects [105]. Besides, the experimental results were announced by High Altitude Society in the United Kingdom that LoRa SemTech transceivers can communicate in distance up to 600 km in an environment without any obstacle and 20-40 km in urban area [163]. Based on these factors, the maximum communication range for all nodes in long-range sensor fields is 40 km in this work. LEO's satellites are chosen in our experiments must have the orbit altitude of less than 600 km. Furthermore, the satellite's relative speed over a fixed point on the Earth's surface must be around 7.5 - 8.0 km/sec [105]. The speed of the satellite is calculated by

$$v = \sqrt{\frac{Gm_E}{R_E + H}} \tag{7.7}$$

Equation 7.7 shows that the speed of the satellite in orbit is in inverse proportion of its altitude [105]. Where G is universal gravitational constant ($G = 6.67\text{x}10^{-11}$ Nm^2/kg^2) and m_E is the mass of the Earth ($m_E = 5.98\text{x}10^{24}$ kg). With orbit altitude of satellite in range 300-600 km, the speed of satellite on orbit must be in range 7.56-7.73 km/sec.

7.2 Optimization method

7.2.1 Verification of the connectivity of DSF

In this work, the top-down construction algorithm is chosen in order to build a ball-tree structure for verifying the network connectivity of the DSF. In this manner, the time complexity can be found in $O(nlog^2n)$ [183].

Algorithm 13 Verification the network connectivity of a DSF

```
 1: procedure VERIFYCONNECTIVITY(D)
 2:     if D.hasAPoint() then
 3:         Create a leaf B is a point in D
 4:         Return B
 5:     else
 6:         Let c is the dimension of greatest spread
 7:         Let L, R are the set of points lying to 2 subsets
 8:         Let r is a radius of ball
 9:         Create B with two children:
10:             B.center $\leftarrow$ c
11:             B.radius $\leftarrow$ r
12:             B.leftChild $\leftarrow$ balltreeConstruct(L)
13:             B.rightChild $\leftarrow$ balltreeConstruct(R)
14:     end if
15: end procedure
```

Algorithm 18 to construct the ball-tree is a recursive process from top to down. The process at each step is to choose the split dimension it and then split the set of values into two subsets. First, it is necessary to choose the point in the ball which is farthest from its center,p_1, and then choose a second point p_2 that is farthest from p_1. It is followed by assigning all points in the ball to closest one of two clusters corresponding to p_1 and p_2. Finally, the centroid of each cluster is calculated and the minimum cluster radius for enclosing all the points are determined. This process is stopped only when the leaf balls contain just two points. Note that D denotes current considered ball-tree structure and B denotes leaf node defined after each process.

7.2.2 Determination the sensorsets corresponding to each sub-satellite point

Defining a sensorset [79] based on BT-DYNSEN is to find k nearest sensor nodes (neighbors) under the satellite's coverage area at each sub-satellite (Algorithm 14). It is a depth-first traversal algorithm for traversing tree or graph data structures, starting with the root node. The value of Q is updated during the search process. At each considered node B, Q obtains k points which are the nearest query q as the following the algorithm.

Algorithm 14 Determination a sensorset

```
procedure DETERMINESENSORSET
    if balltreeNode.isALeaf then
        if d < query.range then
            Write name of balltreeNode to file
        end if
    else
        if d < (query.range + radiusNode) then
            left ← balltreeNode.leftChild
            right ← balltreeNode.rightChild
            rnn: left, query
            rnn: right, query
        end if
    end if
end procedure
```

Table 7.2 – The weights of connection items in a DSF with 4 nodes.

Connection items	Weights
(v_1, v_2, v_4)	1
(v_1, v_2, v_3)	1
(v_1, v_3)	3
(v_1, v_3, v_4)	1

There are two different cases in *BT-DYNSEN* algorithm as follows. If current considered node is a leaf node and the distance from query point *q* to *B* is less than *r* ($d < r$), the obtained result is updated by adding B into *Q*. Otherwise, if *B* is not a leaf node and the distance from query point *q* to *B* is less than the total of *r* and the radius of B ($d < r + B.radius$), it is necessary to perform recursive algorithm for the two child nodes of a parent node B: left-child and right-child.

7.2.3 Selecting the gateways of DSF

If a DSF V has n nodes, there are $2^n - 1$ connection items, $G = \{g_l\}$, l=1..$(2^n - 1)$. It is necessary to find out a set of proper nodes which play as gateways of DSF to provide the longest length of time for the connection. To do this, the *association analysis algorithm* [181] is applied. This way, a DSF is represented in a binary format, where each row corresponds to a connection item and each column corresponds to a node. A node value is one if the node appears in a connection item and zero otherwise. The weight of a connection item determines how often a connection item is applicable to a set of connection items. The weight of a connection item, $g_i \in G$, can be defined as $w(g_i) = \mid \{g_l \mid g_i \subseteq g_l, g_l \in G\} \mid$, where the symbol | . | denotes the number of elements in a set.

Algorithm 15 for selecting the gateways of the DSF is briefly presented in the following list.

Algorithm 15 Selecting the gateways of a DSF

```
Input: list of sensorsets
Output: gateways of DSF
 1: $C_k$: candidate sensors (size k)
 2: $L_k$: set of selected gateways (size k)
 3: procedure SELECTGATEWAY
 4:     $L_1 \leftarrow \{weight1 - sensorsets\}$
 5:     for (k=2;Lk.count>0;k++) do
 6:         $C_{k+1} \leftarrow$ generate candidate sensors $L_k$
 7:         for each $transaction\ t \in data$ do
 8:                 Increment count of the candidate sensors in $C_{k+1}$ that contained in t
 9:         end for
10:         $L_{k+1} \leftarrow$ candidate sensors in $C_{k+1}$
11:     end for
12:     Write counted frequent of all sensors $L_k$ to file
13: end procedure
```

For example, a DSF with 4 nodes, $V = \{v_1, v_2, v_3, v_4\}$ is shown in Figure 7.7. There are 4 sensorsets $A_1 = \{v_1\}$, $A_2 = \{v_1, v_2, v_3\}$, $A_3 = \{v_2, v_3, v_4\}$ and $A_4 = \{v_3, v_4\}$. The weights of connection items are presented in Table 7.3. The connection with the highest weight corresponds to the least number of times required to change the gateways. It leads to the connection duration time of the connection is longest. Because the weight of connection item (v_1, v_3) is highest, this connection is chosen.

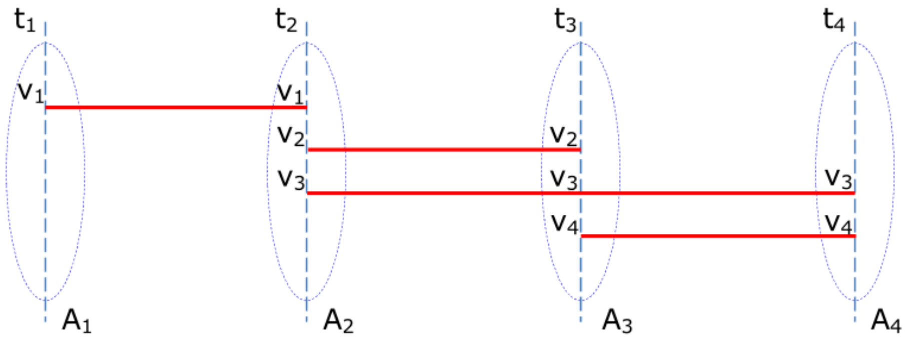


Figure 7.7 – The connections between a 4-node DSF with an LEO satellite.

7.2.4 Finding shortest path for data dissemination

The problem of finding the shortest path from each sensor node of DSF to the gateway can be solved by a graph search method. The algorithm is presented as Algorithm 16.

Algorithm 16 Finding shortest path from each sensor node of DSF to the gateway.

```
procedure FINDSHORTESTPATH
    Init matrix M
    $S \leftarrow$ Start point
    $T \leftarrow$ End point
    *d* is an array
    *free* is an array
    *trace* is as array
    $d[S] \leftarrow 0$
    while TRUE do
        u ← -1
        min ← INFINITE
        for v ← 0 to n-1 do
            if $free[v]$ AND $(d[v] > (d[u] + M[u, v]))$ then
                $d[v] \leftarrow d[u] = arr[u][v]$;
                $trace[v] \leftarrow$ u;
            end if
        end for
        if (u=-1) OR (u=T)) then
            Break
        end if
        $free[u] \leftarrow$ false
        for (v ← 0 to n-1 do
            if free[v]$AND$(d[v]>(d[u]+M[u,v])) then
                $d[v] \leftarrow d[u] = arr[u][v]$;
                $trace[v] \leftarrow$ u;
            end if
        end for
    end while
end procedure
```

The algorithm proceeds in three following steps. First, the balltree graph-based model, G, is constructed. At the next step, the weight matrix M for edges connect each pair of vertices (sensor nodes) in V is computed based on their coordinates. For v_i, v_j are two different vertices in V, in case of $v_i \equiv v_j$ the edge weight is 0. If $v_i \not\equiv v_j$, the edge weight is given by $d(v_j, v_j)$ if $d(v_j, v_j) \leq r$, otherwise it is infinity, ∞. Finally, the shortest path from q to e in the weight matrix is figured out. Note that q is starting point (a sensor node) and e is the destination point (the gateway of DSF).

7.2.5 Parallelization of RNN algorithm to determine sensorsets

As mentioned in section 7.2.2, range nearest neighbors (RNN) search method on BT-Graph can be used to determine the set of nodes in a sensorset associated with each position of the satellite along the ground track. In this section, a greedy algorithm is proposed to nearest neighbors search running on CUDA (namely RNN-CUDA as shown in Algorithm 17). The process of this greedy algorithm consists of two part. One part is parallel processes on GPUs to compute distances from the query the point to all nodes of a dynamic sensor field. Another part runs on CPU to sort the obtained distances in ascending order and then pick up *k* neighbor nodes associated with shortest distance values.

Algorithm 17 RNN-CUDA

```
 1: ▷ Host program executed on CPU
 2: Load data into global memory
 3: Assign the value of k
 4: Copy data from CPU $\rightarrow$ GPU
 5: ▷ Parallel programs executed on GPUs
 6: Compute distances from query point, q, to all nodes $v \in V$
 7: Copy data back from GPU $\rightarrow$ CPU
 8: ▷ Host program executed on CPU
 9: Sort the distance values as a ascending list
10: Pick up k nearest neighbor nodes associated to k first value of the list
11: Release GPU memory
```

7.2.6 Parallelization of search algorithm to find the shortest path for data dissemination

To parallelize Dijkstra algorithm, we propose that each edge of BT-Graph is handled by a thread of CUDA. Figure 7.8 depicts the flowchart of the parallel Dijkstra algorithm for searching BT-Graph model in which two procedures, *Extract min*, and *Update cost*, are performed on NVIDIA GPUs. The parallelization of Dijkstra's algorithm is illustrated in Algorithm 18.

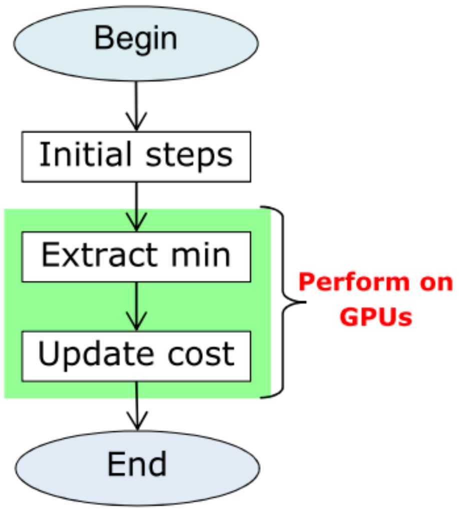


Figure 7.8 – The flow chart of parallel Dijkstra algorithm for searching BT-Graph model.

7.3 BT-DYNSEN implementation

7.3.1 Block diagram of DYNSEN tool

We have developed the BT-DYNSEN tool in C/C++, that enables to model and optimize the connection time between a dynamic sensor field with an LEO satellite. GPredict [184] is used to provide the information about BEESAT-3 satellite's path. Besides, NetGen tool [185] is utilized to generate the abstract network of a 50-node dynamic sensor field from geographic data provided by Google maps service. The obtained results are the nodes of the DSF which should be configured as gateways for the best connection duration time and the shortest paths for data dissemination from each node to these gateways.

Figure 7.9 illustrates the brief view of the process for optimizing the connection time between an LEO satellite and our proposed dynamic sensor field. It shows that DYNamic SENsor field (DYNSEN) tool consists of four main functional blocks: data normalization, data input, satellite communication model and connection time optimization block. These functional blocks operate under a set of constraints that was discussed in section 7.1.5. The details of the other blocks are later described in next sections.

7.3.2 Data normalization

The function of this block is to normalize the input data for building up the input data for the next step. First of all, the structure of the sensor field is generated by NetGen. After reforming, the coordinates of the sensor in the field are used to build up the node location table. Based on the topology of network interaction, routing tables are also carried out. GPredict provides the data about satellite orbit such as altitude, inclination angle etc., and satellite ground track parameters.

Algorithm 18 DijkstraCUDA

```
1: Step 1:
   Initialize weight matrix (input vectors), start point and stop point
2: Step 2:
   Allocate vectors in device memory corresponding to each edge of BT-Graph
3: Step 3:
   Copy vectors from CPU (host) memory to GPU (device) memory
4: Step 4:
   - Search for a free vertex $u$ that has the least cost from the starting vertex $S$ to $u$
    + If no vertex $u$ is found, either it exists a path or not
    + Otherwise, from $u$ we continue to consider the other free vertices
   - Use CUDA to specify the thread for searching the paths between the vertex $u$ and the others
5: Step 5:
   Copy results from GPUs memory to CPU memory
6: Step 6:
   Free device memory and host memory
```

Table 7.3 – The orbit 12794 of BEESAT-3 after being reformed.

Time	Lat	Lon
17/08/2015 15:58:24	-9.81	117.00
17/08/2015 15:58:34	-9.24	117.23
...	...	...

7.3.3 Input data and Satellite communication model

For experiments, an abstract structure of the long-range sensor field for fire forest surveillance was generated by using NetGen [185]. Figure 7.10(a) shows dynamic sensor field consists of 50 sensor nodes that are stretched from South Central Coastal to Southeast and extended up to Mekong River Delta in Vietnam.

From the constraints about the satellite's orbit altitude in section 7.1.5, orbit 12974 of BEESAT-3 [186], an LEO satellite with orbit altitude is around 575 km, which was chosen in our experiments. The ground track data stored in a plain text (.txt file as shown in Table 7.3) was used as input data.

The satellite's parameters specify the characteristics of not only its operations orbit but also the ground track. All these data are used as input data for optimizing the length of time for connection. The satellite communication is built to describe the direct connection between an LEO satellite and a long-range sensor field via radio links. This model is based on the formulas which are mentioned in details in section 7.1.3.

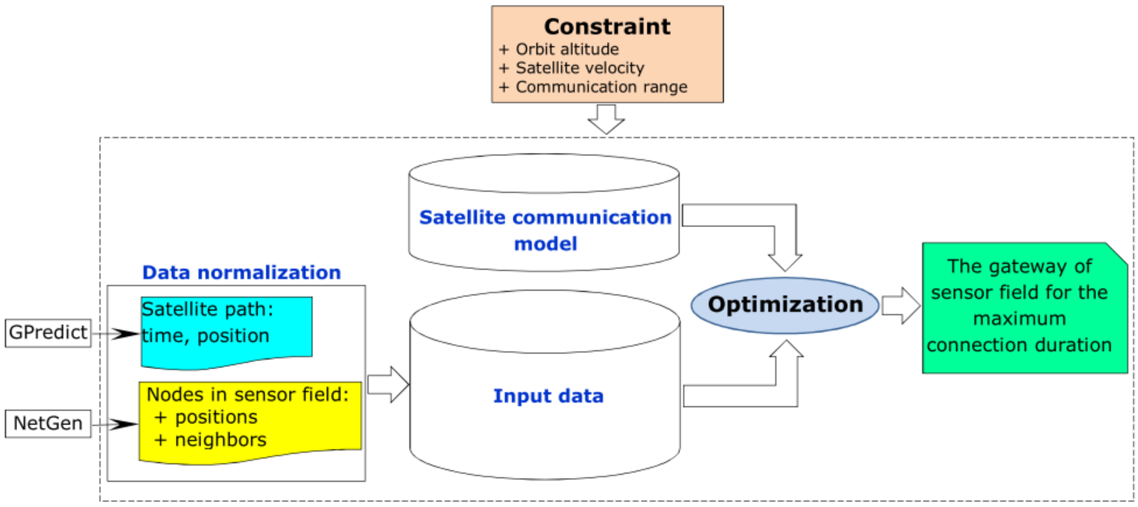

Constraint
+ Orbit altitude
+ Satellite velocity
+ Communication range
Satellite communication model
Data normalization
GPredict
Satellite path: time, position
NetGen
Nodes in sensor field: + positions + neighbors
Input data
Optimization
The gateway of sensor field for the maximum connection duration

Figure 7.9 – The block diagram of BT-DYNSEN implementation.

7.3.4 Connection time optimization

The optimization process aims at finding out the proper gateway of the sensor field for maximum connection duration time based on input data about each sensor node in sensor field coordinates and satellite's ground track pass. The list of connection duration time which is calculated according to the relative distance between node coordinates and the sub-satellite coordinate. A node will become the gateway if its duration time is the maximum value.

7.4 Experiments

7.4.1 Experiment 1: Verification of network connectivity

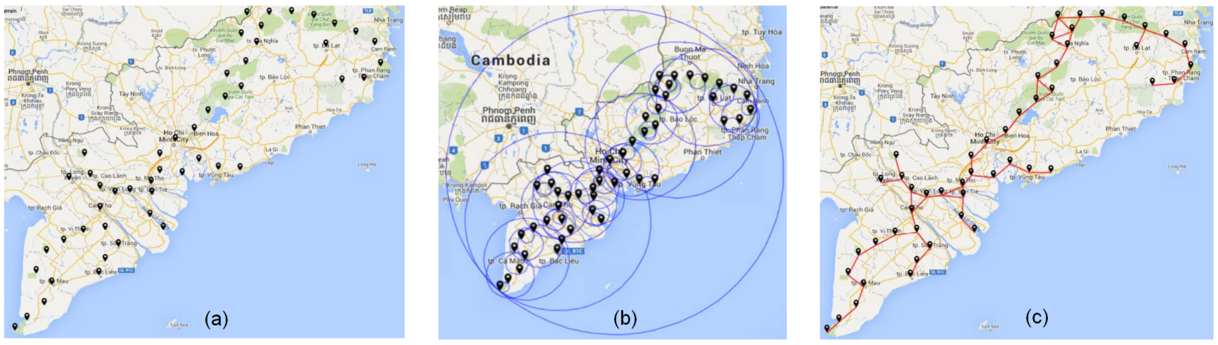
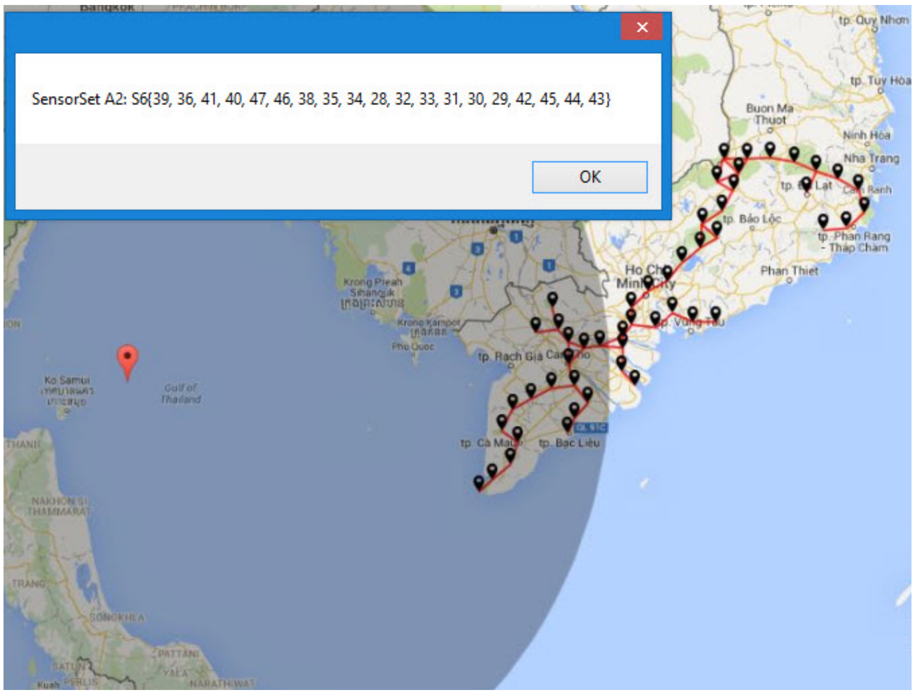


Figure 7.10 – Verify dynamic sensor network connectivity. (a) A 50-node dynamic sensor network, (b) Balltree structure of the DSF, (c) Connectivity of the DSF.

The connectivity of the 50-node dynamic sensor network was verified by applying BT-

DYNSEN as shown in Figure 7.10(a). Figure 7.10(b) depicts the ball-tree structure of this DSF in which there are 49 balls with radii from 10.124 m to 321.165 m. BT-Graph model then was utilized to ensure the full connectivity of all network nodes, the radius 30.497 m was then chosen as shown in Figure 7.10(c).

7.4.2 Experiment 2: Determination of the sensorsets



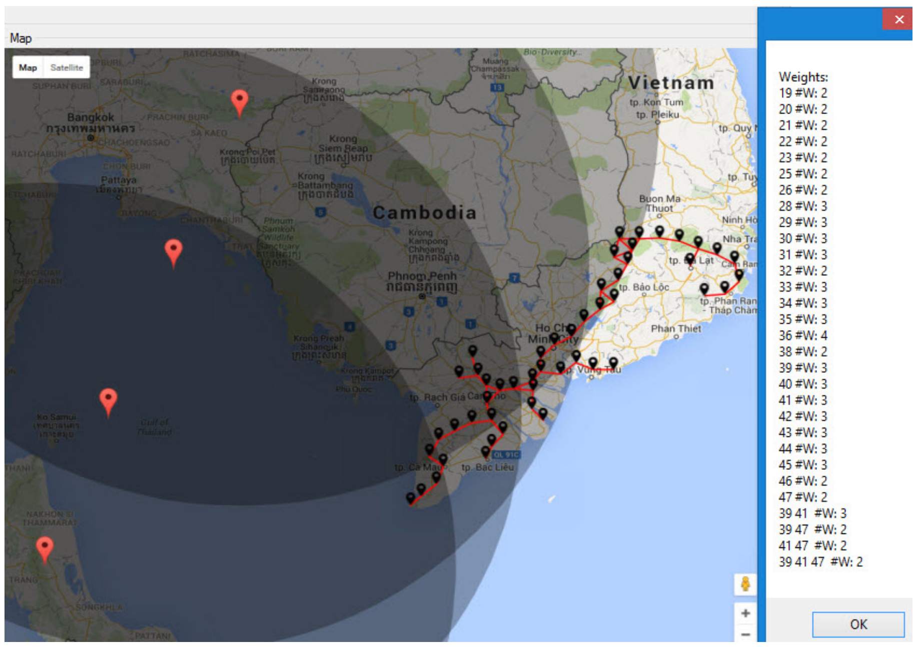
```
SensorSet A2: S6{39, 36, 41, 40, 47, 46, 38, 35, 34, 28, 32, 33, 31, 30, 29, 42, 45, 44, 43}
```

Figure 7.11 – Determine the sensorsets corresponding to each sub-satellite along the ground track of BEESAT-3 in orbit 12794.

BT-DYNSEN tool was employed in determining sensorsets corresponding to sub-satellite points during visiting time. The map in Figure 7.11 shows the sensorset was determined with sub-satellites of BEESAT-3 at (latitude: 14.00, longitude: 102.48) in orbit 12794. Sub-satellite and satellite coverage were indicated by a solid red square and red circle respectively. The sensorset consists of 19 sensor nodes which were under the satellite's coverage area (the shadow area). There are four sensorsets were created along the BEESAT-3's ground track in orbit 12794.

7.4.3 Experiment 3: Selecting gateways

With each sensorset, a subset of connections is established. The weights of each connection in the subset are then computed. A set of connection items is created by combining these subsets. The best connection is chosen based on the weights of connection items. For instance, in Figure 7.12 node v_{36} was chosen as the gateway of the 50-node DSF to connection with BEESAT-3 satellite in orbit 12794 because its weight is highest in sensor nodes.



Weights:
19 #W: 2
20 #W: 2
21 #W: 2
22 #W: 2
23 #W: 2
25 #W: 2
26 #W: 2
28 #W: 3
29 #W: 3
30 #W: 3
31 #W: 3
32 #W: 2
33 #W: 3
34 #W: 3
35 #W: 3
36 #W: 4
38 #W: 2
39 #W: 3
40 #W: 3
41 #W: 3
42 #W: 3
43 #W: 3
44 #W: 3
45 #W: 3
46 #W: 2
47 #W: 2
39 41 #W: 3
39 47 #W: 2
41 47 #W: 2
39 41 47 #W: 2

Figure 7.12 – Select a set of nodes which play as gateways of the DSF according to the weights of connections.

7.4.4 Experiment 4: Routing optimization for DFS

To ensure sensing data to be collected from all sensor nodes and then sent before the satellite leaving, it is necessary to find the shortest path from each sensor nodes to the gateway. The interconnection weights within the DSF are geographic distances between each pair of nodes that were carried out by applying BT-DYNSEN model. Figure 7.13, as an example, illustrates the chosen path (the bold blue line) for data dissemination from sensor node v_{50} to the gateway node v_{36}.

7.4.5 Experiment 5: Parallel RNN algorithm

In order to make the comparison between the parallel algorithms and serial ones for searching BT-Graph, we have run the simulations on a computer which is equipped with an INTEL ®Core i3-3220 3.3 GHz processor, 4 GB DDR3 SDRAM, and an NVIDIA GeForce GTX660 graphic card. In our experiment, a comparison about BT-Graph's search speed is obtained by performing both sequential algorithms on CPU and parallel algorithm on GPUs. On each algorithm with the same input data, which are generated randomly by the program, an experimental program is executed in 10 times for getting the average value of execution time.

The results are illustrated in Table 7.4, where N is the number of initial data points and Q is a number of query points. It shows that the search speed of RNN-CUDA is improved around 2.37 folds compared with that of RNN BT-Graph.

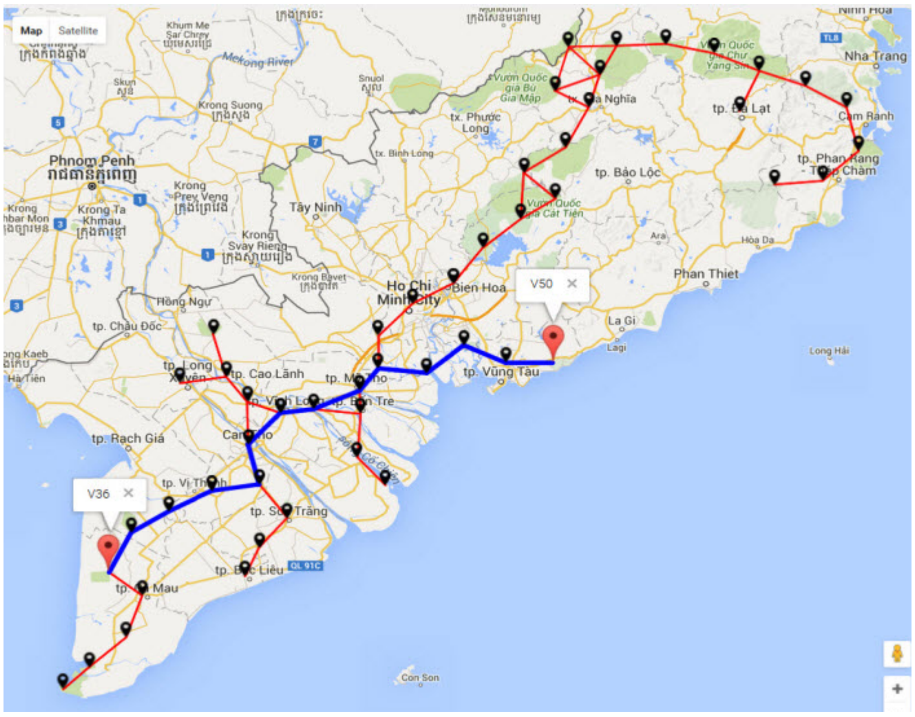


Figure 7.13 – The shortest path for data dissemination from node v_{50} to gateway node v_{36}.

Table 7.4 – Comparison on the execution time of RNN BT-Graph and RNN-CUDA.

	N=10000, Q=10000	N=100000, Q=100000
RNN BT-Graph	179.46	17898.13
RNN-CUDA	123.47	5440.29

7.4.6 Experiment 6: Parallel Dijkstra algorithm

The purpose of this experiment is to compare the search speed of traditional Dijkstra for BT-Graph and the corresponding speed of parallel Dijkstra (Dijkstra CUDA). The first step is to construct the weight matrix of BT-Graph model. Next, to obtain the average execution time, sequential Dijkstra's algorithm and parallel one are run in turns in 10 times. These experimental results are illustrated in Figure 7.14.

