

ECOLE POLYTECHNIQUE DE L'UNIVERSITÉ FRANÇOIS RABELAIS DE TOURS
Département Informatique
64 avenue Jean Portalis
37200 Tours, France
Tél. +33 (0)2 47 36 14 14
polytech.univ-tours.fr

Projet Recherche & Développement
2018-2019

Développement Algorithmique et Conception d'un Simulateur pour un Réseau de Capteurs dynamiques

Tuteur académique
Ameur SOUKHAL

Étudiant
Thibaud BEAUFILS (DI5)

4 avril 2019



Liste des intervenants

Nom	Email	Qualité
Thibaud BEAUFILS	thibaud.beaufils@univ-tours.fr	Étudiant DI5
Ameur SOUKHAL	ameur.soukhal@univ-tours.fr	Tuteur académique, Département Informatique



Avertissement

Ce document a été rédigé par Thibaud Beaufiles susnommé l'auteur.

L'Ecole Polytechnique de l'Université François Rabelais de Tours est représentée par Ameer Soukhal susnommé le tuteur académique.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

L'auteur reconnaît assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

L'auteur atteste que les propos du document sont sincères et assument l'entière responsabilité de la véracité des propos.

L'auteur atteste ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

L'auteur atteste que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

L'auteur reconnaît qu'il ne peut diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable du tuteur académique et de l'entreprise.

L'auteur autorise l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.



Pour citer ce document

Thibaud Beaufils, *Développement Algorithmique et Conception d'un Simulateur pour un Réseau de Capteurs dynamiques*: , Projet Recherche & Développement, Ecole Polytechnique de l'Université François Rabelais de Tours, Tours, France, 2018-2019.

```
@mastersthesis{
  author={Beaufils, Thibaud},
  title={Développement Algorithmique et Conception d'un Simulateur pour un Réseau de
    Capteurs dynamiques: },
  type={Projet Recherche \& Développement},
  school={Ecole Polytechnique de l'Université François Rabelais de Tours},
  address={Tours, France},
  year={2018-2019}
}
```

Table des matières

Liste des intervenants	a
Avertissement	b
Pour citer ce document	c
Table des matières	i
Table des figures	v
Introduction	1
1 Contexte	2
2 Hypothèses	4
3 Objectifs	5
4 Base méthodologique	7
1 Organisation du projet	7
2 Conventions de développement.....	8
I Description Générale	9
5 Environnement du projet	10
6 Caractéristiques des utilisateurs	11
7 Fonctionnalités du système	12

8	Structure générale du système	13
1	Le modèle.....	14
2	La vue.....	14
3	Le contrôleur	14
4	Le moteur.....	14
5	Utilitaires	14
II	État de l’art	15
III	Analyse et conception	16
9	Génération d’un réseau	17
10	Simulation du réseau	19
1	Détermination de l’intervalle de temps	19
2	Configuration de la topologie du réseau.....	19
3	Détermination de l’ensemble dominant du réseau.....	19
4	Détermination du routage des capteurs.....	20
5	Détermination si la meilleure durée de vie a été atteinte.....	20
IV	Mise en oeuvre	22
11	Outils et bibliothèques utilisées	23
1	NetworkX.....	23
2	Plotly	23
3	Matplotlib	24
4	pyQt	24
12	Limites	27
13	Analyse des performances	28
V	Bilan et conclusion	29
14	Bilan sur la recherche	30
15	Bilan sur le développement	32
16	Conclusion	34