As can be seen in Figure 7.14, sequential processing is efficient for a small a number of nodes because it not waste extra time copying the data between CPU and GPU memories. Nevertheless, when a number of nodes are increased more than 8000, the duration time required for processing can be reduced significantly by using techniques of parallelism based on CUDA.

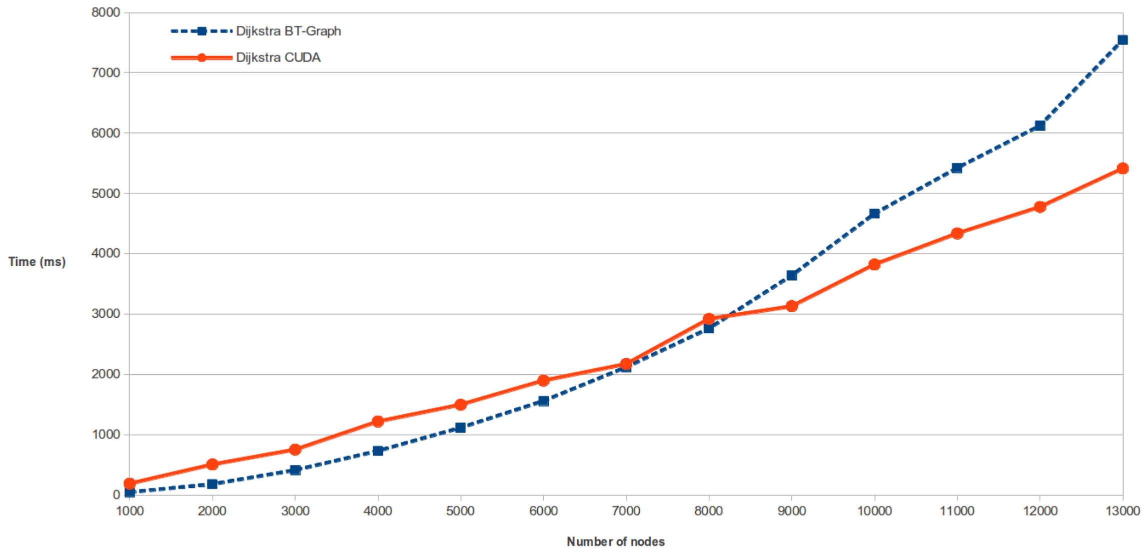


Figure 7.14 – Performance comparison between sequential and parallel processing of Dijkstra's shortest path algorithm on BT-Graph, blue line and red line respectively.

7.5 Conclusion

Based on BT-Graph, we have described a new approach in order to model and optimize the dynamic sensor field for LEO satellite connections. The distances between the sub-satellite points and each node of the sensor field are utilized as a key factor in finding out the proper gateways for the longest connection time. The experimental results were obtained by applying several appreciate algorithms on BT-DYNSEN model to verify the connectivity of the network, determine sensorsets at visiting time, choose set of gateway nodes and find the shortest path for data dissemination in DSF. In order to improve the execution time, we implemented the corresponding parallel algorithms with CUDA on GPU and analyzed its performance. The results show that the parallel algorithm on GPU is considerably superior to the serial one on CPU when BT-Graphs comprising more than 8000 vertices. Thus, our proposed graph-based model helps to increase the amount of time for data communications in long-range sensor field applications using satellite connections.

8 Conclusion and Perspectives

8.1 Conclusion

The UBO tool-set consisting of NetGen, QuickMap, and PickCell is a framework aiming at modeling and facilitating simulations of physical phenomena, with wireless sensor network deployments as an example.

A top-down approach enables to the model abstract presentation of WSNs, generate automatically network descriptions, develop and then evaluate distributed algorithms by invoking simulations on multi-cores CPU or GPUs, and finally synthesis code to execute on hardware in a real system.

Even though wireless sensor networks offers obvious advantages in all aspects of contemporary social life, it still exists critical issues in deployment for high optimization such as scalability, capability, security, and lifetime. The approach, followed to deal with this issue, is to perform a simulation with formal models, high-performance simulators based on parallel distributed algorithms and experimental evaluations.

The certain advantages in terms of the simulation are evident when modeling a WSN as a concurrent system. First, each process in the simulation can describe exactly a node's behaviors in a real network. The global knowledge system could be also obtained at the convergent time relied on exchange information among processes. Second, this approach facilitates the synthesis of execution code for local behaviors at the node level. Third, considering WSNs as the networks of concurrent processes helps to enhance simulation execution by utilizing massive parallel processing.

Information system, as shown in Figure F.1, depicts the relationship between units such as models, observations, and so on. This flow presents clearly the roles of these elements in order to motivate the future research of geographic issues. Analysis and decision-making taking into account geomorphological complexity must utilize the power of massively parallel simulations and observations.

Although the discretization of physical functions might cause errors the cellular automata method is an efficient approach for physical simulations because of its flexible rules for evolution, especially for dynamic complex systems. In addition, by integrating observations the CA model has emerged as a tool to improve physical models.

Computations of water distribution and radio coverage regarding terrain complexity play a critical role in flood risk forecasting and floodplain management. In this study, even though our model of flash flood was based on several hypotheses to simplify simulation conditions, observed data can help to refine model parameters of physical models. These studies need further consideration on the correctness of simulation results in relation to the accuracy of geographical terrain data, cell size selection, rules of water spreading, etc.

Our study on dynamic routing of sensor networks based on an assumption that LoRa technology is feasible to establish the direct radio links between sensor nodes deployed on the ground and LEO satellites. It now is available with Lacuna project [1]. Our proposed tool-flow allows modeling and optimizing the dynamic routing of the nodes to improve the efficiency of air communications which are well suited for remote sensing applications based on low-cost small satellites. It certainly needs further actual experiments for calibration parameters of the model and validation requirements.

HLA combining and sequencing several simulations was studied to demonstrated that physical systems could reproduce routing algorithm similar to those of numeric networks. An example being shown in Appendix J was a surveillance network to warn about invasive waves.

8.2 Perspectives

Main research opportunities opened by this work appear as follows:

- At the first rank we see contributions to tools for LoRa deployment and integrated software binding high level sensing application to effective distributed networks. Rationalities built in this thesis provide deterministic conditions to do this (Chapter 6).

- Synthesizing long-range mesh radio networks would allow to cover very large surfaces. Tuning parallel algorithms can help to overcome the sparse conditions that will appear in this situation.

- Line-of-sight propagation and water spreading are examples of common spatial behaviors found in nature. Many biological, mechanical or physical processes have similar properties. Sensing of environment could take benefits from generic frameworks helping to develop and compose such simulations.

[1]http://lacuna.space/

- Applicability for large problems available for the public can be obtained by publishing web interfaces to services dependent on two separated concerns: zone selection and cell resolution, and cellular system libraries.

With the advent of LoRa technology which facilitates long-range star network deployment and maintenance. Other aspects of network management such as the limits of duty cycle depending on regional or national regulation, spectrum constraints, security, scalability, co-existence, and interference are interesting directions for further studies to continue enhancing network performance.

The methodology of signal propagation simulation proposed in this thesis is not tied to CSS by Semtech LoRa and applies well to other long-range communication technologies such as Ultra Narrow Band (UNB) from Sigfox [187], and Weightless from Ingenu [188].

A hybrid network that uses both short-range communications such as Zigbee, Bluetooth, so on in mesh topology to gather sensing data from sensor nodes to their base stations and long-range communications for base station connections. Another research direction could be related to utilizing massive parallel execution with GPUs to predict coverage of mobile gateway to collect data from WSNs in real-time.

These simulation tools help considerably in understanding geographic deployment properties, time constraints, and even communication energy budgets. The footprint of an 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 the 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 a number of hops, if any.

Bibliography

[1] P. T. Tuyen, H. Xuan Hiep, and P. Bernard, "Cellular Simulation for Distributed Sensing over Complex Terrains," no. June, pp. 1–28, 2018. 11, 25, 45, 67, 216

[2] ——, "Parallel Cellular Automata Based Simulation of Radio Signal Propagation," vol. 14, no. 10, pp. 467–472, 2016. 11, 216

[3] O. Nazra, P. Goubier, H. X. Huynh, T. P. Truong, and M. Traore, "Wireless Sensor Network-based Monitoring , Cellular Modelling and Simulations for the Environment," vol. 2017, pp. 56–63, 2017. 11, 216

[4] P.-Y. Lucas, N. H. Van Long, T. P. Truong, and B. Pottier, *Wireless Sensor Networks and Satellite Simulation.* Cham: Springer International Publishing, 2015, pp. 185–198. [Online]. Available: https://doi.org/10.1007/978-3-319-25479-1_14 11, 19, 23, 37, 46, 123, 164, 215

[5] T. P. Truong, H. V. Tran, and H. X. Huynh, "Optimizing the Connection Time for LEO Satellite Based on Dynamic Sensor Field," pp. 1–15. 11, 215

[6] P. T. Tuyen, H. H. Luong, H. H. Huynh, H. Xuan Hiep, and P. Bernard, "Modeling and Optimizing of Connections for Dynamic Sensor Fields Based on BT-Graph," vol. 1, pp. 297–310, 2016. 11, 215

[7] T. Phong Truong, H. Hoang Luong, H. Hiep Xuan, V. Cong Phan, and B. Pottier, "Modeling the Connections of Dynamic Sensor Fields Based on BT-Graph," *EAI Endorsed Transactions on Context-aware Systems and Applications*, vol. 16, no. 8, Mar. 2016. [Online]. Available: http://hal.univ-brest.fr/hal-01294080 11, 216

[8] H. Dutta, T. Failler, N. Melot, B. Pottier, and S. Stinckwich, "An execution flow for dynamic concurrent systems: simulation of WSN on a Smalltalk/CUDA environment," in *Proceedings of SIMPAR 2010 Workshops Intl. Conf. on simulation, modeling and programming for autonomous robots*, ser. Dynamic languages for robotic and sensors systems (DYROS), Darmstadt, Germany, 15-16 novembre 2010, pp. 290 – 295. 14, 40, 171

[9] I. S. P. IEEE and Governance, "1516-2010 - ieee standard for modeling and simulation (m&s) high level architecture (hla)– framework and rules," https://standards.ieee.org/findstds/standard/1516-2010.html, 2010. 14

[10] H. V. Tran, T. P. Truong, K. T. Nguyen, H. X. Huynh, and B. Pottier, "A federated approach for simulations in cyber-physical systems," in *Context-Aware Systems and Applications - 4th International Conference, ICCASA 2015, Vung Tau, Vietnam, November 26-27, 2015, Revised Selected Papers*, 2015, pp. 165–176. [Online]. Available: https://doi.org/10.1007/978-3-319-29236-6_17 14, 26, 39

[11] I. Ticlo, "Is Tech the New Currency? Why You Need Modern IT," https://www.insight.com/en_US/learn/content/2017/05252017-is-tech-the-new-currency-why-you-need-modern-it.html, 2017, [Online; accessed on 15 June 2018]. 14, 15

[12] J. Howell, "Number of Connected IoT Devices Will Surge to 125 Billion by 2030, IHS Markit Says," https://technology.ihs.com/596542/number-of-connected-iot-devices-will-surge-to-125-billion-by-2030-ihs-markit-says, 2017, [Online; accessed on 15 June 2018]. 14

[13] U. Raza, P. Kulkarni, and M. Sooriyabandara, "Low Power Wide Area Networks: An Overview," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 2, pp. 855–873, 2017. 15, 16

[14] D. Patel and M. Won, "Experimental Study on Low Power Wide Area Networks (LPWAN) for Mobile Internet of Things," 2017. [Online]. Available: http://arxiv.org/abs/1705.06926{%}0Ahttp://dx.doi.org/10.1109/VTCSpring.2017.8108501 16

[15] D. Ismail, M. Rahman, and A. Saifullah, "Low-Power Wide-Area Networks: Opportunities, Challenges, and Directions," *Distributed Computing and Networks*, 2018. [Online]. Available: https://doi.org/10.1145/3170521.3170529 15, 16

[16] A. Saifullah, M. Rahman, D. Ismail, C. Lu, R. Chandra, and J. Liu, "SNOW - Sensor Network over White Spaces," *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM - SenSys '16*, pp. 272–285, 2016. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2994551.2994552 16

[17] A. Saifullah, C. Lu, S. Louis, and R. Chandra, "Enabling Reliable , Asynchronous , and Bidirectional Communication in Sensor Networks over White Spaces," *In Proceedings of SenSys '17*, 2017. [Online]. Available: http://www.cse.wustl.edu/{~}lu/papers/sensys17.pdf 16

[18] L. M. Oliveira and J. J. Rodrigues, "Wireless Sensor Networks: a Survey on Environmental Monitoring," *Journal of Communications*, vol. 6, no. 2, pp. 143–151, 2011. [Online]. Available: http://www.jocm.us/index.php?m=content{&}c=index{&}a=show{&}catid=54{&}id=202 16, 18

[19] K. Mekki, E. Bajic, F. Chaxel, and F. Meyer, "A comparative study of LPWAN technologies for large-scale IoT deployment," *ICT Express*, 2018. [Online]. Available: https://doi.org/10.1016/j.icte.2017.12.005 16

[20] T. P. Truong, H. X. Huynh, and B. Pottier, "Wireless sensor network: Long-range radio coverage for complex terrain areas," in *GDR Ondes*. Journées Thématiques "Capteurs Magnétiques & Électromagnétiques (EM) et Applications, Jun. 2018. 17

[21] T. P. Truong, Udrekh, B. Pottier, and S. Group, "Long-range communications: A promising technology for environmental monitoring applications in developing countries," in *JWG 2018*. The 2018 Joint Working Group France-Indonesia Cooperation in Higher Education, Research, Innovation and Entrepreneurship (JWG2018), Jun. 2018. 17

[22] T. Rappaport, *Wireless Communications: Principles and Practice*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001. 17, 68, 72, 74, 75, 78, 183

[23] S. R. Saunders and S. R. Simon, *Antennas and Propagation for Wireless Communication Systems*, 1st ed. New York, NY, USA: John Wiley & Sons, Inc., 1999. 17

[24] T. P. Truong, H. X. Huynh, and B. Pottier, "Monitoring of Environment : A high performance method for radio coverage exploration," in *IEEE Radio 2016, Hotel Le Récif, Saint-Gilles Les Bains Réunion Island*, 2016, pp. 3–4. 17

[25] W. P. Commun, "Survey on Coverage Problems in Wireless Sensor Networks," 2014. 17

[26] G. Fan, S. Jin, and D. Processing, "Coverage Problem in Wireless Sensor Network : A Survey," *Journal of Networks*, vol. 5, no. 9, pp. 1033–1040, 2010. 17

[27] Y. Wang, S. Wu, Z. Chen, X. Gao, and G. Chen, "Coverage problem with uncertain properties in wireless sensor networks: A survey," *Computer Networks*, vol. 123, pp. 200–232, 2017. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2017.05.008 17

[28] N. Chaudhary and S. Gupta, "A Survey on Coverage Problem in Wireless Sensor Network," vol. 4, no. 5, pp. 11 952–11 955, 2015. 17

[29] I. Khoufi, P. Minet, A. Laouiti, and S. Mahfoudh, "Survey of Deployment Algorithms in Wireless Sensor Networks: Coverage and Connectivity Issues and Challenges," *International Journal of Autonomous and Adaptive Communications Systems (IJAACS)*, vol. 10, no. 4, pp. 341–390, 2017. [Online]. Available: https://hal.inria.fr/hal-01095749 17

[30] A. Tripathi, H. P. Gupta, T. Dutta, R. Mishra, K. K. Shukla, and S. Jit, "Coverage and Connectivity in WSNs: A Survey, Research Issues and Challenges," *IEEE Access*, vol. 6, pp. 26 971–26 992, 2018. 17

[31] V. Marija, T. Dimitar, and F. Sonja, "Durkin's Propagation Model Based on Triangular Irregular Network Terrain," vol. 83, no. January, pp. 10–13, 2011. [Online]. Available: http://www.springerlink.com/index/10.1007/978-3-642-19325-5 17

[32] R. Seixas, M. Mediano, and M. Gattass, "Efficient line-of-sight algorithms for real terrain data," *... e IV Simpósio de Logística da ...*, no. June 1999, 1999. [Online]. Available: http://w3.impa.br/{~}rbs/pdf/spolm99.pdf 17

[33] S. Christoph, "Algorithms and software for radio signal coverage prediction in terrains," Ph.D. dissertation, ETH Zurich, Switzerland). [Online]. Available: https://doi.org/10.3929/ethz-a-004222924 17, 68

[34] A. D. Kora, B. A. Elono Ongbwa, J. P. Cances, and V. Meghdadi, "Accurate radio coverage assessment methods investigation for 3G/4G networks," *Computer Networks*, vol. 107, pp. 246–257, 2016. 17

[35] S. I. Popoola, A. A. Atayero, and N. Faruk, "Received signal strength and local terrain profile data for radio network planning and optimization at GSM frequency bands," *Data in Brief*, vol. 16, pp. 972–981, 2018. [Online]. Available: http://dx.doi.org/10.1016/j.dib.2017.12.036 17

[36] M. S. Aliyu, A. H. Abdullah, H. Chizari, T. Sabbah, and A. Altameem, "Coverage enhancement algorithms for distributed mobile sensors deployment in wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 2016, 2016. 18

[37] L. Feng, T. Qiu, Z. Sun, F. Xia, and Y. Zhou, "A Coverage Strategy for Wireless Sensor Networks in a Three-dimensional Environment." 18

[38] S. Filiposka and D. Trajanov, "Terrain-aware three-dimensional radio-propagation model extension for NS-2," *Simulation*, vol. 87, no. 1-2, pp. 7–23, 2011. 18

[39] K. Veenstra and K. Obraczka, "Guiding Sensor-Node Deployment Over 2 . 5D Terrain," pp. 8347–8353, 2015. 18

[40] N. Ganguly, B. K. Sikdar, A. Deutsch, G. Canright, and P. P. Chaudhuri, "A Survey on Cellular Automata," *Engineering*, pp. 1–30, 2003. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.107.7729 18

[41] D. Das, "A Survey on Cellular Automata and Its Applications," vol. 269, no. September, 2012. [Online]. Available: http://link.springer.com/10.1007/978-3-642-29219-4 18

[42] C. Banerjee and S. Saxena, "Energy conservation in wireless sensor network using block cellular automata," in *2013 International Conference on Computer Communication and Informatics*, Jan 2013, pp. 1–6. 18

[43] I. Banerjee, P. Chanak, and H. Rahaman, "CCABC: cyclic cellular automata based clustering for energy conservation in sensor networks," *CoRR*, vol. abs/1109.2430, 2011. [Online]. Available: http://arxiv.org/abs/1109.2430 18

[44] X. Xu, X. Zhang, and L. Wang, "Simulating energy efficient wireless sensor networks using cellular automata," *Proceedings of the 2011 Winter Simulation Conference (WSC)*, pp. 3202–3211, 2011. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6148018 18

[45] M. Doman, T. Dahlberg, and J. Payton, "Simulation environment scenarios using cellular automata for wireless sensor network analysis," in *Spring Simulation Multiconference 2009 - Co-located with the 2009 SISO Spring Simulation Interoperability Workshop*. Association for Computing Machinery, Inc, 3 2009. 18

[46] W. Li, L. Ma, and Q. Yu, "Cellular automata-based multi-hop wsn routing protocol energy saving technology," in *2013 International Conference on Communications, Circuits and Systems (ICCCAS)*, vol. 1, Nov 2013, pp. 113–117. 18

[47] D. Systems, "Modelling Malware Response in Wireless Networks Using Stochastic Cellular Automata Sensor," vol. VI, no. 4, 2014. 18

[48] S. Roy, J. Karjee, U. S. Rawat, N. Dayama Pratik, and N. Dey, "Symmetric Key Encryption Technique: A Cellular Automata based Approach in Wireless Sensor Networks," *Physics Procedia*, vol. 78, pp. 408–414, 2016. 18

[49] A. Singh and T. P. Sharma, "A survey on area coverage in wireless sensor networks," *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, no. July, pp. 829–836, 2014. [Online]. Available: http://www.scopus.com/inward/record.url?eid=2-s2.0-84921665798{&}partnerID=tZOtx3y1 18

[50] A. Sangwan and R. P. Singh, "Survey on coverage problems in wireless sensor networks," *Wireless Personal Communications*, vol. 80, no. 4, pp. 1475–1500, Feb 2015. [Online]. Available: https://doi.org/10.1007/s11277-014-2094-3 18

[51] A. Tretyakova, F. Seredynski, and P. Bouvry, "Graph cellular automata approach to the maximum lifetime coverage problem in wireless sensor networks," *Simulation*, vol. 92, no. 2, pp. 153–164, 2016. [Online]. Available: https://doi.org/10.1177/0037549715612579 18

[52] S. Wolfram, "Cellular automata: a model of complexity," *Nature 31*, pp. 419–424, 1984. 18

[53] P. S. A.G. Hoekstra, J. Froc, "Simulating Complex Systems by Cellular Automata," no. June, pp. 7–8, 2010. [Online]. Available: http://link.springer.com/10.1007/978-3-642-12203-3 18

[54] W. Li, A. Y. Zomaya, and A. Al-Jumaily, "Cellular automata based models of wireless sensor networks," in *Proceedings of the Seventh ACM International Workshop on Mobility Management & Wireless Access, MOBIWAC 2009, Tenerife,*

Canary Islands, Spain, October 26-27, 2009, 2009, pp. 1–6. [Online]. Available: http://doi.acm.org/10.1145/1641776.1641777 18

[55] N. S. Semtech, M. L. Semtech, T. E. Ibm, and T. K. Ibm, "LoRa MAC Specification v1.0," pp. 1–21, 2015. 18, 69, 118, 175

[56] R. O. Cunha, A. P. Silva, A. a. F. Loureiro, and L. B. Ruiz, "Simulating large wireless sensor networks using cellular automata," *Proceedings of the 38th annual Symposium on Simulation*, pp. 323–330, 2005. 18

[57] B. Farjad, A. Gupta, S. Razavi, M. Faramarzi, and D. J. Marceau, "An integrated modelling system to predict hydrological processes under climate and land-use/cover change scenarios," *Water (Switzerland)*, vol. 9, no. 10, 2017. 18

[58] Z. Lin, "Regional-Scale Assessment and Simulation of Land Salinization Using Cellular Automata-Markov Model," vol. 0300, no. xxxx, 2012. 18

[59] H. Kassogué, A. S. Bernoussi, M. Amharref, and M. Ouardouz, "Cellular automata approach for modelling climate change impact on water resources," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 5760, no. June, pp. 1–16, 2017. [Online]. Available: https://doi.org/10.1080/17445760.2017.1331438 18

[60] M. Di Traglia, F. Attorre, F. Francesconi, R. Valenti, and M. Vitale, "Is cellular automata algorithm able to predict the future dynamical shifts of tree species in Italy under climate change scenarios? A methodological approach," *Ecological Modelling*, vol. 222, no. 4, pp. 925–934, 2011. 18

[61] E. J. Yoon, D. K. Lee, H. G. Kim, H. R. Kim, E. Jung, and H. Yoon, "Multi-objective land-use allocation considering landslide risk under climate change: Case study in pyeongchang-gun, Korea," *Sustainability (Switzerland)*, vol. 9, no. 12, 2017. 18

[62] T. P. VAN, P. A. CARLING, T. J. COULTHARD, and P. M. ATKINSON, "Cellular Automata Approach for Flood Forecasting in a Bifurcation River System," *Publs. Inst. Geophys. Pol. Acad. Sc.*, vol. 401, no. November 2014, pp. 255–262, 2007. 18

[63] J. Cirbus and M. Podhoranyi, "Cellular automata for the flow simulations on the earth surface, optimization computation process," *Applied Mathematics and Information Sciences*, vol. 7, no. 6, pp. 2149–2158, 2013. 18

[64] P. Topa, "A Cellular Automata Approach for Modeling Complex River Systems," vol. 4173, no. May 2014, 2006. [Online]. Available: http://link.springer.com/10.1007/11861201 18

[65] A. KHATIBI, S. POUREBRAHIM, and A. DANEHKAR, "a Cellular Automata Model for Monitoring and Simulating Urban Land Use/Cover Changes Toward Sustainability," *Journal of Environmental Engineering and Landscape Management*, vol. 26, no. 1, pp. 1–7, 2018. [Online]. Available: http://journals.vgtu.lt/index.php/JEELM/article/view/286 18

[66] K. A. Hawick, "Modelling Flood Incursion and Coastal Erosion Using Cellular Automata Simulations," *IASTED International Conference on Environmental Management and Engineering*, pp. 1–8, 2014. [Online]. Available: http://www.hull.ac.uk/php/466990/csi/reports/0007/csi-0007.pdf 18

[67] M. Gług and J. Wąs, "Modeling of oil spill spreading disasters using combination of Langrangian discrete particle algorithm with Cellular Automata approach," *Ocean Engineering*, vol. 156, no. March, pp. 396–405, 2018. 19

[68] G. Guariso and V. Maniezzo, "air quality simulation through cellular automata_Guariso.pdf," vol. 7, no. September 1992, pp. 131–141, 1993. 19

[69] M. Avolio, A. Errera, V. Lupiano, P. Mazzanti, and S. Di Gregorio, "VALANCA: A cellular automata model for simulating snow avalanches," *Journal of Cellular Automata*, vol. 12, no. 5, pp. 309–332, 2017. [Online]. Available: http://www.nhazca.it/pdf/VALANCA{_}2017.pdf 19

[70] G. Machado, V. Lupiano, M. V. Avolio, F. Gullace, and S. Di Gregorio, "A cellular model for secondary lahars and simulation of cases in the VascÃºn Valley, Ecuador," *Journal of Computational Science*, vol. 11, pp. 289–299, 2015. 19

[71] E. S. Mohamed, "Tsunami Wave Simulation Models Based on Hexagonal Cellular Automata," vol. 8, no. 3, pp. 91–101, 2013. 19

[72] A. Vicari, H. Alexis, C. Del Negro, M. Coltelli, M. Marsella, and C. Proietti, "Modeling of the 2001 lava flow at Etna volcano by a Cellular Automata approach," *Environmental Modelling and Software*, vol. 22, no. 10, pp. 1465–1471, 2007. 19

[73] K. C. Clarke, J. a. Brass, and P. J. Riggan, "A Cellular-Automaton Model of Wildfire Propagation and Extinction," *Photogrammetric Engineering and Remote Sensing*, vol. 60, no. 11, pp. 1355–1367, 1994. 19

[74] J. Lahti, "Modelling Urban Growth Using Cellular Automata: A case study of Sydney, Australia," *Transport*, 2008. [Online]. Available: http://www.itc.nl/library/papers{_}2008/msc/gem/lahti.pdf 19

[75] I. Santé, A. M. García, D. Miranda, and R. Crecente, "Cellular automata models for the simulation of real-world urban processes: A review and analysis," *Landscape and Urban Planning*, vol. 96, no. 2, pp. 108–122, 2010. 19

[76] J. Du, Q. Liang, and Y. Xia, "Parallel simulation based on gpu-acceleration," in *AsiaSim 2012*, T. Xiao, L. Zhang, and M. Fei, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 355–362. 19

[77] K. Gulati and S. P. Khatri, *GPU Architecture and the CUDA Programming Model*. Boston, MA: Springer US, 2010, pp. 23–30. [Online]. Available: https://doi.org/10.1007/978-1-4419-0944-2_3 19

[78] Z. Qu, G. Zhang, and J. Xie, "LEO Satellite Constellation for Internet of Things," *IEEE Access*, pp. 1–1, 2017. [Online]. Available: http://ieeexplore.ieee.org/document/8002583/ 19

[79] T. P. Truong, H. V. Tran, H. X. Huynh, and B. Pottier, "Optimizing the connection time for LEO satellite based on dynamic sensor field," in *Context-Aware Systems and Applications - 4th International Conference, ICCASA 2015, Vung Tau, Vietnam, November 26-27, 2015, Revised Selected Papers*, 2015, pp. 380–394. [Online]. Available: https://doi.org/10.1007/978-3-319-29236-6_36 19, 24, 123, 124, 125, 126, 129

[80] W. Colitti, K. Steenhaut, N. Descouvemont, and A. Dunkels, "Satellite based wireless sensor networks: Global scale sensing with nano- and pico-satellites," in *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, ser. SenSys '08. New York, NY, USA: ACM, 2008, pp. 445–446. [Online]. Available: http://doi.acm.org/10.1145/1460412.1460495 19

[81] N. Celandroni, Ferro, and Al., "A survey of architectures and scenarios in satellite-based wireless sensor networks: system design aspects," *International Journal of Satellite Communications and Networking*, vol. 31, no. 1, pp. 1–38, 2013. [Online]. Available: http://dx.doi.org/10.1002/sat.1019 19, 123

[82] L. Akoglu, H. Tong, and D. Koutra, "Graph based anomaly detection and description: A survey," *Data Min. Knowl. Discov.*, vol. 29, no. 3, pp. 626–688, May 2015. [Online]. Available: http://dx.doi.org/10.1007/s10618-014-0365-y 19, 124

[83] I. Khoufi, P. Minet, A. Laouiti, and S. Mahfoudh, "Survey of Deployment Algorithms in Wireless Sensor Networks: Coverage and Connectivity Issues and Challenges," *International Journal of Autonomous and Adaptive Communications Systems (IJAACS)*, vol. 10, no. 4, pp. 341–390, 2017. [Online]. Available: https://hal.inria.fr/hal-01095749 19

[84] J. Dong, Q. S. Chen, and Z. Niu, "Random graph theory based connectivity analysis in wireless sensor networks with rayleigh fading channels," *2007 Asia-Pacific Conference on Communications*, pp. 123–126, 2007. 19, 124

[85] S. A. Çamtepe, B. Yener, and M. Yung, "Expander graph based key distribution mechanisms in wireless sensor networks," *2006 IEEE International Conference on Communications*, vol. 5, pp. 2262–2267, 2006. 19

[86] C. Chaparro and W. Eberle, "Detecting anomalies in mobile telecommunication networks using a graph based approach," in *Proceedings of the Twenty-Eighth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2015, Hollywood, Florida. May 18-20, 2015.*, 2015, pp. 410–415. [Online]. Available: http://www.aaai.org/ocs/index.php/FLAIRS/FLAIRS15/paper/view/10377 19, 124

[87] D. Panwar and S. G. Neogi, "Design of Energy Efficient Routing Algorithm for W Ireless S Ensor N Etwork (Wsn) U Sing P Ascal G Raph," pp. 175–189, 2013. 19

[88] T. Yang, C. Kang, and G. Nan, "An energy-efficient and fault-tolerant convergecast protocol in wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 8, no. 9, p. 429719, 2012. [Online]. Available: https://doi.org/10.1155/2012/429719 19

[89] G. Singla, A. Tiwari, and D. P. Singh, "New approach for graph algorithms on gpu using cuda," 2013. 19

[90] N. Bhatia and C. Author, "Survey of Nearest Neighbor Techniques," *IJCSIS) International Journal of Computer Science and Information Security*, vol. 8, no. 2, pp. 302–305, 2010. 19

[91] P. Harish and P. J. Narayanan, "Accelerating large graph algorithms on the gpu using cuda," in *Proceedings of the 14th International Conference on High Performance Computing*, ser. HiPC'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 197–208. [Online]. Available: http://dl.acm.org/citation.cfm?id=1782174.1782200 20

[92] P. J. Martín, R. Torres, and A. Gavilanes, "CUDA solutions for the sssp problem," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5544 LNCS, no. PART 1, pp. 904–913, 2009. 20

[93] Q. Kuang and L. Zhao, "A practical gpu based knn algorithm," 2009. 20

[94] V. B. Nikam and B. B. Meshram, "Parallel knn on gpu architecture using opencl," 2014. 20

[95] V. Garcia, E. Debreuve, F. Nielsen, and M. Barlaud, "K-nearest neighbor search: Fast gpu-based implementations and application to high-dimensional feature matching," *2010 IEEE International Conference on Image Processing*, pp. 3757–3760, 2010. 20

[96] *Nvidia Cuda Programming Guide 2.0*, NVIDIA, 2008. 20

[97] *Occam 2.1 Reference Manual*, SGS-THOMSON Microelectronics Limited, 1995. 20, 50, 52, 57

[98] University of Kent, "Welcome to kroc," http://projects.cs.kent.ac.uk/projects/kroc/trac. 20, 22, 94

[99] D. May, "Communicating processes and occam," no. February, 1987. 20

[100] C. L. Jacobsen and M. C. Jadud, "The Transterpreter: A Transputer Interpreter," pp. 99–106, 2004. [Online]. Available: http://kar.kent.ac.uk/14104/ 20

[101] P. Welch and F. Barnes, "Communicating Mobile Processes: introducing occam-pi," pp. 175–210, 2005. [Online]. Available: http://kar.kent.ac.uk/14334/ 20, 114

[102] T. P. Truong, H. X. Huynh, and B. Pottier, "Wireless Sensor Network: Long-range Radio Coverage for Complex Terrain Areas," Tech. Rep., 2018. 21, 216

[103] ——, "Sensing and Simulation: Cellular Methods and Tools," Tech. Rep., 2016. 22, 216

[104] F. R. M. Barnes, "Blocking System Calls in KRoC/Linux," in *Communicating Process Architectures 2000*, P. H. Welch and A. W. P. Bakkers, Eds., sep 2000, pp. 155–178. 22, 40

[105] J. R. Wertz and W. Larson, *Space Mission Analysis and Design*, 1999. 23, 125, 126, 127, 128

[106] E. Callaway, P. Gorday, L. Hester, J. A. Gutierrez, M. Naeve, B. Heile, and V. Bahl, "Home networking with IEEE 802.15.4: A developing standard for low-rate wireless personal area networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 70–77, 2002. 25

[107] Semtech, "Sx1276/77/78/79 -137 MHz to 1020 MHz Low Power Long Range Transceiver. Rev. 5 - August 2016." Tech. Rep. August, 2016. [Online]. Available: https://www.semtech.com/uploads/documents/DS_SX1276-7-8-9_W_APP_V5.pdf 25, 68, 69, 71, 88, 89, 175

[108] Sigfox, "Sigfox Technology Overview," https://www.sigfox.com/en/sigfox-iot-technology-overview, 2017, [Online; accessed on 15 December 2017]. 25

[109] Department of Defense - Defense Modeling and Simulation Office, "RTI 1.3-Next Generation Programmer's Guide Version 3.2," p. 148, 2000. 26, 164, 165

[110] M. F. Hutchinson and J. C. Gallant, "Representation of terrain," *Geographical information systems*, vol. 1, no. 1991, pp. 105–124, 1999. 26, 29, 86

[111] G. Strang and K. Borre, *Linear Algebra, Geodesy, and GPS*. Wellesley-Cambridge Press, 1997. 27

[112] J. Zhang, P. M. Atkinson, and M. F. Goodchild, *Scale in Spatial Information and Analysis*. CRC Press, 2014. 27, 85

[113] S. D. Riley, S.J. and R. Elliot, "A terrain ruggedness index that quantifies topographic heterogeneity," *Internation Journal of Science*, vol. Vol. 5, pp. 23–27, 1999. 27, 31

[114] J. Jenness, "Topographic Position Index (TPI) v. 1.2," p. 42, 2006. 28

[115] Openstreetmap, "Slippy map tilenames," https://wiki.openstreetmap.org/wiki/Slippy_map_tilenames, 2017, [Online; accessed on 28 July 2017]. 32, 190

[116] A. Huuskonen, E. Saltikoff, and I. Holleman, "The operational weather radar network in europe," *Bulletin of the American Meteorological Society*, vol. 95, no. 6, June 2014. 34

[117] EUMETSAT, " Monitoring The Atmosphere, Ocean and Climate from Space - Transforming Our World," https://www.eumetsat.int/website/home/AboutUs/Publications/Brochures/index.html, October 2017. 34

[118] worldweatheronline, "Bejaia Historical Weather," https://www.worldweatheronline.com/bejaia-weather-history/bejaia/dz.aspx, 2017, [Online; accessed on 25 July 2017]. 35

[119] Proj4, "Proj4 documentation," https://proj4.ºrg, 2018, [Online; accessed on 4 June 2018]. 35

[120] Wikipedia.org, "Bejaia Historical Weather," https://en.wikipedia.org/wiki/Web_Mercator, 2018, [Online; accessed on 25 April 2018]. 36

[121] P.-Y. Lucas, "Modélisations, simulations, synthèses pour des réseaux dynamiques de capteurs sans fil," Ph.D. dissertation, Universite de Brest (UBO), LabSTICC, UMR CNRS 6285, 2016. [Online]. Available: http://wsn.univ-brest.fr/svn/PYLast/these-rassembler.pdf 36, 37, 167, 183, 190

[122] K. P. Pridal, " Google Summer of Code 2008, project GDAL2Tiles for OSGEO," http://www.maptiler.org/google-maps-coordinates-tile-bounds-projection/, 2011, [Online; accessed on 25 April 2018]. 37

[123] J. V. Neumann, *Theory of Self-Reproducing Automata*, A. W. Burks, Ed. Champaign, IL, USA: University of Illinois Press, 1966. 38

[124] U. Frisch, D. D'Humieres, B. Hasslacher, P. Lallemand, Y. Pomeau, and J.-P. Rivet, "Lattice Gas Hydrodynamics in Two and Three Dimensions," *Complex Systems Publications*, vol. 1, no. 4, pp. 649–707, 1987. 38

[125] N. Margolus and T. Toffoli, *Cellular Automata Machines: A New Environment for Modeling.* MIT Press, 1987. 38

[126] H. Hatzikirou and A. Deutsch, *Lattice-Gas Cellular Automaton Modeling of Emergent Behavior in Interacting Cell Populations.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 301–331. [Online]. Available: https://doi.org/10.1007/978-3-642-12203-3_13 39

[127] M. Traore, "Modélisation cellulaire et simulation physique: contribution à l'analyse de la dynamique de population des insectes ravageurs," Ph.D. dissertation, Université de Bretagne occidentale - Brest, 2018. [Online]. Available: http://wsn.univ-brest.fr/svn/projetCari/theseMT/AllMTV3.pdf 39, 55

[128] C. A. R. Hoare, "Communicating sequential processes," *Communications of the ACM*, vol. 21, pp. 666–677, 1978. 39

[129] P. Quesada-Barriuso, D. B. Heras, and F. Argüello, "Efficient GPU asynchronous implementation of a watershed algorithm based on cellular automata," *Proceedings of the 2012 10th IEEE International Symposium on Parallel and Distributed Processing with Applications, ISPA 2012*, pp. 79–86, 2012. 39

[130] N. Lynch, *Distributed algorithms*. Morgan Kaufman, 1996. 39, 51, 72, 91

[131] D. Pountain and D. May, *A tutorial introduction to Occam programming*. New York, NY, USA: McGraw-Hill, Inc., 1987. 40

[132] A. Iqbal and B. Pottier, "Meta-Simulation of Large WSN on Multi-core Computers," in *Proceedings of the 2010 Spring Simulation Multiconference*, ser. SpringSim SCS Conference. Orlando, USA: Society for Computer Simulation International, April 2010, p. 133. 40

[133] J. Kari, "Theory of cellular automata: A survey," vol. 334, pp. 3–33, 2005. 43

[134] "Cellular Automata," http://mathworld.wolfram.com/topics/, 2018, [Online; accessed on 20 March 2017]. 43

[135] B. Pottier and P.-Y. Lucas, "Dynamic networks : Netgen tools," Jun. 2014, guide for simulation software, installation and use. [Online]. Available: http://hal.univ-brest.fr/hal-01294288 43, 50, 117

[136] NVidia, "NVIDIA Developer," https://developer.nvidia.com/, 2018, [Online; accessed on 20 March 2018]. 50, 124

[137] K. Bouazza, J. Champeau, B. Pottier, and P. Ng, "Implementing cellular automata on the ArMen machine," in *Algorithms and parallel VLSI architectures II*. Elsevier, 1992. 51

[138] E. B. Keita, "Physical models and perception, contributions to sound analysis in urban environment," Theses, Université de Bretagne occidentale - Brest, Jul. 2015. [Online]. Available: https://tel.archives-ouvertes.fr/tel-01596611 55

[139] Météo-France, " Le modéle á maille fine Arome," http://www.meteofrance.fr/prevoir-le-temps/la-prevision-du-temps/le-modele-a-maille-fine-arome, June 2018. 55

[140] F. S., K. T. A. T., R. J.-L., M. V., M. E., and L. M. P., "Ecoclimap-ii/europe: a twofold database of ecosystems and surface parameters at 1 km resolution based on satellite information for use in land surface, meteorological and climate models,," *Geosci. Model Dev., 6*, pp. 563–582, 2013. 55

[141] ITU-R, "Propagation by diffraction, Recommendation P.526-13 (11/2013)," http://www.itu.int/dms_pubrec/itu-r/rec, Tech. Rep., 2013, [Online; accessed on 20 March 2018]. 72

[142] J. S. Seybold, *Introduction to RF Propagation.* John Wiley & Sons Inc., 2005. 68, 74, 78

[143] S. Haykin and M. Moher, *Modern Wireless Communication.* Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2004. 68

[144] M. Aref and A. Sikora, "Free space range measurements with Semtech LoRa technology," *2014 2nd International Symposium on Wireless Systems within the Conferences on Intelligent Data Acquisition and Advanced Computing Systems, IDAACS-SWS 2014*, no. September, pp. 19–23, 2014. 68

[145] C. Phillips, D. Sicker, and D. Grunwald, "A survey of wireless path loss prediction and coverage mapping methods," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 1, pp. 255–270, 2013. 68

[146] A. Hrovat, I. Ozimek, A. Vilhar, T. Celcer, I. Saje, T. Javornik, and . Mobitel, "An Open-Source Radio Coverage Prediction Tool," *LATEST TRENDS on COMMUNICATIONS*, pp. 135–140, 2010. 68

[147] A. Valcarce, G. De La Roche, and J. Zhang, "A GPU approach to FDTD for radio coverage prediction," *2008 11th IEEE Singapore International Conference on Communication Systems, ICCS 2008*, pp. 1585–1590, 2008. 68

[148] Z. Yun and M. F. Iskander, "Ray tracing for radio propagation modeling: Principles and applications," *IEEE Access*, vol. 3, pp. 1089–1100, 2015. 68

[149] C. Goursad and J. M. Gorce, "Dedicated networks for IoT: PHY/MAC state of the art and challenges," *EAI Endorsed Transactions on the Internet of Things*, vol. 1, no. 1, pp. 1–12, 2015. 68

[150] K. Mikhaylov, J. Petäjäjärvi, and T. Hänninen, "Analysis of Capacity and Scalability of the LoRa Low Power Wide Area Network Technology," *European Wireless 2016*, pp. 119–124, 2016. 68

[151] J. Toussaint, N. El Rachkidy, and A. Guitton, "Performance analysis of the on-the-air activation in LoRaWAN," *7th IEEE Annual Information Technology, Electronics and Mobile Communication Conference, IEEE IEMCON 2016*, no. 1, 2016. 68

[152] B. Kim and K. I. Hwang, "Cooperative downlink listening for low-power long-range wide-area network," *Sustainability (Switzerland)*, vol. 9, no. 4, 2017. 68

[153] A. Augustin, J. Yi, T. Clausen, and W. Townsley, "A Study of LoRa: Long Range & Low Power Networks for the Internet of Things," *Sensors*, vol. 16, no. 9, p. 1466, 2016. [Online]. Available: http://www.mdpi.com/1424-8220/16/9/1466 68

[154] M. Cattani, C. Boano, and K. Römer, "An Experimental Evaluation of the Reliability of LoRa Long-Range Low-Power Wireless Communication," *Journal of Sensor and*

Actuator, 2017. [Online]. Available: http://www.mdpi.com/2224-2708/6/2/7/htm 68

[155] J. Petäjäjärvi, K. Mikhaylov, M. Pettissalo, J. Janhunen, and J. Iinatti, "Performance of a low-power wide-area network based on LoRa technology: Doppler robustness, scalability, and coverage," *International Journal of Distributed Sensor Networks*, vol. 13, no. 3, p. 155014771769941, 2017. [Online]. Available: http://journals.sagepub.com/doi/10.1177/1550147717699412 68

[156] M. Bor and U. Roedig, "LoRa Transmission Parameter Selection," 2017. 68

[157] M. C. Bor, U. Roedig, T. Voigt, and J. M. Alonso, "Do LoRa Low-Power Wide-Area Networks Scale?" *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems - MSWiM '16*, no. November, pp. 59–67, 2016. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2988287.2989163 68

[158] D. Bankov, E. Khorov, and A. Lyakhov, "On the limits of LoRaWAN channel access," *Proceedings - 2016 International Conference on Engineering and Telecommunication, EnT 2016*, pp. 10–14, 2017. 68

[159] Semtech, "LoRa Modulation Basics," Tech. Rep. May, 2015. [Online]. Available: http://www.semtech.com/images/datasheet/an1200.22.pdf 68, 89, 175, 203

[160] ——, "LoRa Modem Design Guide," Tech. Rep. July, 2013. [Online]. Available: http://www.semtech.com/images/datasheet/LoraDesignGuideSTD.pdf 68, 175

[161] J. E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Syst. J.*, vol. 4, no. 1, pp. 25–30, Mar. 1965. [Online]. Available: http://dx.doi.org/10.1147/sj.41.0025 73, 100

[162] T. P. Truong and B. Pottier, "Monitoring of environment: A high performance method for radio coverage exploration," in *IEEE Radio, Indian Ocean.* IEEE, La Reunion, Oct. 2016. 216

[163] Manuka, "Balloon project," http://www.instructables.com/id/Introducing-LoRa-/, 2015, [Online; accessed on 25 July 2017]. 85, 128

[164] "Shuttle Radar Topography Mission (SRTM) 1 Arc-Second Global," https://lta.cr.usgs.gov/SRTM1Arc, 2015, [Online; accessed on 25 July 2017]. 86

[165] NVIDIA, "Cuda toolkit documentation, profiler user's guide," NVIDIA, Tech. Rep., 2017. [Online]. Available: http://docs.nvidia.com/cuda/profiler-users-guide 109

[166] G. D. Micheli, "Synthesis and optimization of digital circuits." 112

[167] C. a. R. a. R. Hoare, "Book_ Communicating sequential processes," *Communications of the ACM*, vol. 21, no. 8, pp. 666–677, 1978. [Online]. Available: http://portal.acm.org/citation.cfm?doid=359576.359585 114

[168] R. Milner, "Communication and Concurrency," 1997. 114

[169] P. H. Welch and F. R. M. Barnes, "A CSP model for mobile channels," *Concurrent Systems Engineering Series*, vol. 66, pp. 17–33, 2008. 114

[170] A. Iqbal and B. Pottier, "Meta-simulation of large WSN on multi-core computers," in *SpringSim '10 Proceedings of the 2010 Spring Simulation Multiconference*, A. C. M. SCS, Ed., Orlando, United States, 2010, pp. 133:1–133:8. 115

[171] INMOS Limited, *Transputer instruction set: a compiler writer's guide*, 1988, includes indexes. 115

[172] ——, *Transputer reference manual*, 1988, includes index. Bibliography: p. 315-324. 115

[173] J. Moores, "CCSP-A portable CSP-based run-time system supporting C and occam." 116

[174] M. Raynal, *Distributed algorithms for message-passing systems*, 2013, vol. 9783642381. 118

[175] N. A. Lynch, *Distributed Algorithms*, 2008, vol. 9, no. 2. 118

[176] S. Corporation, "Sx1276/77/78/79 -137 MHz to 1020 MHz Low Power Long Range Transceiver. Rev. 4 - March 2015." Tech. Rep., 2016. 118, 120, 167, 199

[177] F. Factory, "Lora Fabian Shield (ES1.1): LoRa Shield for Arduino," https://froggyfactory.com/, 2018, [Online; accessed on 12 May 2018]. 120, 195

[178] S. Corp., "Semtech SX1272: Long Range, Low Power RF Transceiver 860-1000MHz with LoRa® Technology," https://www.semtech.com/products/wireless-rf/lora-transceivers/SX1272, 2018, [Online; accessed on 12 May 2018]. 120, 195, 199

[179] H. H. Luong and H. X. Huynh, "Graph-based model for geographic coordinate search based on balltree structure," in *Proceeding of the 17th International Conference, Daklak, Vietnam*, 2014, pp. 116–123. 124

[180] M. R. Abbasifard, B. Ghahremani, H. Naderi, G. Shakhnarovich, and T. Darrell, "A survey on nearest neighbor search methods," 2014. 124

[181] P.-N. Tan, M. Steinbach, and V. Kumar, "Association Analysis: Basic Concepts and Algorithms," *Introduction to Data mining*, pp. 327–414, 2005. [Online]. Available: http://www-users.cs.umn.edu/{~}kumar/dmbook/index.php 124, 130

[182] M. Muja and D. G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 11, pp. 2227–2240, Nov 2014. 124

[183] S. M. Omohundro, "Five balltree construction algorithms," Tech. Rep., 1989. 124, 129

[184] A. Csete, "Gpredict project," http://gpredict.oz9aec.net/, 2015. 134

[185] B. Pottier and P.-Y. Lucas, "Dynamic networks : Netgen tools," Jun. 2014, guide for simulation software, installation and use. [Online]. Available: http://hal.univ-brest.fr/hal-01294288 134, 135

[186] "Berlin Experimental and Educational Satellite-2 and -3," https://directory.eoportal.org/web/eoportal/satellite-missions/b/beesat-2-3, 2015, [Online; accessed on 4 November 2015]. 135

[187] "Sigfox Technology Overview," https://www.sigfox.com/en/, 2018, [Online; accessed on 25 April 2018]. 143

[188] W. SIG, "Weightless," http://www.weightless.org/, 2015, [Online; accessed on 25 July 2017]. 143

[189] I. C. Society, *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)– Object Model Template (OMT) Specification*, 2010. 163

[190] F. Kuhl, R. Weatherly, and J. Dahmann, *Creating Computer Simulation Systems: An Introduction to the High Level Architecture.* Upper Saddle River, NJ, USA: Prentice Hall PTR, 1999. 163

[191] B. Pottier and P.-Y. Lucas, "Concevoir, simuler, exécuter, Une chaine de développement pour réseaux de capteurs," in *Ubimob'12*, ser. ISBN 978.2.36493.018.6, P. R. et Nadine Rouillon-Couture, Ed. Bayonne, France: Cépaduès éditions, Jun 2012, pp. 94–107. [Online]. Available: http://hal.univ-brest.fr/hal-00710652 164

[192] T. P. Truong and B. Pottier, "Wireless network on coastal topologies: Parallel simulation of radio coverage on cellular systems," in *A Connected Ocean.* SeaTech week, Brest, Oct. 2016. 167, 216

[193] Modtronix, "Modtronix inAIR9b module," http://modtronix.com/inair9b.html?sef_rewrite=1&currency=AUD, 2017, [Online; accessed on 25 July 2017]. 167, 199

[194] G. Mei, "CudaChain: an alternative algorithm for finding 2D convex hulls on the GPU," *SpringerPlus*, vol. 5, no. 1, p. 696, 2016. [Online]. Available: http://springerplus.springeropen.com/articles/10.1186/s40064-016-2284-4 181

[195] Autodesk, "EAGLE:Schematic Editor and PCB Layout," https://www.autodesk.com/products/eagle/overview, 2017, [Online; accessed on 11 May 2018]. 183

[196] Sparkfun, “SparkFun Venus GPS with SMA Connector,” https://www.sparkfun.com/products/11058, 2017, [Online; accessed on 25 July 2017]. 183

[197] Modtronix, “Wireless SX1276 LoRa Module, 868MHz and 915MHz, +20dBm, 3.3V, SMA Connector,” http://modtronix.com/inair9b.html, 2017, [Online; accessed on 25 July 2017]. 183

[198] “Arduino home page,” http://www.arduino.cc. 183

[199] I. SiRF Technology, “NMEA Reference Manual,” https://www.sparkfun.com/datasheets/GPS/NMEAReferenceManual1.pdf, 2005, [Online; accessed on 25 July 2017]. 184

[200] C. GIS collective, “Slippy map projections explained,” http://giscollective.org/slippy-map-projections-explained, 2017, [Online; accessed on 28 July 2017]. 190

[201] GeoFabrik, “OpenStreetMap Data extracts,” http://download.geofabrik.de/, 2017, [Online; accessed on 28 July 2017]. 190

[202] “Site public pour tvm,” http://concurrency.cc. 199

[203] C. Semiconductor, “CY8CKIT-001 PSoC ® Development Kit Guide,” Tech. Rep., 2012. 209

[204] V. Ranjan, “Transformers in the embedded world,” no. September, pp. 1–6, 2008. 209

[205] E. H. C. Alex Doboli, *Introduction to Mixed-Signal Embedded Design.* Springer, 2007. 209

[206] R. Spurrett, “Back Down to Earth - Semtech and Lacuna Receiving Messages from Space,” http://blog.semtech.com/back-down-to-earth, 2018, [Online; accessed on 12 May 2018]. 212

[207] Qthid, “Qthid Funcube Dongle Controller,” http://oz9ªec.net/radios/funcube-dongle/qthid-funcube-dongle-controller, 2018, [Online; accessed on 02 July 2018]. 212

[208] Quisk, “http://james.ahlstrom.name/quisk/,” http://james.ahlstrom.name/quisk/, 2018, [Online; accessed on 02 July 2018]. 212

[209] gqrx, “Gqrx SDR,” http://gqrx.dk/, 2018, [Online; accessed on 02 July 2018]. 212

[210] T. Hoang van, H. Hiep Xuan, V. Cong Phan, and B. Pottier, “A federation of simulations based on cellular automata in cyber-physical systems,” *EAI Endorsed Transactions on Context-aware Systems and Applications* , vol. 16, no. 7, Feb. 2016. [Online]. Available: http://hal.univ-brest.fr/hal-01294082 215

[211] H. H. L. B, T. P. Truong, and K. M. Nguyen, “Optimizing the Light Trap Position for Brown Planthopper (BPH) Surveillance Network,” vol. 1, pp. 165–178. 215

[212] B. H. L. B, T. P. Truong, and K. M. Nguyen, “An Hierarchical Scheduled Algorithm for Data Dissemination in a Brown Planthopper,” vol. 1, pp. 246–263, 2016. 215

[213] T. P. Truong, H. X. Huynh, and B. Pottier, “A simulation of radio propagation based on cellular automata,” Tech. Rep., 2016. 216

[214] T. P. Truong, Udrekh, and B. Pottier, “Long-range Communications: A Promising Technology for Environmental Monitoring Applications in Developing Countries,” Tech. Rep., 2018. 217

A
High Level Architecture

The High-Level Architecture (HLA), a standard for distributed simulations, is used for the interoperability and reusability of several simulations. In HLA terminology, the whole solution that needs to be modeled and simulated is represented by a federation. Each simulator (sub-models) referring to the federation is called a federate. A set of federates is connected via Run Time Infrastructure (RTI). In this case, RTI can be considered as distributed operating systems for cooperating system federates.

Federates exchange data together via RTI. For example in a federation there are 2 federates which the second federate relies on values provided by the first one. Firstly, the first federate publishes its states to RTI and the second one needs to subscribe them. Whenever the second federate receives states from RTI, values of state variables in its simulator are updated to make suitable behaviors. Similarly, the second federate publishes its states but the insect physical federate does not need to subscribe.

Besides, both federates need to register synchronous points to synchronize their data as well as activities. This can be done by the time management service of RTI.

An architecture is designed in accordance with the high-level architecture (HLA). In which, the entire system is viewed as a federation containing several federates linked via the central component run-time infrastructure (RTI). The HLA is formally defined by three components [189, 190].

1. A set of rules describes the responsibilities of federates and their relationship with RTI. An example is that all exchange of data among federates should occur via the RTI during a federation execution.

2. An interface specification provides services for managing federates and interactions. For example, it indicates how a federate join or leave a federation.

3. An Object Model Template defines how information is communicated between federates, and how the federates and federation have to be documented (using Federation Object Model FOM). FOM defines the shared objects, attributes, and interactions for a whole federation.

For the design of cellular automata (CA) systems, time steps must match a sequencing reality. It can be very fast in the case of physical propagation (waves, sounds, earthquakes), or very slow, as in the evolution of environmental biology. Simulation of CA is in general driven by the time, so one simulation step match one CA step and one physical step. Simulation steps can be folded to match computer characteristics.

CA can be composed using HLA (Figure A.3), and they can also be composed with another framework, as it is the case for network simulation, with a synchronous simulation of WSN.

It has been shown that mobile visitor nodes (vehicles, satellites) can also be managed in this framework [191, 4].