ANNEXE 1 : Spécifications Non Fonctionnelles	35
1 Contraintes de développement et conception	35
2 Contraintes de développement et conception	35
2.1 Performances et capacités.....	35
2.2 Contrôlabilité.....	35
2.3 Sécurité	35
ANNEXE 2 : Spécifications fonctionnelles	36
1 Contrôler les paramètres - <i>controleParametres</i>	36
2 Fonction Génération d'un réseau - <i>generationReseau</i>	36
2.1 Entrées	36
3 Fonction sauvegarde réseau - <i>sauvegardeReseau</i>	36
3.1 Entrées	37
4 Fonction charger réseau - <i>chargerReseau</i>	37
4.1 Entrées	37
5 Fonction de détermination de l'intervalle de temps - <i>determinerIntervalleTemps</i>	37
6 Détermination de la configuration topologique - <i>configurationTopologique</i>	37
6.1 Entrées	37
7 Détermination de l'ensemble dominant - <i>ensembleDominant</i>	37
7.1 Entrées	38
8 Détermination du routage des données - <i>routage</i>	38
8.1 Entrées	38
9 Lancer la simulation - <i>simulation</i>	38
9.1 Entrées	38
10 Simulation de la consommation énergétique - <i>consommationEnergie</i>	38
10.1 Entrées	38
11 Sauvegarde du résultat de la simulation - <i>sauvegardeResultats</i>	38
11.1 Entrées	39
12 Chargement du résultat d'une simulation - <i>chargerResultats</i>	39
12.1 Entrées	39
13 Afficher le réseau - <i>afficherReseau</i>	39
13.1 Entrées	39
14 Afficher les statistiques du réseau - <i>afficherStatistiques</i>	39
15 Détermination de la fin de vie du réseau - <i>determinationFinDeVie</i>	39
16 Comparaison des durées de vie - <i>comparaisonDureeDeVie</i>	39
16.1 Entrées	40
17 Récupération de la liste des états du réseau - <i>listeEtats</i>	40
ANNEXE 3 : Diagramme de classes	41

ANNEXE 4 : Évaluation des risques	43
ANNEXE 5 : Guide d'installation	44
ANNEXE 6 : Guide d'utilisation	51
ANNEXE 7 : Cahier de tests	61
ANNEXE 8 : État de l'art	88
Bibliographie	110

Table des figures

1 Contexte

- 1 Taxonomie des applications de réseaux de capteurs sans fils (Source : Thèse « Energy-efficiency in wireless sensor networks », traduction : Thibaud Beaufils) 3

3 Objectifs

- 1 Frise chronologique d'un cycle de vie du réseau telle que défini dans le projet. (Source : Thibaud Beaufils) 6

4 Base méthodologique

- 1 Schéma du cycle de développement en V (Source)..... 7

7 Fonctionnalités du système

- 1 Diagramme des cas d'utilisations (Source : Thibaud Beaufils) 12

8 Structure générale du système

- 1 Diagramme de classes simplifié (Source : Thibaud Beaufils)..... 13

9 Génération d'un réseau

1	Algorithme de répartition des points sur une surface fixe à deux dimensions. A - Le premier point est placé aléatoirement sur la surface. En rouge est représenté la surface où aucun autre point ne peut être placé. Le reste de la surface est divisé en 8 nouvelles surfaces. B - Le second point est placé aléatoirement dans la plus grande surface. Cette dernière est redivisée en 9 surfaces. Puis un autre point est placé de la même manière jusqu'à ce que tous les points soient placés. D - Un point peut être placé sur le bord d'une surface et ainsi voir sa zone tronquée. Ce choix a été effectué pour simplifier l'algorithme mais permet à certains points de se retrouver proches entre eux. (Sources : Thibaud Beaufiles).....	17
2	Diagramme d'activité. A gauche diagramme pour la création d'un réseau, à droite pour rendre un graphe connexe. (Sources : Thibaud Beaufiles)	18
10 Simulation du réseau		
1	Diagramme d'activité. A gauche diagramme pour la simulation du réseau, à droite pour la détermination de l'intervalle de temps. (Sources : Thibaud Beaufiles)	20
2	Diagramme d'activité. De gauche à droite et de haut en bas : le diagramme pour configurer topologiquement le réseau, le mécanisme pour déterminer si la durée de vie maximale a été atteinte, la détermination de l'arbre couvrant et l'assignation du routage aux noeuds (Sources : Thibaud Beaufiles)	21
11 Outils et librairies utilisées		
1	Logo de Plotly (Source).....	23
2	Un réseau de 59 capteurs représenté avec plotly. En haut à droite les fonctionnalités de visualisation (zoom, déplacement dans la vue, etc.). À droite l'échelle du niveau de batterie des capteurs. Au centre le réseau : l'hexagone jaune correspond à la passerelle, les ronds orange aux capteurs, les traits bleus la représentation des connexions entre capteurs. Sont entourés en rouge les capteurs de l'arbre couvrant (c'est-à-dire les capteurs émetteurs/récepteurs) et les traits rouges correspondent aux connexions comprises dans l'arbre couvrant. (Source : Thibaud Beaufiles)	24
3	Les deux graphiques affichés grâce à Matplotlib (Source : Thibaud Beaufiles)	25
4	Figure Rendu obtenu en incorporant le fichier html de plotly et les graphiques Matplotlib dans la même fenêtre. (Source : Thibaud Beaufiles)	25
5	Comparaison de différents algorithmes proposés par NetworkX pour la détermination d'un arbre couvrant A. Réseau initial B. Réseau après application de l'algorithme de détermination d'ensemble dominant et redondance des liens 1. Application de l'algorithme de Boruvka 2. Application de l'algorithme de Kruskal 3. Application de l'algorithme de Prim En rose le chemin unique d'un point violet vers la passerelle en rouge. On remarque que les deux premiers algorithmes sont similaires et moins performants en comparaison avec le troisième où la construction à partir de la racine est plus marquée. (Source : Beaufiles Thibaud)	26
12 Limites		
1	Graphique illustrant le temps d'exécution de la simulation en fonction du nombre de capteurs contenus dans le réseau. (Source : Thibaud Beaufiles)	27

13 Analyse des performances

- 1 Graphique illustrant l'amélioration obtenue avec le simulateur en fonction du nombre de capteurs contenus dans le réseau. (Source : Thibaud Beaufils) 28

14 Bilan sur la recherche

- 1 Retroplanning du premier semestre. (Source : Thibaud Beaufils) 31
- 2 Retroplanning du second semestre. (Source : Thibaud Beaufils)..... 31

15 Bilan sur le développement

- 1 Planning du second semestre. En orange avec une croix les étapes retardées puis reportées à la fin du développement. En horaire avec une horloge fléchée les étapes qui ont été reportées. En bleue les étapes du projet reportées telles qu'elles se sont déroulées. (Source : Thibaud Beaufils)..... 33

Introduction

Le Projet de Fin d'Etudes (PFE), réalisé en 5ème année, s'inscrit dans la formation dispensée à l'Ecole Polytechnique de Tours et constitue une réelle expérience en termes de conduite de projet. Le PFE se déroule du 19 septembre 2018 à mars 2019 à raison de deux jours par semaine. Il donne lieu à la rédaction d'un cahier de spécifications, d'un cahier d'analyse, d'un rapport et d'une présentation du travail effectué lors d'une soutenance.

La maîtrise d'ouvrage MOA est représentée par Ameer Soukhal membre de Polytech Tours. Ce document est le rapport du projet " Développement Algorithmique et Conception d'un Simulateur pour un Réseau de Capteurs dynamiques". Il définit les besoins, l'environnement du projet et les objectifs à réaliser. Ce rapport reprend les points évoqués dans le cahier des spécifications, présente la solution mise en œuvre et inclut l'ensemble des documents produits.

1

Contexte

Le projet se focalise sur l'étude des réseaux de capteurs. L'objectif est d'optimiser la consommation énergétique des capteurs afin d'augmenter la durée de vie du réseau. L'application peut se faire à petite comme à grande échelle. A petite échelle la technologie peut être appliquée au corps humain à des fins de contrôle de santé, interne ou externe au corps. A grande échelle, elle peut être retrouvée sur des surfaces agricoles pour améliorer les récoltes, dans une usine afin de contrôler des points peu accessibles ou bien encore sur une surface aquatique afin de contrôler la qualité de l'eau. La figure ci-dessous résume les nombreuses applications possibles.

L'optimisation de la consommation énergétique d'un tel réseau est capitale pour plusieurs raisons. Premièrement, et tout simplement pour des soucis d'autonomie, ensuite car certains cas d'utilisations incluent des capteurs dont l'accès est difficile puis finalement pour minimiser les coûts de maintenance du réseau. Ces points impliquent donc une minimisation du nombre de recharges nécessaires.

Le travail attendu à travers ce projet s'inscrit dans ce contexte où l'objectif final sera la gestion de la consommation énergétique des capteurs en assurant la continuité du service.

Au-delà de minimiser la consommation énergétique des capteurs, l'outil d'aide à la décision à développer permettrait de maximiser la durée du fonctionnement du réseau et ainsi déduire un plan de recharge des capteurs.