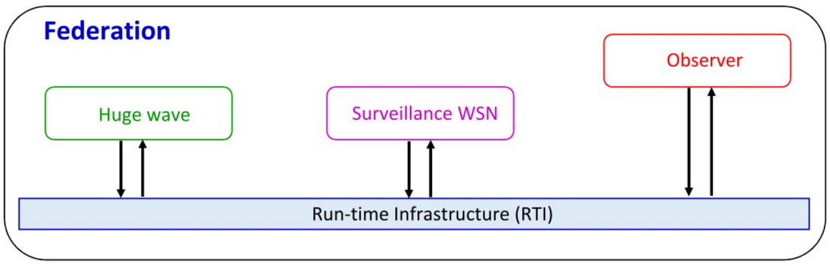


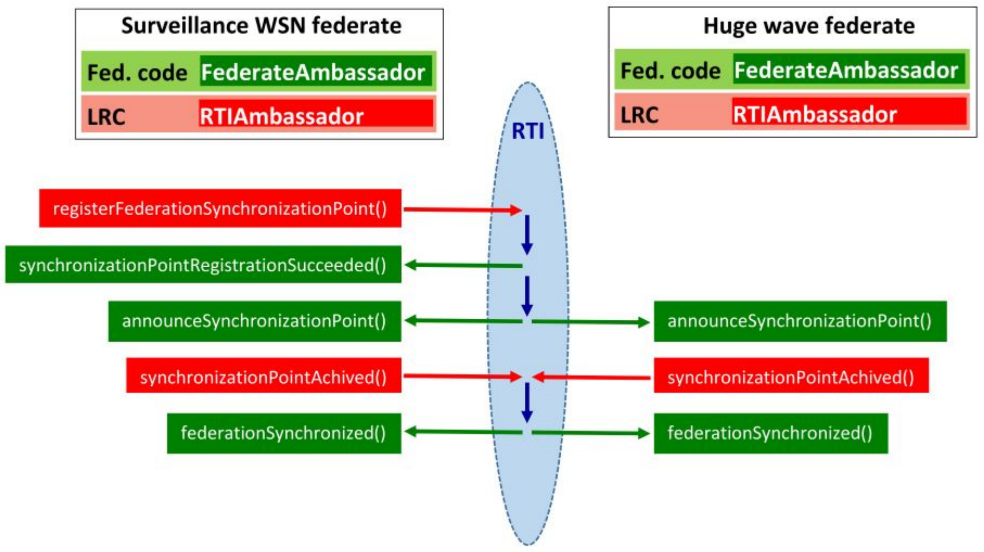
Figure A.1 – A HLA federation of three federates: huge wave, surveillance WSN, and observer.

In HLA terminology, different simulators (*federates*) could be assembled in an interconnection, which is called a *federation*, with communication being supported by a *Runtime Infrastructure* (RTI). The RTI provides services for simulation control, synchronization and data exchange such as *Federation Management*, *Time Management*, and *Object Management* [109].

Federation Management Synchronization functions composing of *RTIambassador* and *FederateAmbassador* allows federates to communicate through explicit synchronization points. It is possible to create and destroy federations, join and resign federates to federations, save and restore federations, and observe federate synchronization points. Figure A.2 depicts the RTIambassador and FederateAmbassador services support the synchronization capability [109].

As can be seen in Figure A.2[1], *registerFederationSynchronizationPoint()* method of FederateAmbassador needs to be initialized first to establish a named checkpoint enabling to synchronize federates according to federation-defined semantics. If the registration succeeds, a synchronization point is produced by *synchronizationPointRegistrationSucceeded()* service.

[1]Note that LRC is an abbreviation of Local RTI Components.



Surveillance WSN federate
Fed. code
FederateAmbassador
LRC
RTIAmbassador
Huge wave federate
Fed. code
FederateAmbassador
LRC
RTIAmbassador
RTI
registerFederationSynchronizationPoint()
synchronizationPointRegistrationSucceeded()
announceSynchronizationPoint()
announceSynchronizationPoint()
synchronizationPointAchived()
synchronizationPointAchived()
federationSynchronized()
federationSynchronized()

Figure A.2 – Federate synchronization between surveillance WSN federate and huge wave federate produce synchronization and data exchange. Services sent from RTI to federates (green) and federates to RTI (red) [109].

Then all federates will receive an announcement through a *announceSynchronizationPoint()* callback. The federates give confirmation to the federation about their achievement of synchronization with *SynchronizationPointAchieved()*. When all synchronization or resignation services are completed, the federation informs to the relevant federates in form of a *federationSynchronized()* callback.

As shown in Figure A.4, we present the federation of two main federates representing the two parallel simulations being huge ocean waves or tsunami, and tsunami early warning network (Figure A.1). An observer federate is also designed to visualize the federation.

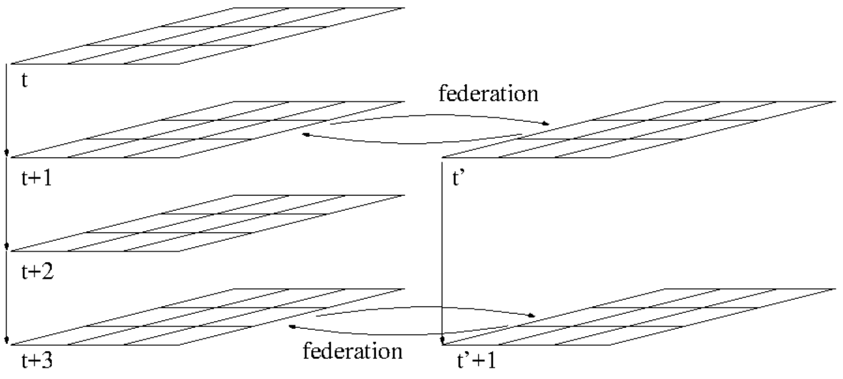


Figure A.3 – Cell systems progress step by step according to time steps. Several cellular systems can be federated to exchange data synchronously, that represents a different aspect of the same problem (HLA), following different clocks. Federation are data exchanges occurring between different simulations that contribute to the same problem. Network sensor activities can be federated to physical simulation.

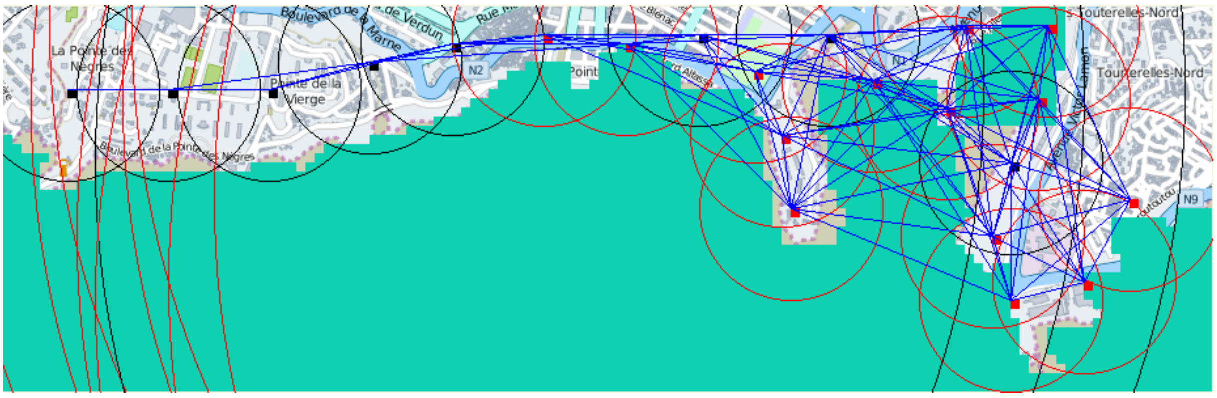


Figure A.4 – Simulation of a huge wave together with reaction of a warning system in Martinique seashore. The figure shows the wave hitting the shore (physical simulation) federated with a wireless sensor surveillance network. It is noticeable that the wave can damage the monitoring system, that should react appropriately. For instance, the behavior of an abstract surveillance network is simulated in which each sensor node is depicted by a square dot with sensing range (small circle) and communication range (larger circle). When a huge wave approaches coastline within a distance less than nodes' critical warning condition, this sensor node broadcasts alarm signal to its neighbors indicated in red color.

B Computing radio coverages: experiments on archipelagoes

Each rocky shore is an ecosystem where it is critical to understand, protect and preserve biodiversity in the context of climate changes. Efficient autonomous observatories can be developed and deployed to collect data with benefits for environment management, thanks to compact electronic devices and new radio systems [192].

Wireless sensor networks (WSNs) offer attractive solutions for observation because of the large availability of physical sensor devices, and compact radio link transceivers, for example Modtronix inAIR9b in Fig. B.1 [193]. Radio links enable communications between sensors, gateways, and information systems on permanent or periodic basis [121].

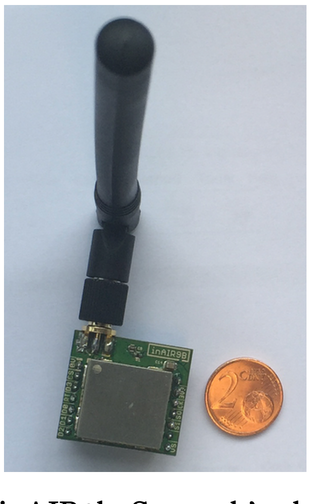
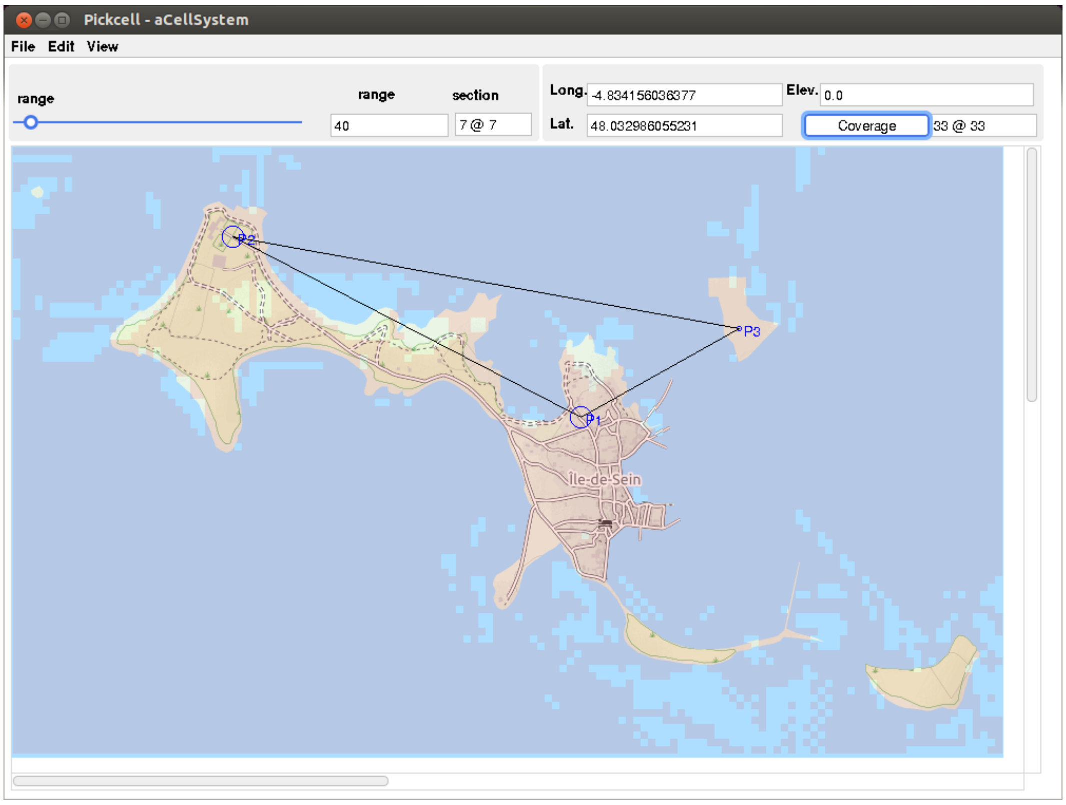


Figure B.1 – LoRa transceiver inAIR9b: Semtech's chip sx1276, 868-915 MHz [176].

In the case of marine environments such as shores or islands, it is practically difficult to design radio sensor deployments having an accurate *coverage*. *Coverages* is the way to specify where information is accessible, whatever it is either radio signal, biological, or physical influences. Therefore computing coverage is essential for monitoring purposes and can take various forms. The case of marine shores is difficult because of complex topologies, with rocks of various shape and elevations interleaved with the sea. Managing the sea from radio buoys is another element.

This work relates to principles and results-oriented to the exploration of radio link capabilities, taking into account sea shore geography topologies. By modeling geography into cell systems, it becomes possible to simulate many physical phenomena such as wave effects, rain effects, pollution, etc. It is also possible to model the natural behavior of WSN radio signal propagation. This defines the ability of signals coming from one point to reach another point in line-of-sight (LoS).

For example, the results of the LoS computation are shown in Fig. B.2, where cells highlighted with pink color can communicate with the P1 emitter.



range

range: 40

section: 7 @ 7

Long. -4.834156036377

Elev. 0.0

Lat. 48.032986055231

33 @ 33

Figure B.2 – Example of a coverage for *Ile de Sein*: cells are illuminated by a WSN consisting of 3 nodes with P1 as a master. There are 9971 cells under the communication coverage of this WSN corresponding to 89.58% area. The Pickcell window displays geo-location and elevation at a mouse position, the cell size in pixels and meters (33 meters in this case). The positions of P1, P2, P3 were decided manually, each selection showing immediately the cover of the respective selection.

Computing covers in presence of obstacles are known to be a compute-intensive task, in the complexity class of 3D image synthesis because emitted rays must be compared to each other point in an image taking obstacles into account. We explain and discuss a massively parallel execution led on cellular systems representing the geographic zone. Computations were firstly achieved on communicating processes suitable for multi-core processors, then they were ported on Graphics Processing Units (GPUs) producing performances in the real-time order.

The tools developed to allow to select arbitrary study zones from a specific map browser called *QuickMap*[1]. As a result, automatic and manual coverage computations were applied to a set of archipelagos. Practical results are given in terms of performances and functional results section B.1.

B.1 Practical effects

In practice, for execution, the lab tools allocate cells in the accelerator memory and represent channel connectivity by data structures. The execution itself is done by a so-called *CUDA kernels* sweeping the node system. We have been able to map and execute systems with as many as 125000 cells. The level of effective parallelism is high: common GPUs have several hundred processors, thus the computations finish at an impressive speed (see Table B.1).

Using these tools is easy and allows to investigate case studies rapidly. As a demonstration, 6 complex shores were explored with a variety of geographic resolutions, system sizes, and cell sizes. The table B.1 summarize the results. Computation delays are compatible with a fast mobile that needs to guess what is happening below or beside, provided that geographic data have been stored previously.

Table B.1 – Six different experimental areas: *Size of cell* column indicates the actual sizes in meter. The third column gives total of cells in cell systems. In the next columns, statistics show how many cells would be visible and percentage coverage associated with figure of deployed emitters. The computation times, column *Time*, which depend on both the number of cells and the number of emitters, were evaluated, as shown in the last column of this table.

Area name	Size (m)	Cells	Emitters	Visible cells	Coverage	Time (ms)
Sept Iles	38	43610	3	39064	89.58%	365.39
Scyllies	191	12544	3	10523	83.89%	81.219
Ile de Sein	33	10988	3	9971	90.74%	63.98
Chausey	33	32802	2	27594	84.12%	174.48
Brest bay	76	55986	2	49694	88.76%	304.55
I'ile Saint-Nicolas	38	34965	3	33131	94.75%	285.46

[1]Tool designed by Pierre-Yves Lucas that can handle a variety of tile systems for maps or satellite imaging

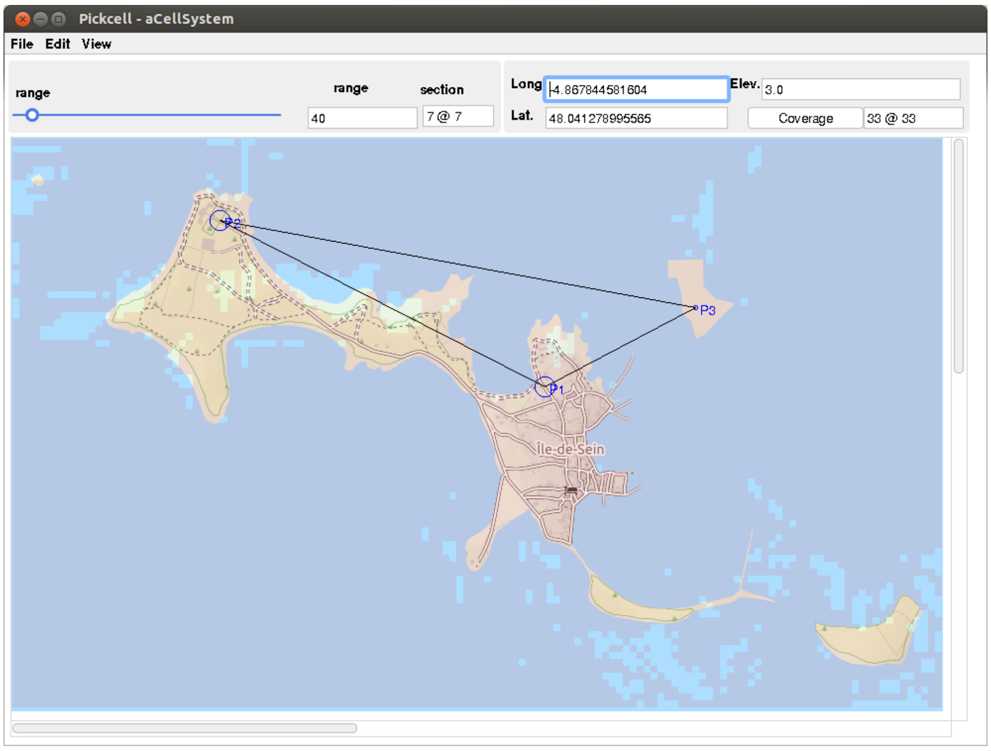


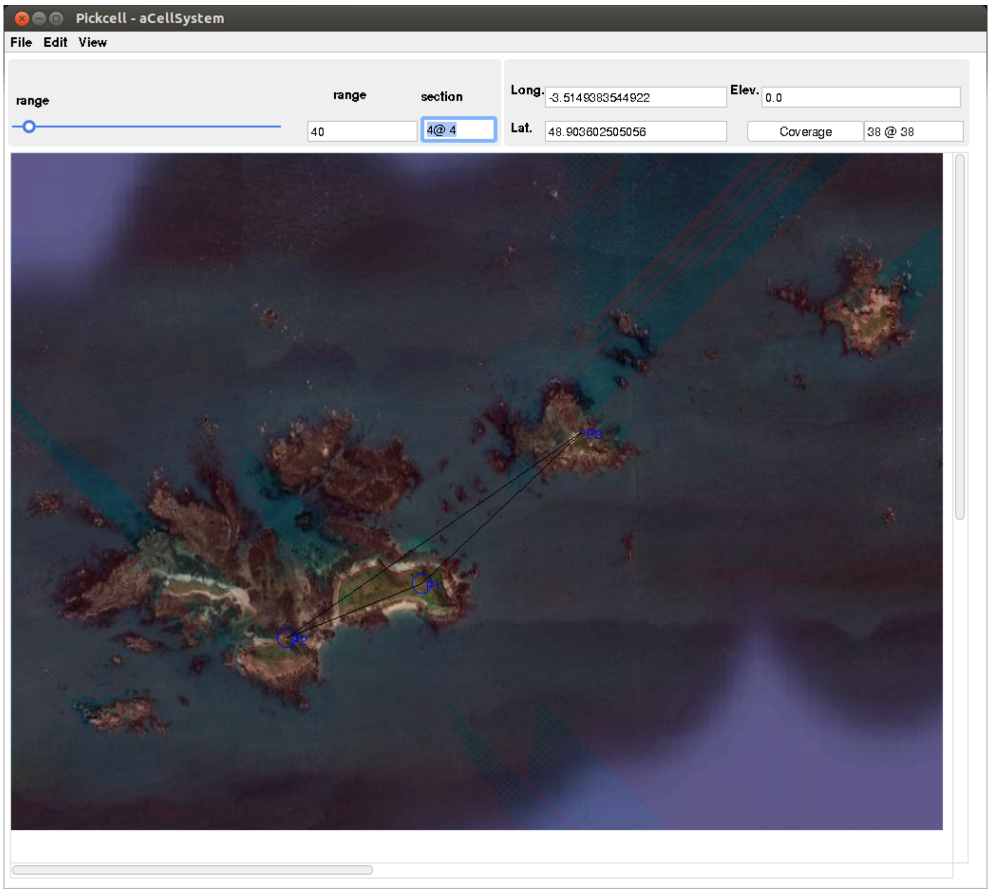
Figure B.3 – An experiment with the offset 5 meters of elevation for emitters was conducted for the WSN in Ile de Sein. This can be described as the height of transceiver antennas. Consequently, the total of cells within the coverage of the WSN increases up to 10385 cells corresponding to 94.51% area. This result proves that the elevation is a key factor in LoS communications.

B.1.1 Coverage exploration

Our tools provide two ways in order to determine emitter positions: manual selection or coverage exploration strategy.

In manual mode, each time an emitter is located on the map (Quickmap/Pickcell) the coordinate (longitude, latitude) and elevation of this point are used as parameters to execute the LoS algorithm on GPUs. The obtained result is cells within the LoS communication range of this root associated with the received power signal. These values are fetched back into Pickcell to show corresponding communication coverage. Another cell in this coverage can be chosen to put next emitters or repeaters as planning a wireless network. Due to a large number of cells and complexity of simulation, this process needs high computing power to perform in real time. A video clip to describe this process can be accessed at the following link:*https://youtu.be/iO94UiFx7KE*

For coverage exploration, the highest point to put the first emitter is found out by the program automatically. The values of this point are also sent to the CUDA procedure running on GPUs to determine a set of cells under its coverage. This coverage area is shown on PickCell and then a cell in the area is selected to put the next emitter based on one of the different strategies such as maximum distance from the previous emitter, maximum number



range
range: 40
section: 4@ 4
Long. -3.5149383544922
Elev. 0.0
Lat. 48.903602505056
Coverage: 38 @ 38

Figure B.4 – A satellite photo tile is used to present a possible sensor deployment on Sept Iles archipelago. QuickMap tool provides several types of maps such as OpenStreetMap, Thunderforest, Google map, etc. (for more details see Fig. B.5). Moreover, Pickcell tool also allows to process on images, for example, weather radar images, X-ray/MRI images, to model and simulate physical facts based on cellular methods.

of invisible neighbors, and so on. The process continues in the same way until all cells are highlighted.

B.2 A summary of tool flow to predict communication coverage

PickCell/NetGen is a set of tools sharing abstract models for a network of sensors and physical simulation. They also share software generators for concurrent executions [8] that are compatible and mergeable.

PickCell allows the analysis of a geographical zone, in the form of geo-localized cells in 2 dimensions. The cells are defined on a browser of maps. The cell systems can be produced with additional information such as elevation, geological data. The cells let the computation of radio signals line-of-sight taking into account the obstacles.

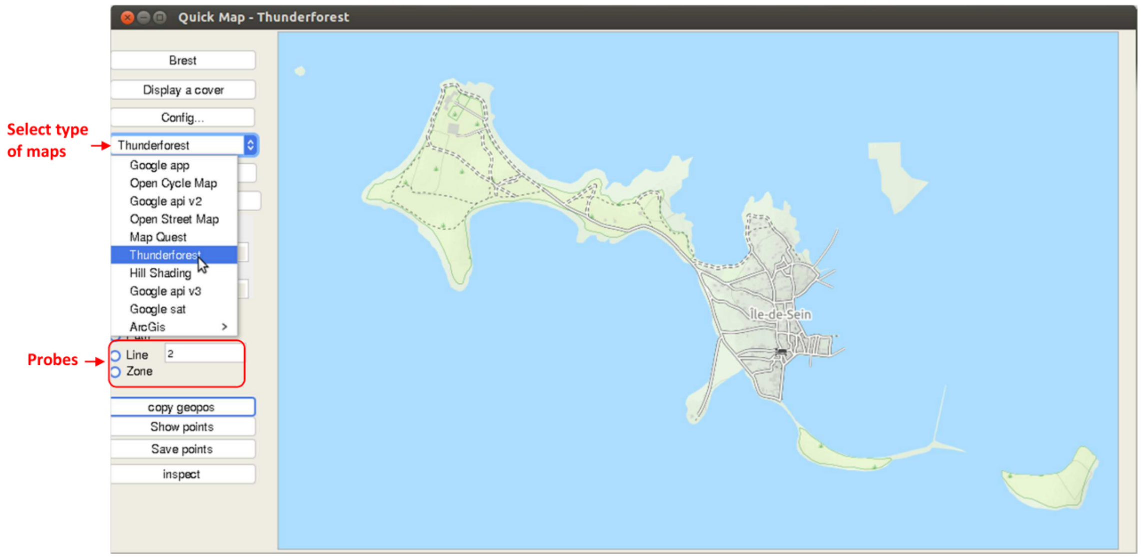
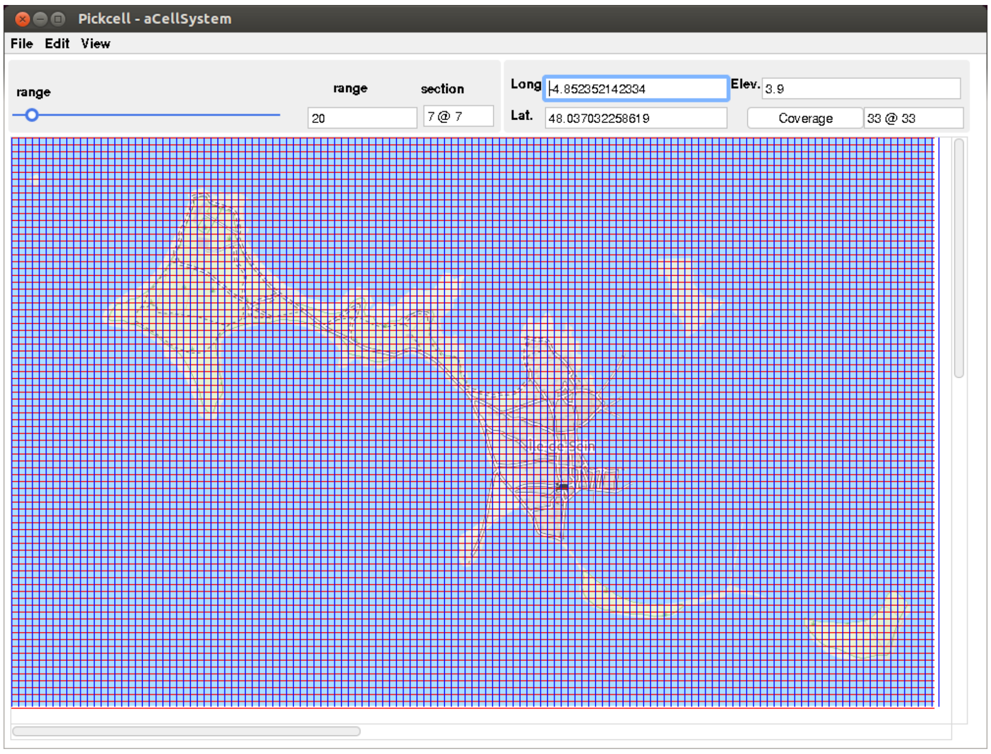


Figure B.5 – Quickmap: a map browser allowing to select several kind of maps, change tile and walk anywhere.

The common flow proceeds top-down from space to modeling and simulation as follows:

- *Space* descriptions are selected from maps, photographs, measures, human body model, nano-architecture model.
- *Cell partition of Space* regular segments are produced keeping geometric references (possibly geo-coordinates).
- *Cell subsystems* are produced by grouping based on criteria (colors, elevation, and other data).
- *Cellular automata* are built by connecting neighbor cells according to a connectivity pattern (Von Neumann, Moore).
- *Cell behavior* describes physical evolution data with transitions and physical neighborhood influences.
- *WSN sensor systems* are networks of measure points with geometric references, wireless communication capabilities, estimated coverage ranges.
- *WSN behavior* is the logical behavior of sensors cooperating inside networks, achieving local perception and contributing to distributed algorithms.

Merging network and physical simulation defines a cyber-physical machine model embedding physical reality, perception, distributed decision, and possible physical control in the real world. Many applications can come from this approach such as flooding, pollution, wildlife simulations.



range

range: 20

section: 7 @ 7

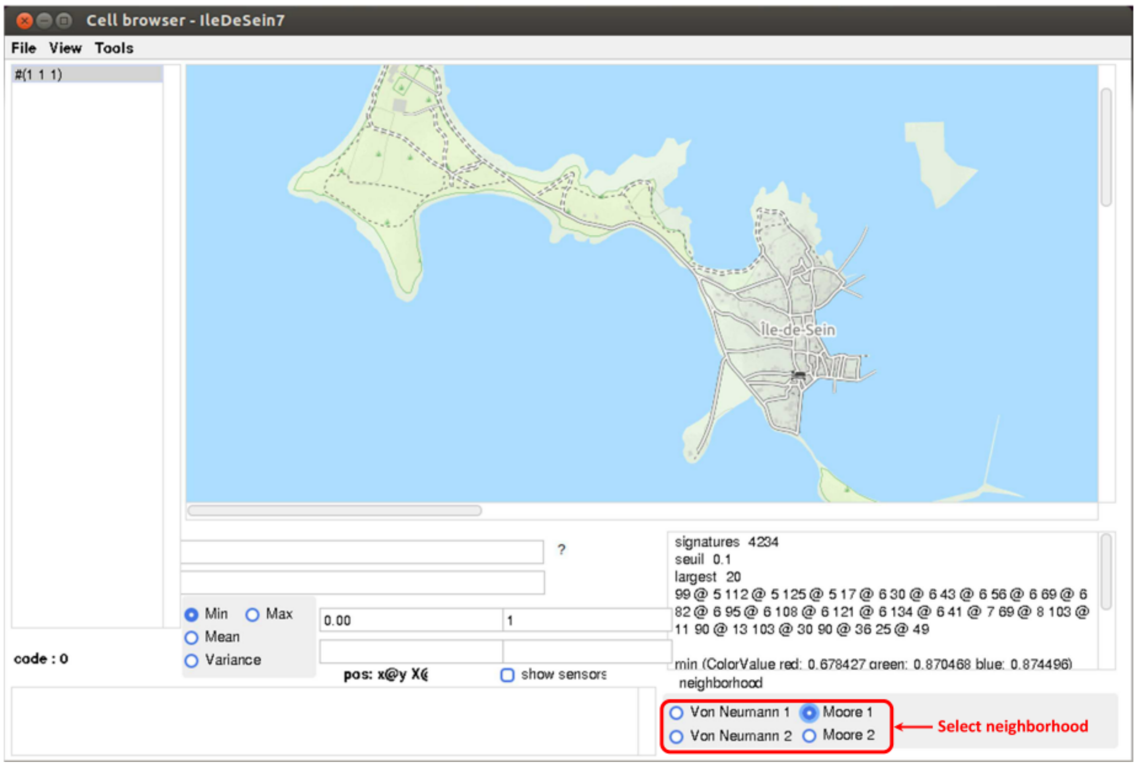
Long: -4.852352142334

Elev.: 3.9

Lat.: 48.037032258619

Coverage: 33 @ 33

Figure B.6 – Cell system: (x,y,z), distance, range.