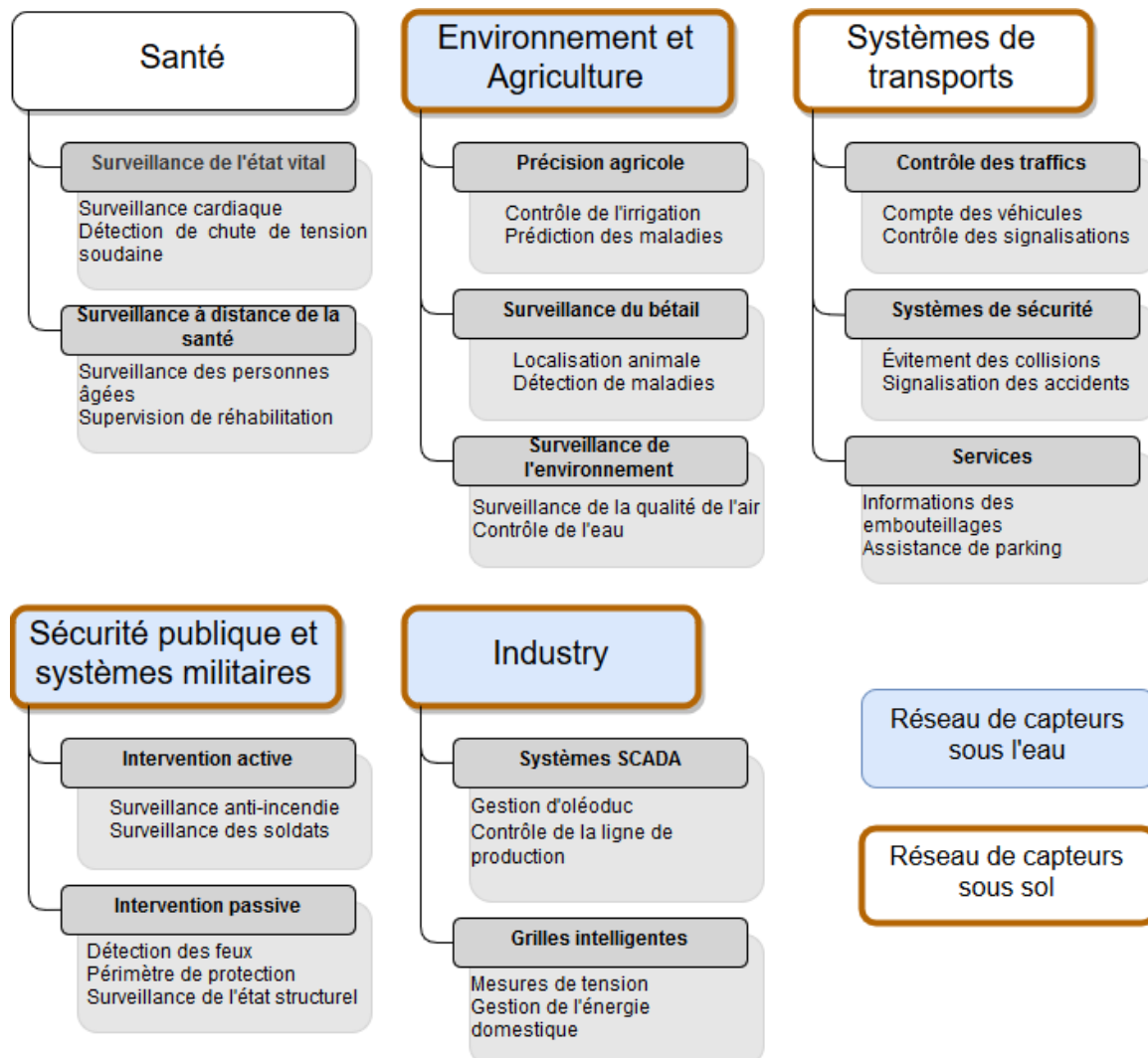


Figure 1 – Taxonomie des applications de réseaux de capteurs sans fils (Source : Thèse « Energy-efficiency in wireless sensor networks », traduction : Thibaud Beaufils)

2

Hypothèses

Pour la suite, nous avons décidé que le type de réseau étudié serait un réseau de capteurs avec batteries de capacité identique. Nous partirons du principe que tous les capteurs peuvent communiquer avec au moins un de leurs voisins.