#(1 1 1)

code : 0

[✓ Min] Max Mean Variance

0.00 1

pos: x@y X@

show sensors

```
signatures  4234
seuil  0.1
largest  20
99 @ 5 112 @ 5 125 @ 5 17 @ 6 30 @ 6 43 @ 6 56 @ 6 69 @ 6
82 @ 6 95 @ 6 108 @ 6 121 @ 6 134 @ 6 41 @ 7 69 @ 8 103 @
11 90 @ 13 103 @ 30 90 @ 36 25 @ 49

min (ColorValue red: 0.678427 green: 0.870468 blue: 0.874496)
```

neighborhood: Von Neumann 1 [✓ Moore 1] Von Neumann 2 Moore 2

Figure B.7 – Cell weaving and simulation synthesis. Cell system specification produces automatically.

C
Preliminary Results on Scheduling Algorithms of LoRa Network

C.1 Basic algorithm

According to LoRa modulation [160] the time on air (ToA), duration time required for sending a packet, depends on three critical parameter bandwidth (BW), coding rate (CR) and spreading factor (SF).

The operation of the LoRa network can be divided into two states. First, it is necessary to carry out programs for building a network and scheduling the radio links. Second, the network operates rely on the former configuration to collect and store data.

In the beginning, base station spread out a *"hello"* message to explore other nodes within its communication coverage. Broadcast message in LoRa protocol [107, 55] is defined by destination address 255 (0xff). All active nodes receiving this message need to respond by an acknowledgment (*ACK*) if they would like to participate in this network.

After knowing the information about neighbors, a critical algorithm in order to enhance the performance of nodes is performed. The input data of this optimization program is radio signal strength indicators corresponding to received messages at the base station. The result of optimizing process, the set of LoRa parameters for each node is proposed. This work aims at improving the operation of the network by increasing the data rate, saving transmit power and still retaining the robustness of connections.

From Semtech's document [159], LoRa network communication time is reduced significantly as LoRa chip working in the configuration with wider bandwidth, lower coding rate denoting less redundant bits, and slower chirps corresponding to higher SF values. The duration for sending a packet in a certain configuration of LoRa can be found as follows.

The symbol rate is given by

$$R_{sym} = \frac{BW}{2^{SF}} \tag{C.1}$$

so the time of one symbol is

$$T_{sym} = \frac{1}{R_{sym}} \tag{C.2}$$

Besides time of preamble can be derived from

$$T_{preamble} = (n_{preamble} + 4.25) \times T_{sym} \tag{C.3}$$

with $n_{preamble}$ is the programmed preamble length

From the structure of the packet, the number of payload symbols is calculated by the formula

$$PL_{sym} = 8 + max\left(ceil\left(\frac{8PL - 4SF + 28 + 16 - 20H}{4(SF - 2DE)} \right) \times (CR + 4), 0 \right) \tag{C.4}$$

where

PL is the number of payload bytes (1 to 255)

SF is the spreading factor (6 to 12)

IH=0 when the header is enabled, IH=1 when no header is present

DE=1 when LowDataRateOptimize=1, DE=0 otherwise

CR is the coding rate (1 corresponding to 4/5 and so on)

The formula gives the time of payload as

$$T_{payload} = PL_{sym} \times T_{sym} \tag{C.5}$$

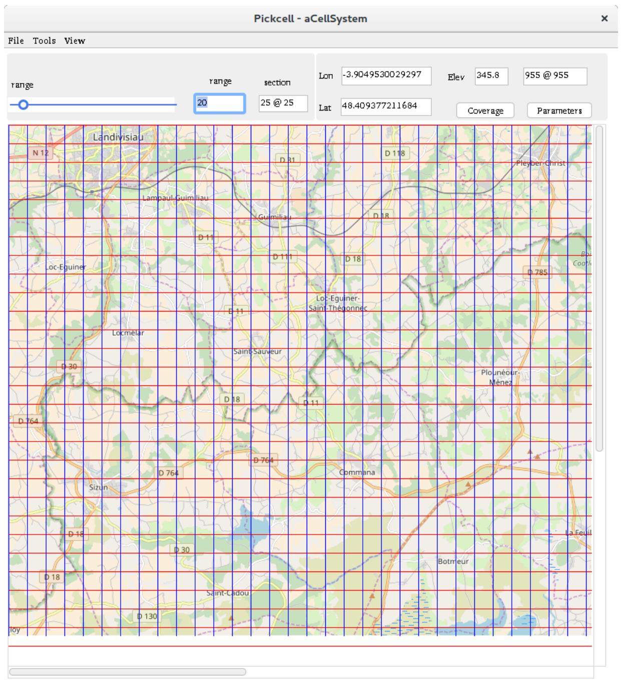
Finally, the duration of the packet is

$$T_{packet} = T_{preamble} + T_{payload} \tag{C.6}$$

Period of times for collecting data for each node is calculated by applying Equation C.6. Consequently, a scheduling algorithm is utilized in order to determine communication time slots for sensor nodes. The information about time slots is then delivered to all nodes in the network. A super-frame stored at the base station is composed of all time slots. It helps to build up a scheduling table to regulate all connections between the base station and sensor nodes. Based on scheduling communications, the long-range network for data dissemination is performed effectively.

C.2 A study case of scheduling LoRa networks

This section describes design flow to optimize LoRa networks. Figures from C.1 to C.6 provide visual explanations in step-by-step how to deploy and optimize a LoRa star network with our tools.

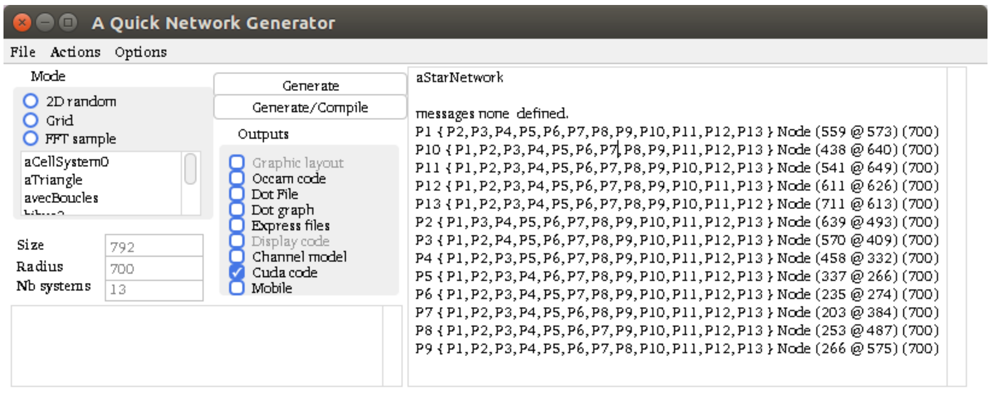


range 20 section 25 @ 25

Lon -3.9049530029297 Elev 345.8 955 @ 955

Lat 48.409377211684

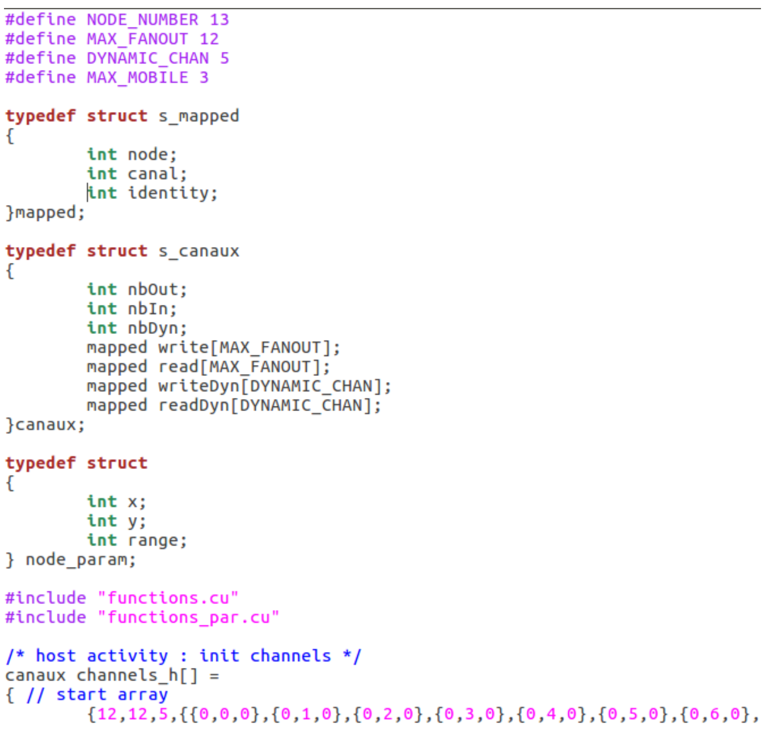
Figure C.1 – QuickMap is used to select a zone at Roc'h Trevezel (a mountain area in Bretagne, France) for a LoRa network deployment.



Size 792
Radius 700
Nb systems 13
[✓ Cuda code]

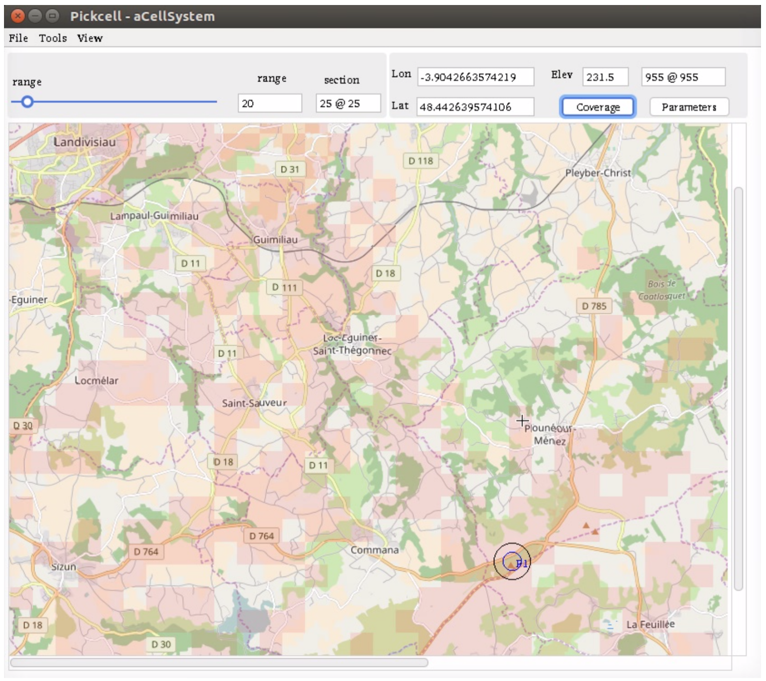
```
aStarNetwork

messages none  defined.
P1 { P2,P3,P4,P5,P6,P7,P8,P9,P10,P11,P12,P13 } Node (559 @ 573) (700)
P10 { P1,P2,P3,P4,P5,P6,P7,P8,P9,P11,P12,P13 } Node (438 @ 640) (700)
P11 { P1,P2,P3,P4,P5,P6,P7,P8,P9,P10,P12,P13 } Node (541 @ 649) (700)
P12 { P1,P2,P3,P4,P5,P6,P7,P8,P9,P10,P11,P13 } Node (611 @ 626) (700)
P13 { P1,P2,P3,P4,P5,P6,P7,P8,P9,P10,P11,P12 } Node (711 @ 613) (700)
P2 { P1,P3,P4,P5,P6,P7,P8,P9,P10,P11,P12,P13 } Node (639 @ 493) (700)
P3 { P1,P2,P4,P5,P6,P7,P8,P9,P10,P11,P12,P13 } Node (570 @ 409) (700)
P4 { P1,P2,P3,P5,P6,P7,P8,P9,P10,P11,P12,P13 } Node (458 @ 332) (700)
P5 { P1,P2,P3,P4,P6,P7,P8,P9,P10,P11,P12,P13 } Node (337 @ 266) (700)
P6 { P1,P2,P3,P4,P5,P7,P8,P9,P10,P11,P12,P13 } Node (235 @ 274) (700)
P7 { P1,P2,P3,P4,P5,P6,P8,P9,P10,P11,P12,P13 } Node (203 @ 384) (700)
P8 { P1,P2,P3,P4,P5,P6,P7,P9,P10,P11,P12,P13 } Node (253 @ 487) (700)
P9 { P1,P2,P3,P4,P5,P6,P7,P8,P10,P11,P12,P13 } Node (266 @ 575) (700)
```

Figure C.2 – PickCell controls segmenting and weaving processes and NetGen produces data for radio coverage computation. In addition, a graph file describing the abstract network and dynamic links for mobile nodes are also provided by this tool.



```
#define NODE_NUMBER 13
#define MAX_FANOUT 12
#define DYNAMIC_CHAN 5
#define MAX_MOBILE 3

typedef struct s_mapped
{
        int node;
        int canal;
        int identity;
}mapped;

typedef struct s_canaux
{
        int nbOut;
        int nbIn;
        int nbDyn;
        mapped write[MAX_FANOUT];
        mapped read[MAX_FANOUT];
        mapped writeDyn[DYNAMIC_CHAN];
        mapped readDyn[DYNAMIC_CHAN];
}canaux;

typedef struct
{
        int x;
        int y;
        int range;
} node_param;

#include "functions.cu"
#include "functions_par.cu"

/* host activity : init channels */
canaux channels_h[] =
{ // start array
        {12,12,5,{{0,0,0},{0,1,0},{0,2,0},{0,3,0},{0,4,0},{0,5,0},{0,6,0},
```

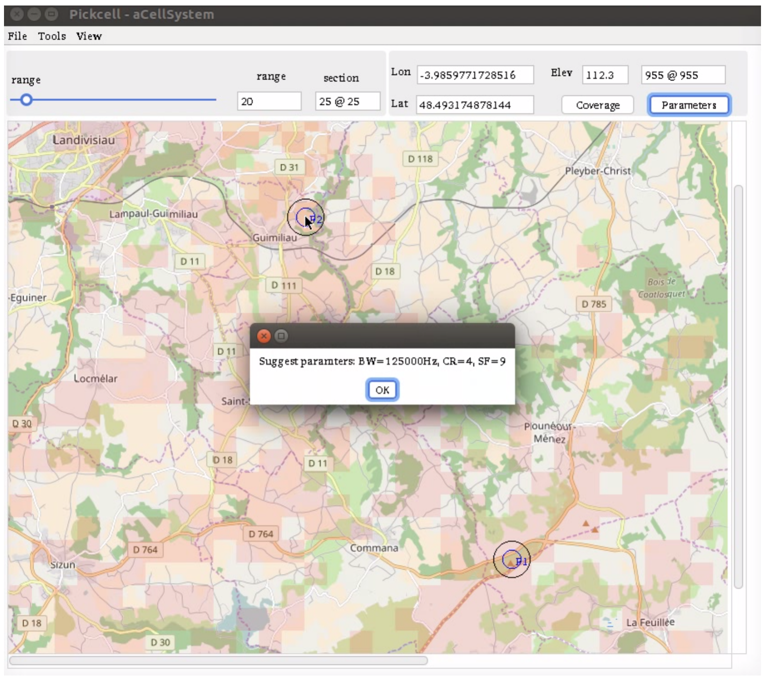
Figure C.3 – A fragment of CUDA code was generated by using PickCell/NetGen tool-set. Parallel computation can be achieved by associating cells with communicating processes. Channels are mapped on share memory for exchanging data synchronously as discussed in Section 2.2.4.



range

range: 20
section: 25 @ 25
Lon: -3.9042663574219
Lat: 48.442639574106
Elev: 231.5
955 @ 955

Figure C.4 – A place where an emitter to be located can be select by mouse click on the map. This also invoke a coverage computation on GPUs. The coverage computation result is shown on PickCell by coupling Smalltalk and CUDA using DLL, comprising of cells highlighted in pink.



range 20

section 25 @ 25

Lon -3.9859771728516

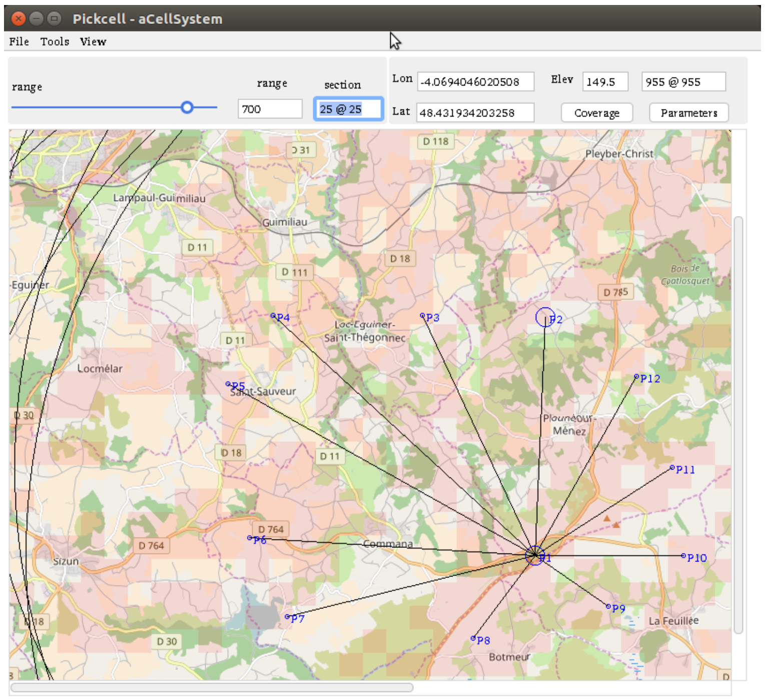
Lat 48.493174878144

Elev 112.3

955 @ 955

Suggest paramters: BW=125000Hz, CR=4, SF=9

Figure C.5 – By locating a node at a visible point from the base station, optimization process gives the suggestion of LoRa parameters for that node. These values for LoRa configuration aims at optimizing both robust of link and data rate and energy consumption taking into account topography of geography.



range 700

section 25 @ 25

Lon -4.0694046020508

Lat 48.431934203258

Elev 149.5

955 @ 955

Figure C.6 – The result is a star network with LoRa's optimized parameters BW, CR, SF of every node to enhance the performance system.

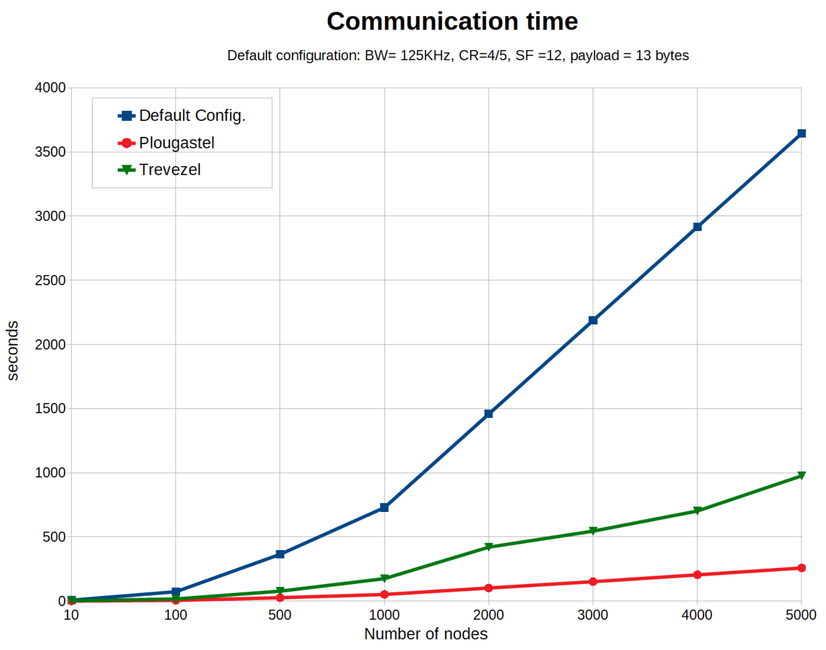


Figure C.7 – The statistics of communication times in two experiments with respect to the complexity of terrains. The blue line is communication of the network corresponding to LoRa's default setting (BW=125 KHz, CR=4/5, SF=12). After optimizing, communication times for the Roc'h Trevezel and the Plougastel are presented by the green and red line respectively. Note that a number of sensors are generated randomly in simulations.

The communications of LoRa networks are strongly impacted by transmission environment in reality. Hence, the actual coverage prediction must be taken topographic complexity into account as a critical factor. Our proposed tool flow is a useful approach to facilitate the deployment of a long-range communication network for monitoring environment, especially in complex topographical areas.

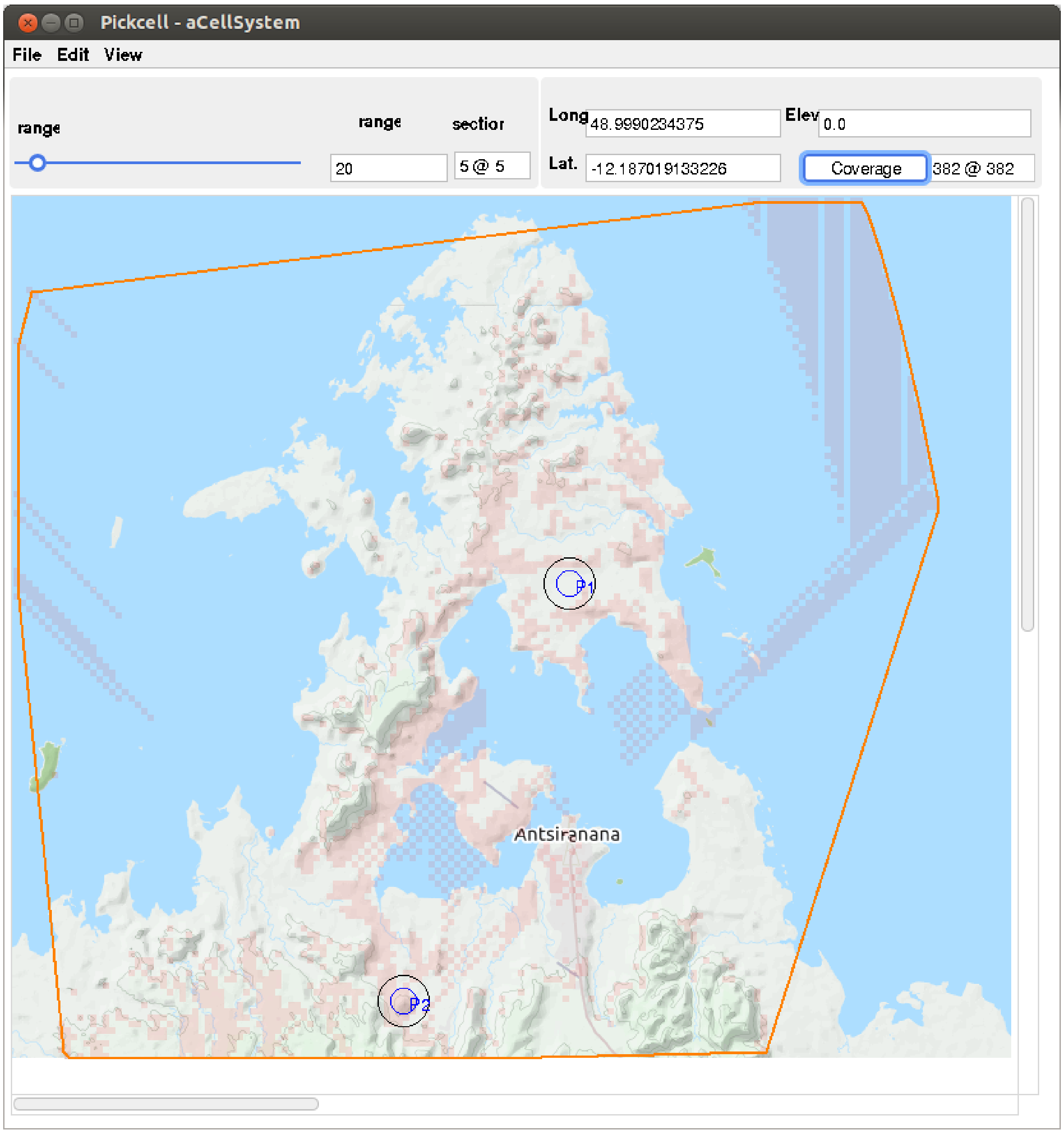
D
Communication boundary of WSN: an implementation of parallel 2D convex hull algorithm

To indicate the boundary of WSN: sensing coverages, communication coverages, or a group of cells in a zone having the same features such as pollution density, a number of insects, etc, our approach is presented here to employ a parallel algorithm using NVIDIA Thrust library to compute the convex hull performing on GPUs and display its results on QuickMap/Pickcell tool.

The general flow of processes for a WSN's communication boundary is described as following:

- **Step 1**: Select experimental zone and generate data, cell system, for simulations (Quickmap/PickCell/NetGen).
- **Step 2**: Launch invocation of CUDA program running on NVIDIA GPUs for radio signal propagation simulation based on cellular automata approach. It produces the communication coverage of a given emitter.
- **Step 3**: Obtain the simulation result and then reform it to an linear 2-dimension array consisting of (x,y) locations of cells. Noticed that (0,0) reference is the cell at the furthest top-left corner.
- **Step 4**: Invoke 2D convex hull routine on GPUs. This algorithm is derived from CudaChain [194] with several modifications to fit with our system such as data type, an organization of data.

- **Step 5**: Capture convex hull computation results, (x,y) locations of vertices, and draw this boundary on PickCell. As mentioned at *Step 4*, our system operates on geographic data (latitude, longitude), so that it is necessary to use float number in double precision with at least 9 digits floating point. The structure of data must be convenient both for computing on GPUs also retrieving from VisualWorks aiming at showing the result on the GUI of PickCell.



range: 20
sectior: 5 @ 5
Long: 48.9990234375
Elev: 0.0
Lat.: -12.187019133226
382 @ 382

Figure D.1 – The communication boundary of a network is displayed on the map for the North of Madagascar.

E
System tools presentation

E.0.1 Boards and components

Our aim is to establish a radio link between a fixed node (base station) and a mobile node moving around for collecting the strength of radio signal corresponding to geology location. For sake of simplicity, we use the Arduino board as a micro-controller. We designed and implemented a board namely RECoco (Radio Estimation Coverage). Free software Autodesk Eagle [195]: schematic editor and PCB layout were employed to design RECoco. RECoco shield can be stacked on Arduino board including a GPS module and a LoRa transceiver. Venus GPS module [196] used to obtain the geo-location of the node from the satellite. Modtronix inAir9b LoRa transceiver [197] is a radio circuitry for sending and receiving data in long distance.

To collect the data from a mobile node moving around, at the base station (BS) (fixed node) both of nodes are equipped RECoco. At the mobile node, the information consisting of local time, latitude, longitude, altitude from GPS is packed and then sent to the BS via LoRa radio channel. A particular map browser and control called QuickMap [121] running on PC at BS side are employed to establish the connection between PC and RECoco board through USB port for receiving messages from the mobile node. The received message is unpackaged to retrieve data and then display position of the node on the map in real time. Received signal strength indicator (RSSI) captured at LoRa circuitry is used to evaluate the quality of radio connection [22]. These data are also logged in files or insert into the database at data center directly through 3G/4G connection available on PC.

As can be seen in Figure E.1b, group of jumpers (1) are compatible with SPI driver signals (MISO, MOSI, SCK) to fit on either Arduino mega2560 or UNO [198] and jumpers (2) and (3) aim at switching slave devices (SS) and interrupt pins. Consequently, interrupt routines utilize to handle LoRa chip for long-range communications. In addition, the serial port on

Arduino board is used to get the coordinate location of the node from the integrated GPS module and connect to PC can be configured by using a jumper (4).

GPS modules equipped processors and antennas receive the data sent by the satellites and compute position and time. Notice that the computation is more accurate if GPS antennas can receive signals from at least four satellites.

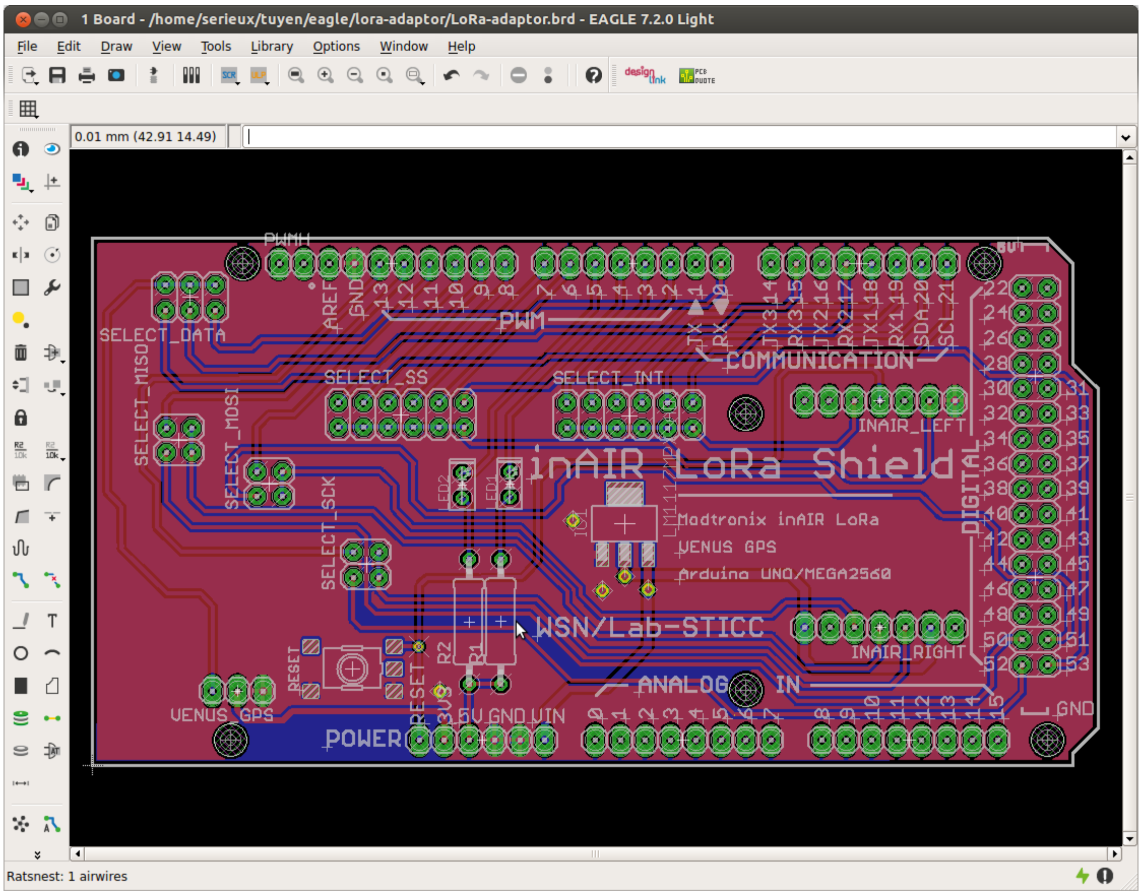
The GPS modules output data in NMEA (National Marine Electronics Association) standard which is formatted in lines of data called sentences [199]. Each sentence contains various data organized. For instance, the format of *GPGGA* sentence is exploited to extract concerning data (geolocation, and time) in our experiments shown in Table E.1.

Table E.1 – GPGGA sentence [199].

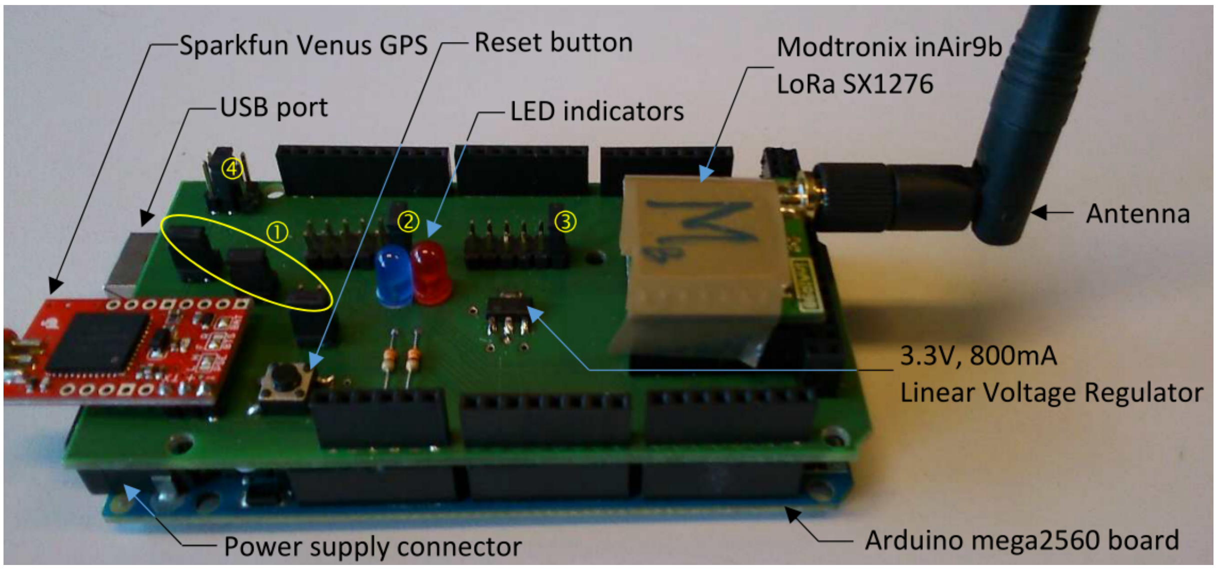
Name	Example	Units	Description
Message ID	$GPGGA		GGA protocol header
UTC Time	161229.487		hhmmss.sss
Latitude	3723.2475		ddmm.mmmm
N/S Indicator	N		N=north or S=south
Longitude	12158.3416		dddmm.mmmm
E/W Indicator	W		E=east or W=west
Position Fix Indicator	1		
Satellites Used	0.7		Range 0 to 12
HDOP	1.0	Horizontal	Horizontal Dilution of Precision
MSL	Altitude	meters	
Units	M	meters	
Geoid	Separation	meters	
Units	M	meters	
Age of Diff. Corr		second	Null fields when DGPS is not used
Diff. Ref. Station ID	0000		
Checksum	*18		
<CR><LF>			End of message termination

E.0.2 Software interactions

A Graphic User Interface is developed to establish the connection between PC and RECoco board via USB port for receiving GPS sentences continuously. The GPS sentences are processed to get information about local time, latitude, longitude, and altitude of the integrated node and then display on PickCell (blues circles). These GPS coordinate values are also stored either into log files or insert into the database at data center directly through 3G/4G connection available on PC.



(a)



(b)

Figure E.1 – (a) Design of RECoco board uses the same pinout of Arduino Mega 2560. (b) A RECoco board composes of main components: Arduino Mega 2560, Modtronix's inAIR9b LoRa SX1276, Sparkfun Venus GPS.

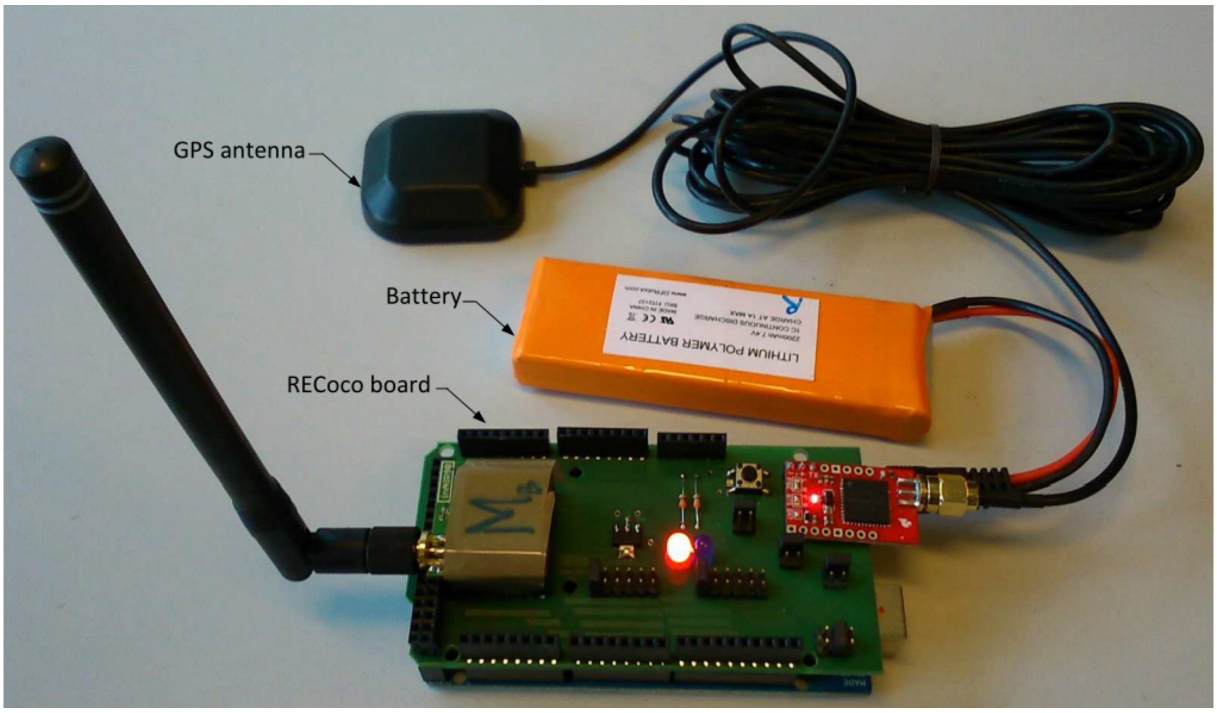


Figure E.2 – A mobile node consists of RECoco board with proof GPS antenna, battery power for receiving geo-location of node from GPS services and then sending to base station.

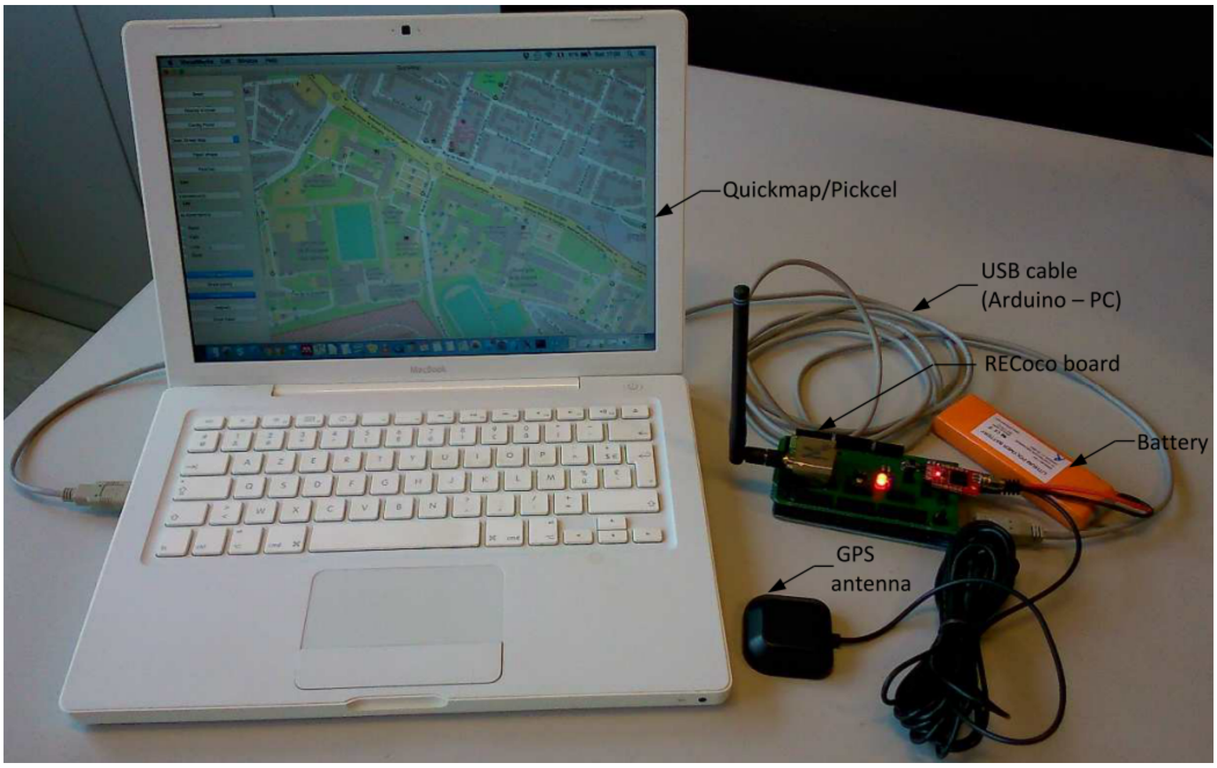
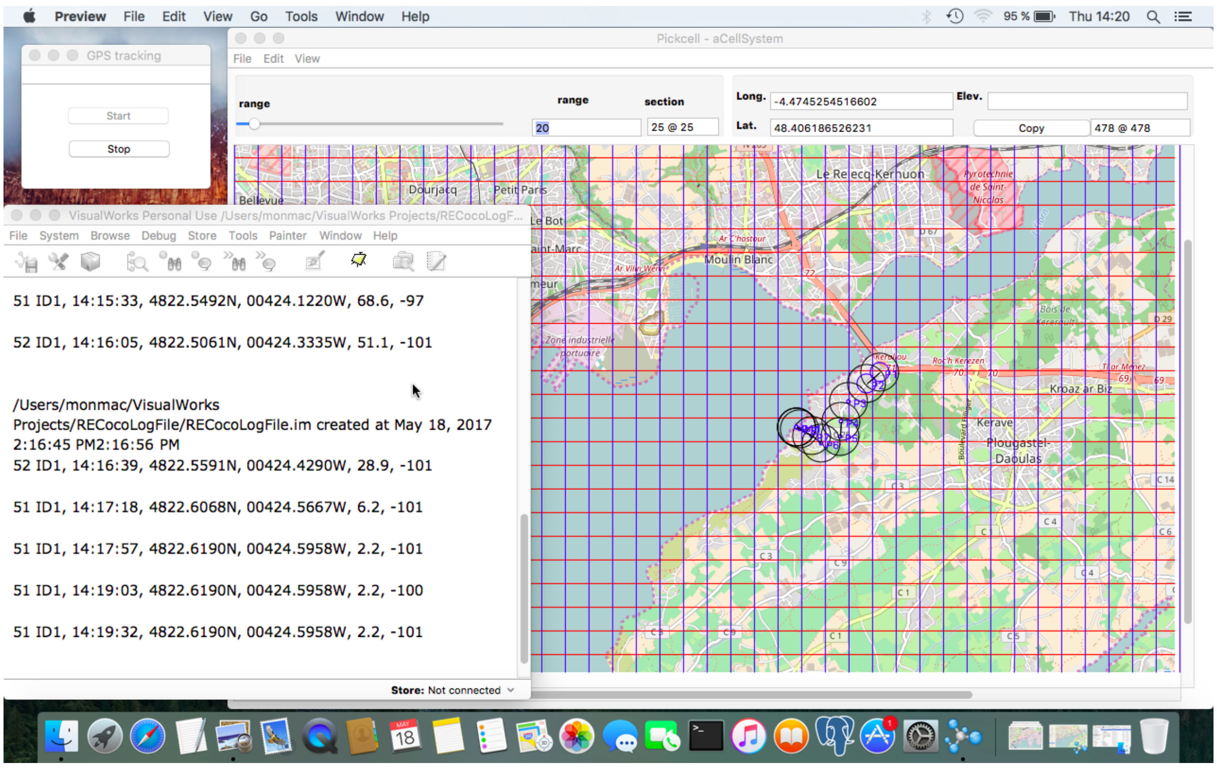


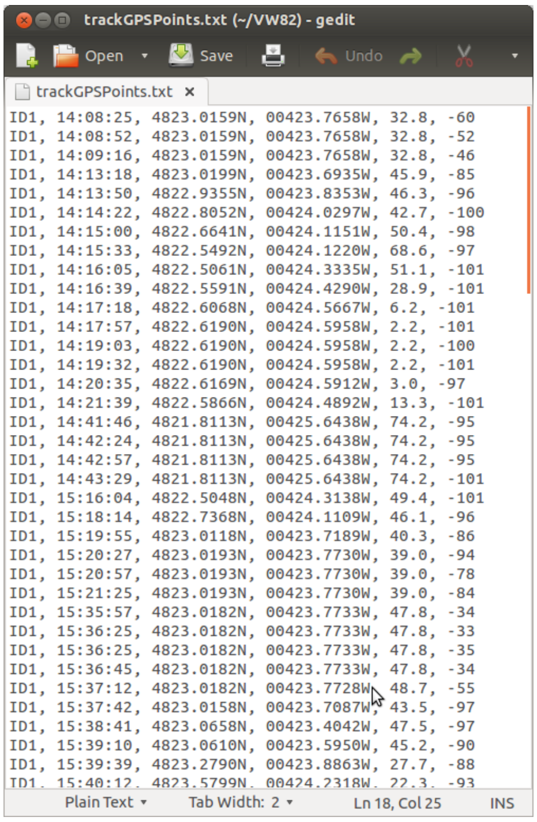
Figure E.3 – Base station for data dissemination: RECoco board transfers collected data to PC running Quickmap/Pickcell via USB port. Data is stored directly into either log files (.log) or PostgreSQL object-relational database.



```
-4.4745254516602
48.406186526231
20
25 @ 25
478 @ 478
```

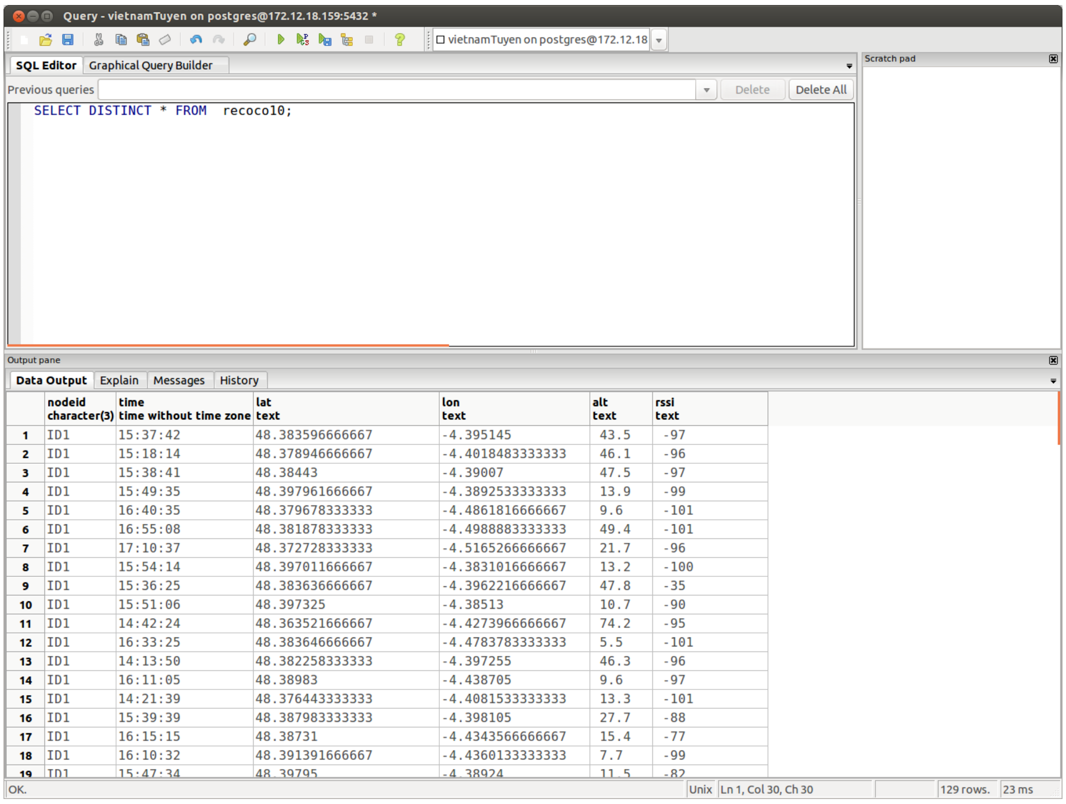
```
51 ID1, 14:15:33, 4822.5492N, 00424.1220W, 68.6, -97

52 ID1, 14:16:05, 4822.5061N, 00424.3335W, 51.1, -101


/Users/monmac/VisualWorks
Projects/RECocoLogFile/RECocoLogFile.im created at May 18, 2017
2:16:45 PM2:16:56 PM
52 ID1, 14:16:39, 4822.5591N, 00424.4290W, 28.9, -101

51 ID1, 14:17:18, 4822.6068N, 00424.5667W, 6.2, -101

51 ID1, 14:17:57, 4822.6190N, 00424.5958W, 2.2, -101

51 ID1, 14:19:03, 4822.6190N, 00424.5958W, 2.2, -100

51 ID1, 14:19:32, 4822.6190N, 00424.5958W, 2.2, -101
```

Figure E.4 – A tool-set consists of three GUIs: GPS tracking, QuickMap/PickCell and status windows. GPS tracking window is employed to handle USB connection with RECoco board. The GPS coordinates of node while moving are depicted on QuickMap/Pickcell. Status window shows the received values and saves them in a log file or inserts into database (see Figure E.5 and E.6.



```
ID1, 14:08:25, 4823.0159N, 00423.7658W, 32.8, -60
ID1, 14:08:52, 4823.0159N, 00423.7658W, 32.8, -52
ID1, 14:09:16, 4823.0159N, 00423.7658W, 32.8, -46
ID1, 14:13:18, 4823.0199N, 00423.6935W, 45.9, -85
ID1, 14:13:50, 4822.9355N, 00423.8353W, 46.3, -96
ID1, 14:14:22, 4822.8052N, 00424.0297W, 42.7, -100
ID1, 14:15:00, 4822.6641N, 00424.1151W, 50.4, -98
ID1, 14:15:33, 4822.5492N, 00424.1220W, 68.6, -97
ID1, 14:16:05, 4822.5061N, 00424.3335W, 51.1, -101
ID1, 14:16:39, 4822.5591N, 00424.4290W, 28.9, -101
ID1, 14:17:18, 4822.6068N, 00424.5667W, 6.2, -101
ID1, 14:17:57, 4822.6190N, 00424.5958W, 2.2, -101
ID1, 14:19:03, 4822.6190N, 00424.5958W, 2.2, -100
ID1, 14:19:32, 4822.6190N, 00424.5958W, 2.2, -101
ID1, 14:20:35, 4822.6169N, 00424.5912W, 3.0, -97
ID1, 14:21:39, 4822.5866N, 00424.4892W, 13.3, -101
ID1, 14:41:46, 4821.8113N, 00425.6438W, 74.2, -95
ID1, 14:42:24, 4821.8113N, 00425.6438W, 74.2, -95
ID1, 14:42:57, 4821.8113N, 00425.6438W, 74.2, -95
ID1, 14:43:29, 4821.8113N, 00425.6438W, 74.2, -101
ID1, 15:16:04, 4822.5048N, 00424.3138W, 49.4, -101
ID1, 15:18:14, 4822.7368N, 00424.1109W, 46.1, -96
ID1, 15:19:55, 4823.0118N, 00423.7189W, 40.3, -86
ID1, 15:20:27, 4823.0193N, 00423.7730W, 39.0, -94
ID1, 15:20:57, 4823.0193N, 00423.7730W, 39.0, -78
ID1, 15:21:25, 4823.0193N, 00423.7730W, 39.0, -84
ID1, 15:35:57, 4823.0182N, 00423.7733W, 47.8, -34
ID1, 15:36:25, 4823.0182N, 00423.7733W, 47.8, -33
ID1, 15:36:25, 4823.0182N, 00423.7733W, 47.8, -35
ID1, 15:36:45, 4823.0182N, 00423.7733W, 47.8, -34
ID1, 15:37:12, 4823.0182N, 00423.7728W, 48.7, -55
ID1, 15:37:42, 4823.0158N, 00423.7087W, 43.5, -97
ID1, 15:38:41, 4823.0658N, 00423.4042W, 47.5, -97
ID1, 15:39:10, 4823.0610N, 00423.5950W, 45.2, -90
ID1, 15:39:39, 4823.2790N, 00423.8863W, 27.7, -88
ID1, 15:40:12, 4823.5799N, 00424.2318W, 22.3, -93
```

Figure E.5 – Structure of the log files to be stored in base station.

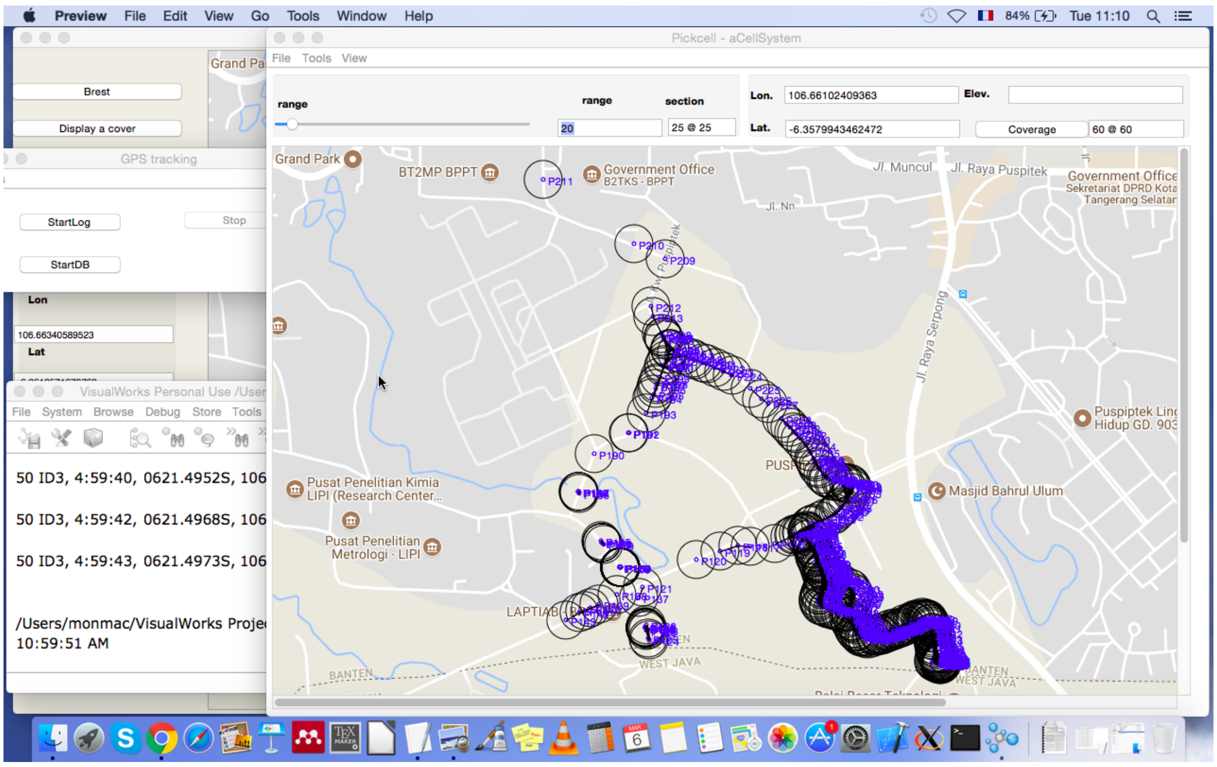


```
SELECT DISTINCT * FROM  recoco10;
```

	nodeid character(3)	time time without time zone	lat text	lon text	alt text	rssi text
1	ID1	15:37:42	48.383596666667	-4.395145	43.5	-97
2	ID1	15:18:14	48.378946666667	-4.4018483333333	46.1	-96
3	ID1	15:38:41	48.38443	-4.39007	47.5	-97
4	ID1	15:49:35	48.397961666667	-4.3892533333333	13.9	-99
5	ID1	16:40:35	48.379678333333	-4.4861816666667	9.6	-101
6	ID1	16:55:08	48.381878333333	-4.4988883333333	49.4	-101
7	ID1	17:10:37	48.372728333333	-4.5165266666667	21.7	-96
8	ID1	15:54:14	48.397011666667	-4.3831016666667	13.2	-100
9	ID1	15:36:25	48.383636666667	-4.3962216666667	47.8	-35
10	ID1	15:51:06	48.397325	-4.38513	10.7	-90
11	ID1	14:42:24	48.363521666667	-4.4273966666667	74.2	-95
12	ID1	16:33:25	48.383646666667	-4.4783783333333	5.5	-101
13	ID1	14:13:50	48.382258333333	-4.397255	46.3	-96
14	ID1	16:11:05	48.38983	-4.438705	9.6	-97
15	ID1	14:21:39	48.376443333333	-4.4081533333333	13.3	-101
16	ID1	15:39:39	48.387983333333	-4.398105	27.7	-88
17	ID1	16:15:15	48.38731	-4.4343566666667	15.4	-77
18	ID1	16:10:32	48.391391666667	-4.4360133333333	7.7	-99
19	ID1	15:47:34	48.39795	-4.38924	11.5	-82

OK. Unix Ln 1, Col 30, Ch 30 129 rows. 23 ms

Figure E.6 – Collected data to be inserted into database via 2G/3G connections.