Nous considérerons que sur un intervalle de temps donné, en plus de récolter des données, un capteur peut avoir deux rôles différents : celui d'envoyer ses données, dit capteur source, et celui de transmettre les données des autres capteurs, dit capteur relais. Ces derniers sont aussi émetteurs. On parle alors à un instant donné de capteurs émetteurs et de capteurs émetteurs/récepteurs. Chaque action identique consommerait la même quantité d'énergie et chaque émetteur envoie plus ou moins la même quantité d'information sur l'intervalle de temps considéré.

3

Objectifs

La finalité du projet est la production d'un logiciel d'optimisation du routage des données au sein d'un réseau de capteurs en prenant en compte la consommation énergétique locale. Il faut déterminer le rôle individuel des capteurs à un instant donné, sachant qu'une réception/émission d'un capteur consomme davantage qu'une simple émission.

Le logiciel devra d'abord générer une topologie d'un réseau sous forme d'un graphe connexe, où l'ensemble des nœuds représentent les capteurs et les arcs les connexions possibles entre ceux-ci. On considère qu'une connexion entre deux capteurs est possible s'ils sont à portée, à la manière d'une connexion sans-fil.

Dans cette étude nous nous intéressons au cas où il y a un unique nœud puit, passerelle entre le réseau et l'extérieur. Cependant elle devra être adaptable au cas où il y aurait plusieurs puits.

Divers paramètres variables sont à prendre en compte :

- La capacité de la batterie
- La durée de vie du réseau (en pourcentage fonctionnel du réseau), c'est-à-dire à partir de combien de capteurs non reliés à la passerelle le réseau est considéré comme inopérant.
- La consommation de chaque action du capteur

Étant donné que chaque capteur peut avoir deux rôles possibles, un de nos objectifs sera de savoir quel sera le rôle de chacun, émetteur ou récepteur/émetteur, pour chaque période de temps considérée.

Temporellement les objectifs sont :

- Faire l'état de l'art afin de :
 - Pouvoir calculer le taux de consommation énergétique d'un capteur en fonction de l'action effectuée (émission ou réception/émission)
 - Sélectionner les méthodes de résolution optimale de routage, de test de connexité et de détermination d'ensemble dominant.
 - Analyser les générateurs de topologies existants.
 - Étudier les travaux proches du problème traité.
- Générer aléatoirement la topologie d'un réseau suivant les hypothèses émises précédemment.
- Déterminer la configuration topologique dynamique, c'est-à-dire déterminer le rôle de chaque capteur durant un cycle de durée optimale à calculer.
- Développer et implémenter une configuration donnée du réseau, un algorithme de routage.

- Développer et implémenter un algorithme qui permettra de déterminer le temps minimum, noté ΔT , entre les changements de rôle des capteurs tout en maximisant la durée de vie du réseau, noté T (cf. Figure 1).

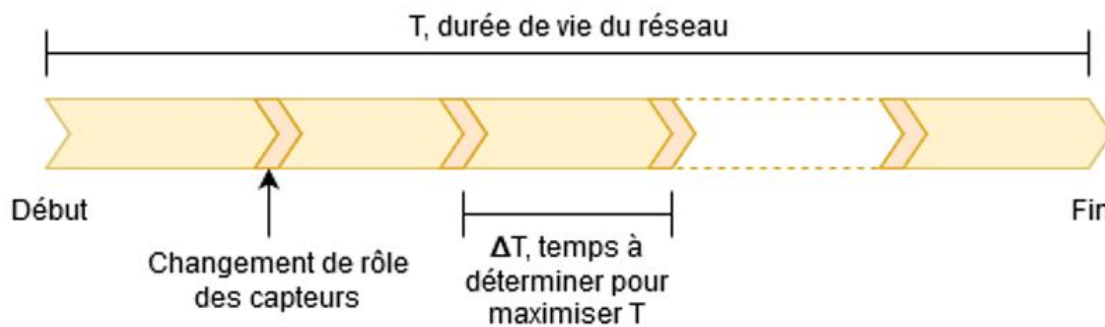


Figure 1 – Frise chronologique d'un cycle de vie du réseau telle que défini dans le projet. (Source : Thibaud Beaufils)

Autrement dit, ce travail permettra entre autres de répondre aux questions suivantes :

- Quels capteurs sont émetteurs et lesquels sont émetteurs/récepteur durant ΔT pour trouver l'ensemble dominant ?
- Quel est le routage qui doit être mis en place à un instant donné ?
- Quelle est la durée de vie maximale du réseau ?