Preview File Edit View Go Tools Window Help
84% Tue 11:10
Pickcell - aCellSystem
File Tools View
Brest
Display a cover
GPS tracking
StartLog
Stop
StartDB
VisualWorks Personal Use /User
File System Browse Debug Store Tools
Grand Park
BT2MP BPPT
Government Office B2TKS - BPPT
Jl. Muncul
Jl. Raya Puspitek
Government Office Sekretariat DPRD Kota Tangerang Selatan
Jl. Raya Serpong
Puspiptek Lin Hidup GD. 903
Masjid Bahrul Ulum
Pusat Penelitian Kimia LIPI (Research Center...
Pusat Penelitian Metrologi - LIPI
LAPTIAB
BANTEN
WEST JAVA
P211
P210
P209
P212
P193
P192
P190
P121
P120
P187

range 20 section 25 @ 25 Lon. 106.66102409363 Elev. Lat. -6.3579943462472 Coverage 60 @ 60

Lon 106.66340589523 Lat

50 ID3, 4:59:40, 0621.4952S, 106
50 ID3, 4:59:42, 0621.4968S, 106
50 ID3, 4:59:43, 0621.4973S, 106
/Users/monmac/VisualWorks Proje
10:59:51 AM

Figure E.7 – Usage of tools for an experimental measurement at Center for Disaster Risk Reduction Technology (BPPT), Seprong city, Indonesia.

F
Software tools presentation

The central role in the tools is a geographic database supporting OpenStreetMap formats and display (figure F.1). However, all the transformations and code production are built around a core graph model called NetGen. This core model is addressed by cellular systems and wireless sensor networks. Thus, network simulation and cellular automata share the same paradigm grouping graph production, and code generation for a variety of backends.

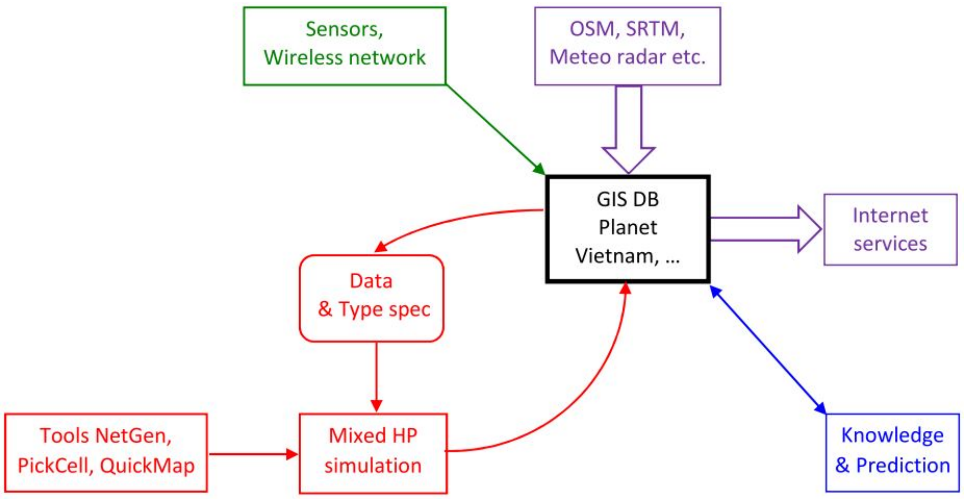


Figure F.1 – Information system architecture around a Postgis database, presenting OpenStreetMap and other inputs (top), web presentation (right), simulation framework (bottom left), and knowledge analysis services (bottom right).

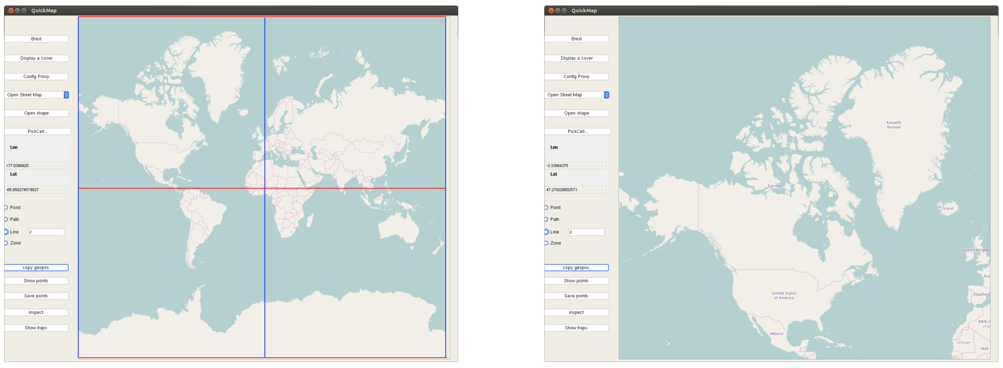
F.0.1 Graphical front-ends: Quickmap and Pickcell

Quickmap Map presentations are of common usage thanks to a technique called *tile rendering*. It consists of hierarchical decomposition following (x,y,z) coordinates, z (zoom) presenting the level of depth in a hierarchy, and the precision of the presentation. Map browsers make use of these parameters to retrieve level of details, x and y being integers representing geographic coordinates. The default presentation is the standard *OpenStreetMap* for maps (see also [200]).

Thus, Quickmap (see figure F.2b, [121] thesis) is a generic navigator to browse this tile organization [200]. It implements 3 dimensions allowing to visit and select geographic zone, plus some capabilities relative to sensor systems and location selection. Figures and present the upper levels of the tile hierarchy.

Pickcell Zones selected in the Quickmap window can be sent to another tool called PickCell. The tool is oriented to the definition of cellular systems and can generate cellular systems regular or irregular organizations, according to grid definition. Another feature is the possibility to select and display network graphs and represent 3D geographic positions.

In addition to the central window, the classification tool and code generation command is used from another window presenting sub-systems and controlling the weaving of cells according to neighborhoods.



(a) (b)

Figure F.2 – (a) The earth map at tile 0 fetched from an OSM tile server [201]. (b) The top left subsection of the earth map at tile 1 fetched from same OSM tile server. Tiles are retrieved from a server using HTTP request in the format `http://MyServer/MyDatabase/z/x/y.png`, z being the depth in a hierarchical organization of tiles [115].

F.0.2 Core tools

NetGen As explained section 2.2.3, simulation of systems is achieved from NetGen, which provides supports to rewrite an internal abstract model into program syntaxes: Occam for concurrent process systems executed on multi-core or distributed architectures, or CUDA for Graphics Processing Unit (GPU) architectures. Both of them follow a synchronous method of execution, making back ends compatible. The generators are suitable for sensor network simulation and physical level simulation, or association of such computations as presented figure A.4.

Netgen can produce a different kind of graphs, as logic graphs for small systems, geometric presentation for computation result or manual specification. There is a simple specification of textual grammar to express node lists and connexions.

For CUDA execution, cells are mapped into a large array and are fitted by channels that describe where to read, and where to write. A NetGen architecture description is fully automatic for both alternatives. However, the programmer still needs to define the cells behavior Cellular Automata as transition rules.

F.0.3 Advantage tools

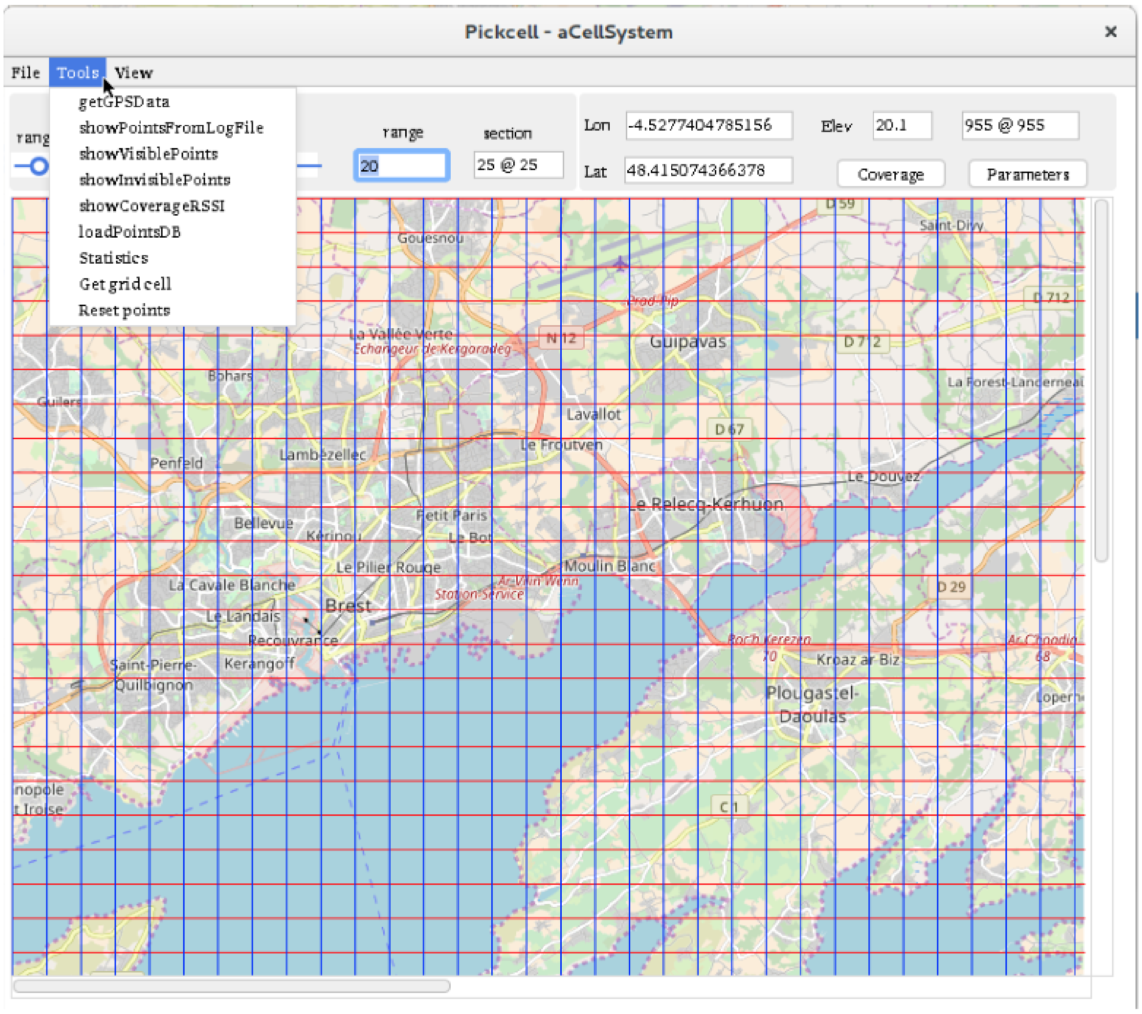


Figure F.3 – Advantage tools is developed in Smalltalk based on Pickcell tool.

Advantage tools are developed for experimental measurements to evaluate the correctness of simulation results.

- *Coverage*: dispatch CUDA program for coverage computation, fetch the computation result and display the coverage on map.
- *Parameters*: optimize the configuration of LoRa nodes.
- *getGPSdata*: control receiving data from a RECoco board, and display the location of nodes on map (Pickcell).
- *showPointsFromLogFile*: retrieve data from a log file and then display the location of nodes on map.
- *loadPointsDB*: retrieve data from the database and then display the location of nodes on map.
- *showVisiblePoints*: show received points which are matched with simulation results.
- *showInvisiblePoints*: show received points which are mismatched with simulation results.
- *Statistics*: do a statistics on a coverage prediction percentage error in comparison to the corresponding experiment.
- *Get grid cell*: produce grid data for computations.
- *Reset points*: clear all points on a current map.

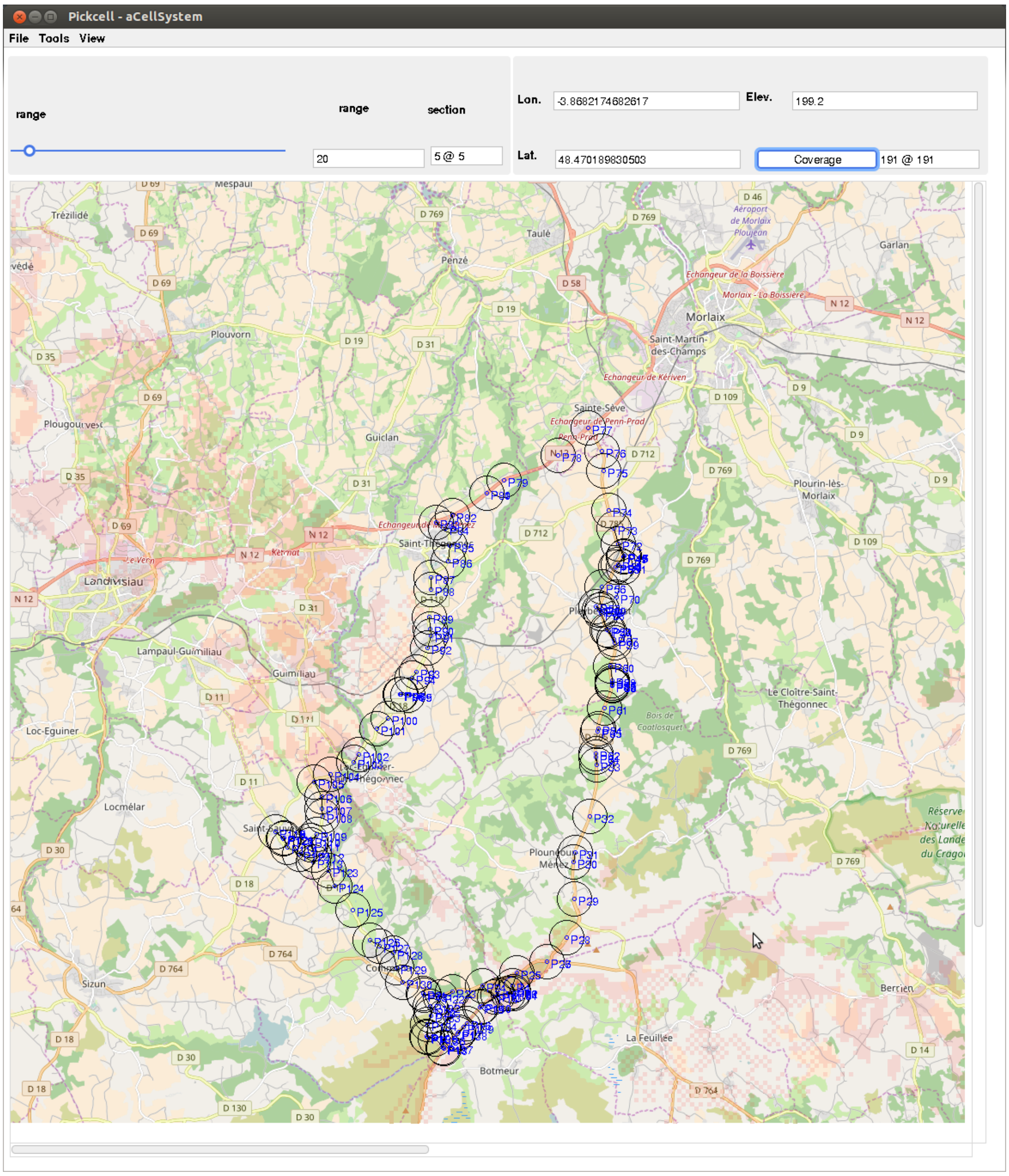


Figure F.4 – GUI of PickCell tool: an experiment at the Roc'h Trevezel mountain, in Brittany, France.

G
Froggy LoRa library

Froggy Factory produces LoRa shield with integrated SX1272 LoRa chip, thereby facilitating implement of LoRa radio transmission with Arduino boards [178, 177].

In this thesis, *Froggy LoRa* library was developed, allowing to establish rapidly long-range wireless communication applications in occam-pi programming language which operate on TVMs stacked on Arduino boards.

In our experiments, the payload of package was structured as
latitute,N/S,longitude,W/E,altitude,hour:minute:second
For example, *4824.0185,N,00430.0687,W,54.2,08:09:10*. The meaning of each part as follows:

824.0185,N: Latitude 48 deg 24.0185' N

00430.0687,W: Longitude 4 deg 30.0687' W

54.2: Altitude (meters) above mean sea level

08:09:10: local zone hour:minute:second

and collected data from sensors, if any.

G.1 SPI library

SPI library provides functions allowing to establish a connection between TVM and LoRa module via SPI interface.

Table G.1 – SPI library.

PROC name	Function
spi.begin	Initialize SPI
spi.end	Disconnect SPI
spi.set.bit.order	Select transmission with LSB/MSB
spi.set.data.mode	Set data mode
spi.set.clock.divider	Set clock rate
spi.transfer	Start data transfer
attatch.interrupt	Enable interrupt
detach.interrupt	Disable interrupt

Table G.2 – LoRa library

PROC name	Function
lora.int	Initialize LoRa
lora.send	Send message
lora.available	Received message available
lora.read	Get message
lora.rf.config	Select RF
lora.freq.channel	Select channel
lora.sf	Set spreading factor
lora.bw	Set bandwidth
lora.cr	Set coding rate
lora.tx.power	Set output power
lora.last.snr	Get SNR value
lora.last.rssi	Get RSSI value

G.2 LoRa library

LoRa library provides functions allowing to configure the operation parameters of LoRa chip and capture the specific registers for QoS management such as RSSI, SNR.

G.3 Network library

Network library provides functions allowing to control the communication.

Code G.1 – Constants

```
VAL BYTE MAX.HOP              IS    4:
VAL INT ROUTING.TABLE.SIZE    IS    4:
VAL INT HEADER.LEN            IS    7:
VAL INT MAX.PAYLOAD.LEN       IS   38:

```

```
VAL BYTE NULL                IS    0:
VAL BYTE INVALID             IS    1:
VAL BYTE DISCOVERING         IS    2:
VAL BYTE VALID               IS    3:

VAL BYTE FLAGS.REQUEST       IS  #10:
VAL BYTE FLAGS.RESPONSE      IS  #20:
VAL BYTE FLAGS.APPLICATION   IS  #40:
VAL BYTE FLAGS.FAILURE       IS  #80:
```

Code G.2 – Data types

```
DATA TYPE headerStructure
  RECORD
    BYTE dest:
    BYTE source:
    BYTE numHops:
    BYTE finalDest:
    BYTE originalSource:
    BYTE id:
    BYTE flags:
:

DATA TYPE payloadStructure IS [MAX.PAYLOAD.LEN]BYTE:

DATA TYPE packet
  RECORD
    headerStructure     header:
    payloadStructure    payload:
:

DATA TYPE routingTableEntry
  RECORD
    BYTE dest:
    BYTE nextHop:
    BYTE state:
:
```

Table G.3 – LoRa network library

PROC name	Function
reset.routing.table	Reset routing table
clear.routing.table	Clear the content
delete.route	Remove a route
delete.oldest.route	Remove the oldest route
delete.route.to	Remove a sending route
retire.oldest.route	Clear the content of oldest route
add.route.to	Add a sending route
get.route.to	Retrieve a sending route
print.routing.table	Print out current routing table
init.header	Initialize a header
init.payload	Initialize a payload
prepare.packet	Form a packet
send.packet	Send a packet
receive.packet	Receive a packet

H
inAir LoRa library

Modtronix produces a series of inAir LoRa modules (e.g. inAir4, inAir9, inAir9B) with integrated SX1276 or SX1278 LoRa chip. These are high quality, and low cost LoRa modules, enabling long-range communications in the 433, 868 and 915 MHz license-free ISM band [176, 193].

inAir LoRa library was also developed in occam-pi programming language aiming at doing experiments on Semtech SX1276 chip which provides better communication features and performances than SX1272 such as output power of 20 dBm, 168 dB maximum link budget[176, 178].

The following tables provide the list of libraries [1] for developing applications with Modtronix inAir LoRa modules. A virtual machine on Arduino [202] allows executing these occam-pi programs for wireless network deployment.

H.0.1 spidriver.module

spidriver library provides functions allowing to configure a SPI interface between TVM and LoRa module.

H.0.2 spilib.module

spilib library provides functions allowing to read/write the values from/to registers of LoRa chip via SPI interface.

[1]Note that it is quite easy to define a function with a name in a PROC expression.

Table H.1 – SPI driver for setting SPI connections.

PROC name	Note
spi.begin	
spi.end	
spi.set.bit.order	
spi.set.data.mode	
spi.set.clock.divider	
spi.transfer	
attatch.interrupt	
detach.interrupt	

Table H.2 – SPI library for read/write via SPI.

PROC name	Note
spi.init	
spi.write	
spi.brust.write	
spi.read	
spi.brust.read	

H.0.3 inaircad.module

Packet structures for inAirCAD library (equivalent to Physical layer of The OSI model layers): ***toAddress, fromAddress, id, flags***

H.0.4 networkcad.module

Packet structures for networkCAD (equivalent to Data Link of The OSI model layers): ***dest, source, originalSource, numHops, id, flags***

Table H.3 – inAirCAD library for setting communication parameters.

PROC name	Note
inair.set.mode.idle	
inair.set.mode.sleep	
inair.set.mode.rx	
inair.set.mode.tx	
inair.set.preamle.length	
inair.set.modem.config	
inair.set.tx.power	
inair.validate.rx.buf	
inair.available	
inair.wait.available	
inair.wait.available.timeout	
inair.wait.packet.sent	
inair.is.channel.active	
inair.wait.cad	Channel Activity Detection (CAD)
inair.set.promiscuous	
inair.set.this.address	
inair.set.header.to	
inair.set.header.from	
inair.set.header.id	
inair.set.header.flags	
inair.set.mode	
inair.send	
inair.clear.rx.buf	
inair.recv	
inair.handle.interrupt	
inair.init	
inair.receive	

Table H.4 – inAir nework CAD library for establishing communications.

PROC name	Note
set.header.to	
set.header.from	
set.header.id	
set.max.hops	
clear.routing.table	
reset.routing.table	
get.route.to	
delete.route.to	
delete.oldest.route	
delete.route.to	
add.route.to	
retire.oldest.route	
printing.routing.table	
send	
send.ack	
receive	
receive.ack	
do.arp	Address Resolution Protocol
inair.set.header.id	
inair.set.header.flags	
inair.set.mode	
inair.send	
inair.clear.rx.buf	
inair.recv	
inair.handle.interrupt	
inair.init	
inair.receive	

I
A LoRa star network experiment: package structure and occam-pi programs

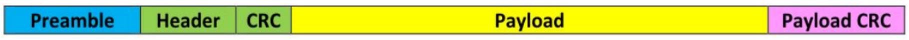
I.1 Package structure for LoRa communications

The format of package in LoRa data link layer composes five parts as following [159]:

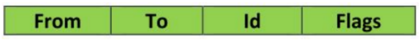
- *Preamble* in default setting is 12 symbols (corresponding to value 8 in configuration).
- *Header* is able to choose either implicit or explicit. Explicit header has 4 bytes with 2 bytes CRC (noted that header CR is always 4/8).
- *Payload* contents user's data. The maximum size of payload is is 255 bytes with payload CRC on (CR = 4/5, 4/6, 4/7, 4/8).
- *From*: address of sender, [1]
- *To*: destination address,
- *Id*: a unique number generated by sender to identify packet (could be used to calculate Packet Error Rate , PER)
- *Flags*: status bits which may be used at PHY layer.

In our experiment, the payload is constructed to track the location of mobile sensor nodes while moving. A typical payload contains several fields, including unique identity, local time, geo-location.

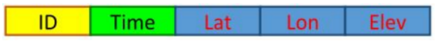
[1]Unique identifications of node in network are usually used as addresses in communications.



(a) A typical package provided by LoRa protocol.



(b) The structure of an explicit header.



(c) Payload format being used in our experiments.

Figure I.1 – Package format to be used in our experiments.

- *ID*: unique identification of nodes in the network being used by applications.
- *Time*: Local time of node in format hour:minute:second.
- *Lat, Lon, Elev*: geographic values of place where a sensor node is located.[2]
- The last part is CRC (2 bytes) for payload.

After receiving a message from a mobile node and checking the consistency of package format, if the sensing message is collected successfully its value is analysis. The structure of a received message as presented in Figure I.2. At the base station, when receiving a message from mobiles, for received a message the corresponding strength of signal (RSSI) values are also measured.

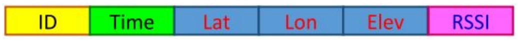


Figure I.2 – Formation of the received message.

Received messages of several nodes with different were stored in a log file or inserted directly into database server for later use. The messages also processed by a user's interface to show the node's position and sensing values, if any, on the map in real time. This is a visual interface allowing to observe the operation of the experimental network.

I.2 Occam programs of a sink nodes and sensor nodes

Code I.1 – Occam-pi program running on a sink node

```
#INCLUDE "network.module"
#INCLUDE "sensing.module"

PROC BSNode ()
  VAL INT numNeighbor IS 2:
  VAL BYTE thisAddress IS 1:
  BYTE avail:
```

[2]Time, Latitude, Longitude and Elevation values are obtained from integrated GPS module at each node.

```
BYTE c:
headerFormat     header:
payloadFormat   payload:
[HEADER.LEN + MAX.PAYLOAD.LEN]BYTE buffer:
[1]BYTE character:
INT rfIndex:
BYTE idNeighbor:
packet pack:
routingTableEntry route:
[ROUTING.TABLE.SIZE]routingTableEntry routingTable:
SEQ
-- initilizing RF
  lora.init ()
  lora.rf.config (4)
  lora.freq.channel (3)
  -- always perform serial.start after lora.init
  serial.start (TX0, 9600)
  -- initilizing packet (null)
  init.header (header, thisAddress)
  init.payload (payload)
  reset.routing.table (routingTable)
  print.routing.table(routingTable)
  -- mainloop
  WHILE TRUE
    SEQ
      idNeighbor := 2
      SEQ i=0 FOR numNeighbor
        SEQ
          header[dest] := idNeighbor
          header[finalDest] := idNeighbor
          header[source] := thisAddress
          header[originalSource] := thisAddress
          -- sensing process
          collect.data (payload)
          prepare.packet (pack, header, payload)
          send (pack)
          lora.available (avail)
          WHILE (avail = 0)
            SEQ
              delay(1)
              lora.available (avail)
          serial.write.newline (TX0)
          serial.write.dec.int (TX0, INT avail)
          serial.write.string (TX0, " bytes received: "
            )
```

```
-- read bytes  one by one
SEQ j=0 FOR INT avail
  SEQ
    lora.read (c)
    buffer[j] := c
IF
  (buffer[3] = thisAddress)
    SEQ
      SEQ k=0 FOR ((INT avail) - HEADER.LEN)
        SEQ
          character[0] := buffer[k + HEADER.LEN]
          serial.write.string (TX0, character)
      -- receive and print SNR
      serial.write.string (TX0, "*nSNR: ")
      lora.last.snr (rfIndex)
      serial.write.dec.int (TX0, rfIndex)

      -- receive and print RSSI
      serial.write.string (TX0, ", RSSI: ")
      lora.last.rssi (rfIndex)
      serial.write.dec.int (TX0, rfIndex)
      add.route.to (routingTable, buffer[4], buffer[1], VALID)
  (buffer[2] < MAX.HOP) -- forward packet
    SEQ
      get.route.to (routingTable, buffer[3], route)
      IF
        (route[state] = VALID)
          SEQ
            header[dest]:= route[nextHop]
            header[source]:= thisAddress
            header[numHops] := buffer[2] + 1
            header[finalDest] := buffer[3]
            header[originalSource] := buffer[4]
            header[id] := buffer[5]
            header[flags] := buffer[6]
        TRUE
          serial.write.string(TX0, "*nCannot find the path to delivery packet")
  (buffer[2] = MAX.HOP)
```

```
                serial.write.string(TX0, "*nReach to
                  maximum hops")
              TRUE
                SKIP
            print.routing.table(routingTable)
            idNeighbor := idNeighbor + 1
            delay (5000)
:
```

Code I.2 – Occam-pi program running on a sensor node

```
#INCLUDE "network.module"
#INCLUDE "sensing.module"

PROC Node ()
  VAL BYTE thisAddress IS 2:
  BYTE avail:
  BYTE c:
  headerFormat   header:
  payloadFormat payload:
  [HEADER.LEN + MAX.PAYLOAD.LEN]BYTE buffer:
  [1]BYTE character:
  INT rfIndex:
  packet pack:
  [ROUTING.TABLE.SIZE]routingTableEntry routingTable:
  SEQ
    lora.init ()
    lora.rf.config (4)
    lora.freq.channel (3)
    -- always perform serial.start after lora.init
    serial.start (TX0, 9600)
    init.header (header, thisAddress)
    reset.routing.table (routingTable)
    print.routing.table(routingTable)
    WHILE TRUE
      SEQ
        lora.available (avail)
        IF
          avail <> 0
            SEQ
              SEQ i=0 FOR INT avail
                SEQ
                  lora.read (c)
                  buffer[i] := c
              IF
                (buffer[0]=thisAddress)
```

```
                  SEQ
                    serial.write.newline (TX0)
                    SEQ j=0 FOR ((INT avail) - HEADER.LEN)
                      SEQ
                        character[0] := buffer[j + HEADER.LEN]
                        serial.write.string (TX0, character)
                    -- receive and print SNR
                    serial.write.string (TX0, "*nSNR: ")
                    lora.last.snr (rfIndex)
                    serial.write.dec.int (TX0, rfIndex)
                    -- receive and print RSSI
                    serial.write.string (TX0, ", RSSI: ")
                    lora.last.rssi (rfIndex)
                    serial.write.dec.int (TX0, rfIndex)
                    add.route.to (routingTable, buffer[4], buffer[1], VALID)
                    print.routing.table(routingTable)
                    -- <dest, source, numHops, finalDest, originalSource, id, flags><payload>
                    header[dest] := buffer[1]
                    header[finalDest] := buffer[4]
                    collect.data (payload)
                    prepare.packet (pack, header, payload)
                    send (pack)
                TRUE
                  SKIP
          TRUE
            SKIP
:
```

J Experiments on reconfigurable hardware for distributed remote sensing systems and LoRa modulation features

J.1 Dynamic configuration

J.1.1 Reconfigurable hardware platform in wireless sensor networks

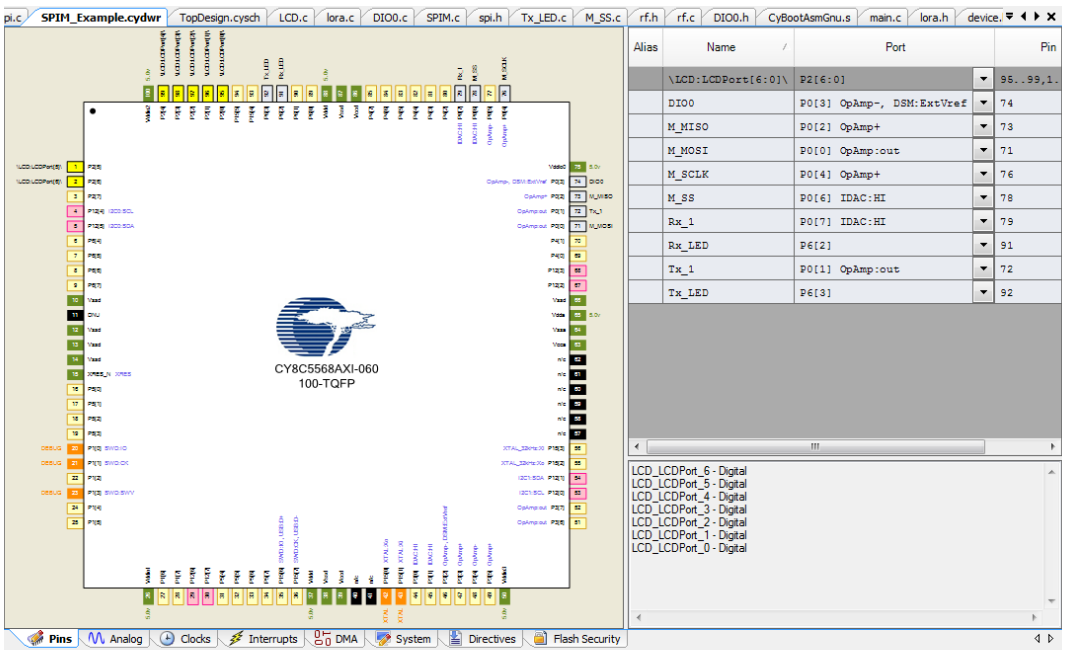
Nowadays sensor networks are used worldwide in the automatic surveillance of environmental systems. The key issues concerned in designing such networks are scalability, and low energy consumption to self-adapt with the number and various types of sensors, improving the lifetime of the battery-powered system respectively.

A promising way to fulfill these requirements is to utilize Programmable System on Chip (PSoC) built on ultra-low-power process technology, providing high performance, especially dynamic reconfiguration system. This dynamic reconfiguration enables to make changes to the hardware resources, namely universal digital, analog, mixed-signal and radio frequency blocks, to perform the functions of peripheral components [203].

Each of configurations is achieved by attaching and/or detaching blocks in software without any downtime during run-time. It is by adopting this approach not only the configurations can be reloaded at any time but also multiple configurations can be active simultaneously. This allows optimizing network with respect to performance as well as energy saving. Apart from mentioned advantages, dynamic reconfiguration mechanism the also provides self-healing capability by reorganizing themselves as some nodes is lost in order to improve the robustness and efficiency of deployed networks [204, 205].

A case study on Cypress PSoC

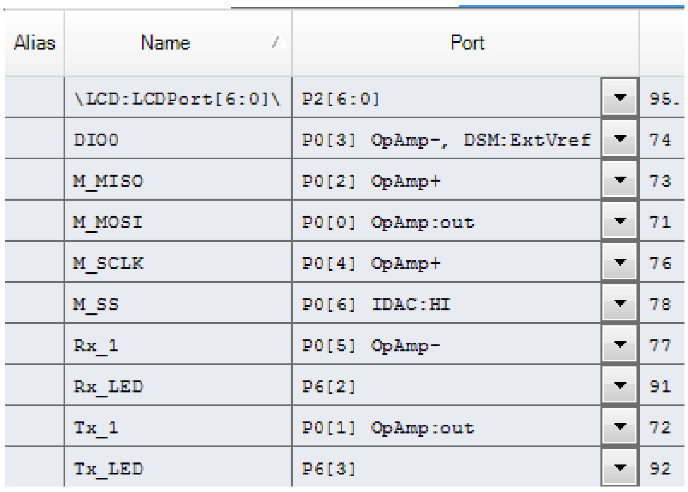
Cypress company provides a lot of tools for developing conveniently a dynamic system on their PSoC products. As an example, we implemented a dynamic system based on dynamic configuration on Cypress PSoC for sending and receiving through LoRa links.



Alias	Name	Port	Pin
	\LCD:LCDPort[6:0]\	P2[6:0]	95..99,1.
	DIO0	P0[3] OpAmp-, DSM:ExtVref	74
	M_MISO	P0[2] OpAmp+	73
	M_MOSI	P0[0] OpAmp:out	71
	M_SCLK	P0[4] OpAmp+	76
	M_SS	P0[6] IDAC:HI	78
	Rx_1	P0[7] IDAC:HI	79
	Rx_LED	P6[2]	91
	Tx_1	P0[1] OpAmp:out	72
	Tx_LED	P6[3]	92

LCD_LCDPort_6 - Digital
LCD_LCDPort_5 - Digital
LCD_LCDPort_4 - Digital
LCD_LCDPort_3 - Digital
LCD_LCDPort_2 - Digital
LCD_LCDPort_1 - Digital
LCD_LCDPort_0 - Digital

Figure J.1 – A dynamic system was designed on PSoC Creator, then upload into CY8CKI-001 PSoC Development Kit.



Alias	Name	Port	
	\LCD:LCDPort[6:0]\	P2[6:0]	95.
	DIO0	P0[3] OpAmp-, DSM:ExtVref	74
	M_MISO	P0[2] OpAmp+	73
	M_MOSI	P0[0] OpAmp:out	71
	M_SCLK	P0[4] OpAmp+	76
	M_SS	P0[6] IDAC:HI	78
	Rx_1	P0[5] OpAmp-	77
	Rx_LED	P6[2]	91
	Tx_1	P0[1] OpAmp:out	72
	Tx_LED	P6[3]	92

Figure J.2 – Pin mapping of PSoC designed system for testing point-to-point connection using LoRa inAir module via SPI bus.

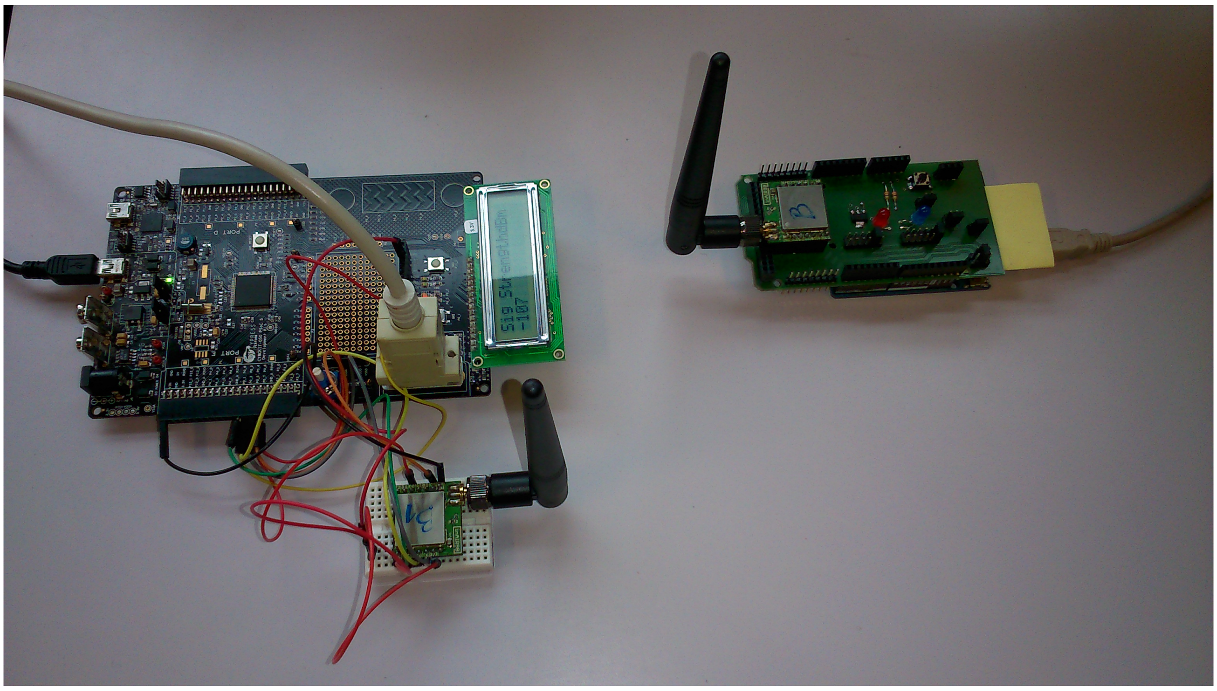


Figure J.3 – Radio signal strength (RSSI) value is displayed on LCD at PSoC site in an experiment.

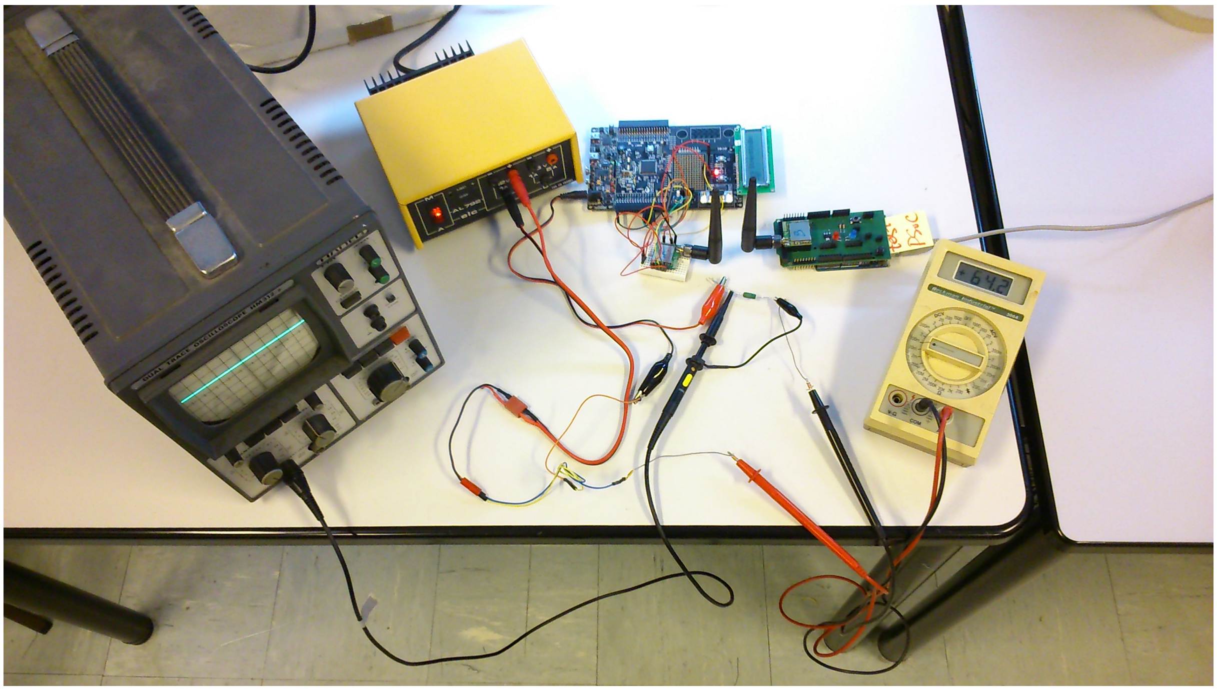


Figure J.4 – Power consumption experiments.

Table J.1 – System design description.

Component	Function	Description
ARM Cortex-M3	Processor	67 MHz, 64 KB SRAM, 256 KB Flash program memory
SPIM	Provide SPI communication	Master device
LCD	Display status of system	46x16 segments, mapping on pins P2[6:0]
LEDs	Display status of transceiving	Mapping on pin P6_3 and P6_2
DIO0	Reset system	Mapping on pin P0_3

Table J.2 – Energy consumption: PSoC3 vs Arduino UNO. We captured 10 values of current and voltage and then compute the mean value of them. Note that the size of package in these measurements is 48 bytes (both antennas 1 dBi, output power 20 dBm, preamble = 8 (12 bytes), explicit header with CRC, Fc=868 MHz, Bw=125 KHz, CR=4/8, SF=10).

	PSoC3			Arduino UNO	
Value	unattached SPI block	Receiving	Sending	Receiving	Sending
Current (mA)	63.4	64.5	72.6	89.4	94.2
Voltage (mV)	64.2	64.1	72.3	90.2	97.1

J.2 LoRa spectrum observation with FUNCube Dongle Pro+

The main features of LoRa technology were introduced in Section 4.1. In order to experience insight into how LoRa modulation operates, we did several experimental measurements to capture and analysis LoRa modulated signal using FUNcube Dongle Pro+. This module is a delicate design for LF to L band software-defined radio, offering to receive messages from LEO satellites [206]. A set of software on Ubuntu including Qthid, Quisk, and gqrx SDR allows controlling the connection with the FUNcube module via USB port for capturing, observing and recording LoRa modulated signal [207, 208, 209]. With salient advantages such as ultra-low power, long-range communications being up to hundred kilometers, LoRa technology is a promising technology for environmental remote sensing from low orbit space station, especially isolated areas where lack of telecommunication infrastructure.

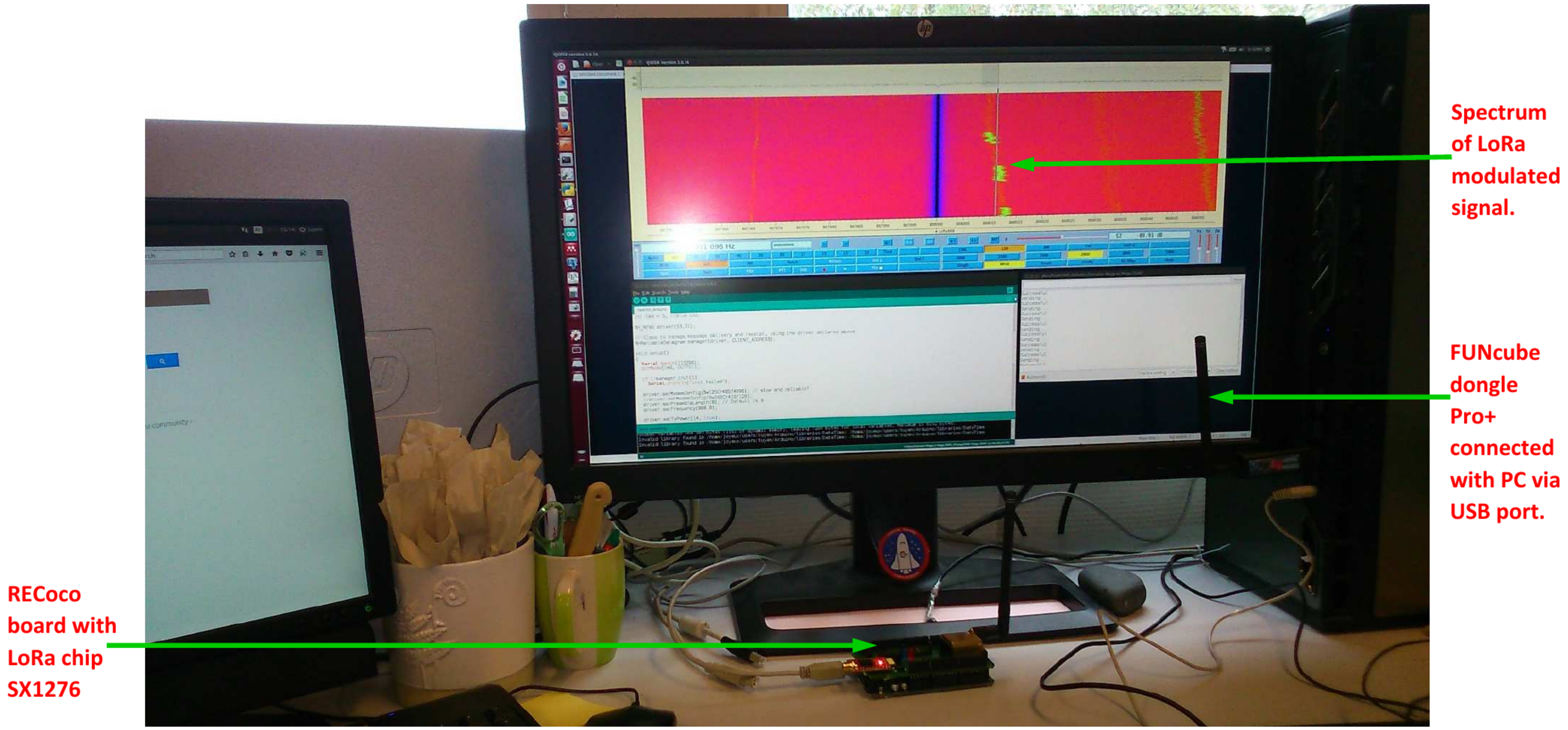

Spectrum of LoRa modulated signal.
FUNcube dongle Pro+ connected with PC via USB port.
RECoco board with LoRa chip SX1276

Figure J.5 – Experiment on spectrum of LoRa modulated signals.

Publications during Ph.D.

Book chapters:

[4] Lucas PY., Van Long N.H., **Truong T.P.**, Pottier B. (2015). *"Wireless Sensor Networks and Satellite Simulation"*. In: Pillai P., Hu Y., Otung I., Giambene G. (eds) Wireless and Satellite Systems. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 154. Springer, ISBN 978-3-319-25478-4.

[5] **Truong T.P.**, Van Tran H., Nguyen K.T., Huynh H.X., Pottier B. (2016). *"Optimizing the Connection Time for LEO Satellite Based on Dynamic Sensor Field"*, Vinh P., Alagar V. (eds) Context-Aware Systems and Applications. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 165. Springer, ISBN 978-3-319-29235-9.

[210] Van Tran H., **Truong T.P.**, Nguyen K.T., Huynh H.X., Pottier B. (2016). *"A Federated Approach for Simulations in Cyber-Physical Systems"*. Vinh P., Alagar V. (eds) Context-Aware Systems and Applications. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 165. Springer, ISBN 978-3-319-29235-9.

[6] **Truong T.P.**, Luong H.H., Huynh H.H., Huynh H.X., Pottier B. (2016). *"Modeling and Optimizing of Connections for Dynamic Sensor Fields Based on BT-Graph"*. In: Vinh P., Barolli L. (eds) Nature of Computation and Communication. ICTCC 2016. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 168. Springer, ISBN 978-3-319-46908-9.

[211] Luong H.H., **Truong T.P.**, Nguyen K.M., Lam B.H., Huynh H.X. (2016). *"Optimizing the Light Trap Position for Brown Planthopper (BPH) Surveillance Network"*. In: Vinh P., Barolli L. (eds) Nature of Computation and Communication. ICTCC 2016. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 168. Springer, ISBN 978-3-319-46908-9.

[212] Lam B.H., **Truong T.P.**, Nguyen K.M., Huynh H.X., Pottier B. (2016). *"An Hierarchical Scheduled Algorithm for Data Dissemination in a Brown Planthopper Surveillance Network"*. In: Vinh P., Barolli L. (eds) Nature of Computation and Communication. ICTCC 2016. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 168. Springer, ISBN 978-3-319-46908-9.

International journal articles:

[7] **Tuyen Phong Truong**, Hiep Xuan Huynh, Huong Hoang Luong, Vinh Cong Phan, Bernard Pottier. 2016. *"Modeling The Connections of Dynamic Sensor Based on BT-Graph"*. EAI Endorsed Transactions on Context-aware Systems and Applications. Volume 3, Issue 8.

[2] **Tuyen Phong Truong**, Hiep Xuan Huynh, Bernard Pottier. 2016. *"Parallel Cellular Automata Based Simulation of Radio Signal Propagation"*. International Journal of Computer Science and Information Security. Volume 14 No. 10. pp. 467-472. Emerging Sources Citation Index (ESCI).

[3] Onil Nazra Persada Goubier, Hiep Xuan Huynh, **Tuyen Phong Truong**, Mahamadou Traoré. 2017. *"WSN based Monitoring, Cellular Modeling and Simulations for Environment"*. ASM Science Journal. Special Issue 2017(1) ICT-Bio. 56-63. From International Workshop BIOICT2016, 30-31 May, 2016, Kuala Lumpur, Malaysia.

[1] **Tuyen Phong Truong**, Bernard Pottier, and Hiep Xuan Huynh. *Cellular Simulation for Distributed Sensing over Complex Terrains"* (30 pages). Sensors (MDPI). Special Issue: Dependable Monitoring in Wireless Sensor Networks. Sensors 2018, 18, 2323. Available online: `http://www.mdpi.com/1424-8220/18/7/2323`

International conference papers with selection committee:

[162] **Tuyen Phong Truong**, Hiep Xuan Huynh, Bernard Pottier. 2016. *"Monitoring Of Environment: A High Performance Method for Radio Coverage Exploration"*. IEEE Radio 2016, Hotel Le Récif, Saint-Gilles Les Bains Réunion Island, 10-13 October 2016. IEEE Xplore. DOI: 10.1109/RADIO.2016.7772031

International workshops:

[213] **Tuyen Phong Truong**, Hiep Xuan Huynh, Bernard Pottier. *"A simulation of radio propagation based on cellular automata"*. International workshop RESSACS, 9-13 May 2016, IRD, Bondy, France.

[103] **Tuyen Phong Truong**, Hiep Xuan Huynh, Bernard Pottier. *"Sensing and Simulation: Cellular Methods and Tools"*. Journée Thématique Capteurs Magnétiques - GDR CNRS Ondes. May 25-27, 2016, Montpellier, France.

[192] **Tuyen Phong Truong**, Hiep Xuan Huynh, Bernard Pottier. 2016. *"Wireless Network on Coastal Topologies: Parallel Simulation of Radio Coverage on Cellular Systems"*. A Connected Ocean (ACO): new approaches, new technologies, new challenges for knowledge of ocean processes, 11-13 Oct 2016, Brest, France.

[102] **Tuyen Phong Truong**, Hiep Xuan Huynh, Bernard Pottier. *"Wireless Sensor Network: Long-range Radio Coverage for Complex Terrain Areas"*. Journées Thématiques (GDR CNRS Ondes) "Capteurs Magnétiques & Électromagnétiques (EM) et Applications". 31 May - 01 Jun 2018, Grenoble, France.

[214] **Tuyen Phong Truong**, Udrekh, Bernard Pottier, SAMES Group. *"Long-range Communications: A Promising Technology for Environmental Monitoring Applications in Developing Countries"*. The 2018 Joint Working Group France-Indonesia Cooperation in Higher Education, Research, Innovation and Entrepreneurship (JWG2018). 26-28 Jun 2018, Poitiers, France. Organized by French Embassy in Indonesia.

Visits and public experiments in SAMES project framework:

1. Contribution to the training "Cellular Automata, Multi Agent and WSN", CIRELA Association, for Jakarta BPPT in Paris, France, 26-8 October 2016 with topic "Long-range Communications in Wireless Sensor Networks".
2. Visit to Center for Disaster Risk Reduction Technology (BPPT), Seprong, Indonesia, from 4 to 7 March 2018 with the objective of experimental measurements on long-range (LoRa) communication technology, Indonesia context.
   (`http://wsn.univ-brest.fr/tuyen/visitingBPPT.pdf`)
3. Demonstrations for Le Télégramme newspaper, published as "Inondations. L'UBO peut les prévoir" in June 6, 2018 (D. Deniel). Article about flash flooding diagnostic in Morlaix, June 3, 2018.
   (`http://wsn.univ-brest.fr/tuyen/Inondation_Morlaix.pdf`)
4. Technical reports about experimental measurements on RECoco LoRa system, City of Brest, Bay of Brest, Arrée Mountains.
   - for simulator validation, with Vincent Rodin, Bernard Pottier, Jean-François Dorville, Edwin Ogodo in June, 2017.
     (`http://wsn.univ-brest.fr/tuyen/WSNs.pdf`)
   - during visit of Prof. Moussa Kerkar (University of Béjaia), Brest harbor in 2-3 December, 2017.
     (`http://wsn.univ-brest.fr/tuyen/Pre-etude-Soummam.pdf`)
5. Diagnostics about Maria storm and flooding, La Guadeloupe, joint work with Jean-François Dorville.
   (`http://wsn.univ-brest.fr/tuyen/Guadeloupe.mp4`)
6. Evaluation and movie of real time radio coverage computation given a fast aerial mobile.
   (`http://wsn.univ-brest.fr/tuyen/MobiGateway.mp4`)

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

UBO Université de Bretagne Occidentale

Titre : Simulation et Support du Compilateur pour la Communication et la Mobilité pour la Surveillance de l'environnement

Mots clés : Automates cellulaires; Terrain complexe; LoRa; Simulation physique parallèle; Propagation de signaux radio

Résumé : Les transmissions radio à longue portée et basse énergie ouvrent de nouveaux champs d'application pour les capteurs, en particulier pour la surveillance de l'environnement. Le protocole radio LoRa permet, par exemple, de connecter des capteurs à une distance pouvant aller jusqu'à dix kilomètres en ligne de visée. Cependant, la grande surface couverte amène plusieurs difficultés, telles que le placement spatial en regard de la topologie géographique, ou la variabilité de la latence des communications. Le positionnement dans l'environnement comporte également des contraintes liées à l'intérêt des points de mesure du phénomène physique. Les critères de conception de ces réseaux tranchent donc avec les méthodes existantes (disques) quand on s'attaque aux terrains complexes. Cette thèse décrit des techniques de simulation basées sur l'analyse géographique cellulaire pour calculer les couvertures radio à longue portée et déduire les caractéristiques radios dans ces situations. Comme la propagation radio n'est qu'un cas particulier de phénomènes physiques, on montre qu'une approche unifiée cellulaire permet de caractériser beaucoup de comportements physiques potentiels. Le cas des fortes pluies et des inondations est étudié. L'analyse de la géographie est réalisée en utilisant des outils de segmentation pour produire des systèmes cellulaires qui sont à leur tour traduits en code pour des calculs de haute performance. La thèse fournit des résultats d'expériences de terrain complexes pratiques en utilisant LoRa, permettant de qualifier l'exactitude de la simulation des couvertures, et les caractéristiques d'ordonnancement des communications. Nous produisons des tables de performance pour les simulations sur les unités de traitement graphique (GPUs) qui montrent que le choix d'une algorithmique parallèle est pertinent sur ces problèmes.

Title : Simulation and Compiler Support for Communication and Mobility for Environment Sensing

Keywords : Cellular automata; Complex terrain; LoRa; Parallel physical simulation; Radio signal propagation

Abstract : Long-range radio transmissions open new sensor application fields, in particular for environment monitoring. For example, the LoRa radio protocol enables to connect remote sensors at distance as long as ten kilometers in a line-of-sight. However, the large area covered also brings several difficulties, such as the placement of sensing devices in regard to topology in geography, or the variability of communication latency. Sensing the environment also carries constraints related to the interest of sensing points in relation with a physical phenomenon. Thus criteria for designs are evolving a lot from the existing methods, especially in complex terrains. This thesis describes simulation techniques based on geography analysis to compute long-range radio coverages and radio characteristics in these situations. As radio propagation is just a particular case of physical phenomena, it is shown how a unified approach also allows to characterize the behavior of potential physical risks. The case of heavy rainfall and flooding is investigated. Geography analysis is achieved using segmentation tools to produce cellular systems which are in turn translated into code for high-performance computations. The thesis provides results from practical complex terrain experiments using LoRa which confirm the accuracy of the simulation, and scheduling characteristics for sample networks. Performance tables are produced for these simulations on current Graphics Processing Units (GPUs).