4

Base méthodologique

1 Organisation du projet

Un rapport hebdomadaire est rédigé afin de rendre compte de l'avancement du projet et de fixer des objectifs pour la semaine qui suit.

Étant donné que le projet peut se décomposer en fonctionnalités, nous avons choisis d'utiliser l'agilité hybridée avec un cycle en V comme méthode de développement. Cependant, n'ayant pas d'intégration dans un environnement particulier à effectuer, les parties « Conception Architecturale » et « Tests d'intégration » ne nous intéresseront pas.

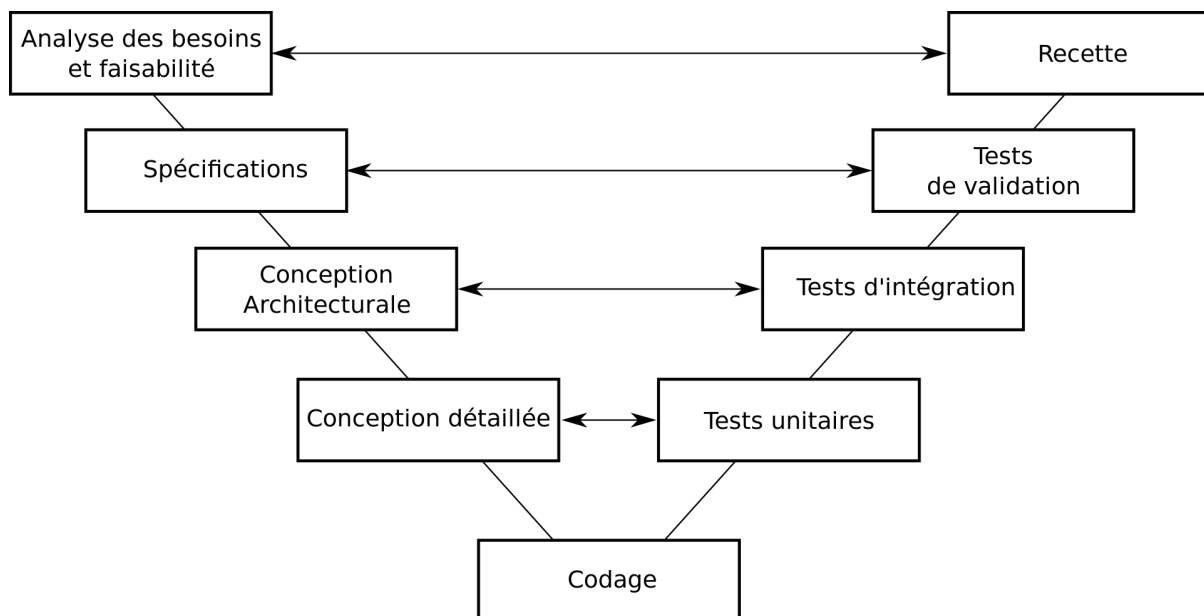


Figure 1 – Schéma du cycle de développement en V (Source)

2 Conventions de développement

Lors du développement du projet quelques règles de bonnes pratiques sont à suivre. Parmi celles-ci se trouvent des conventions de nommage :

- **Noms de classes** commençant par une majuscule, chaque nouveau mot avec une majuscule. Ex : *SuperClasse*
- **Noms de méthodes** commençant par l'anagramme du nom de classe, puis minuscule au premier mot, chaque nouveau mot avec une majuscule. Ex : *SCmethodeGeniale*
- **Noms d'attributs** commençant par l'anagramme du nom de classe, puis caractère de soulignement puis minuscule à chaque mot, chaque nouveau mot séparé par un caractère de soulignement. Ex : *SC_attribut_merveilleux*
- **Objets** commençant par l'anagramme du nom de classe puis chaque mot avec une majuscule. Ex : *SCInstanceMagnifique*
- **Fonctions** commençant par une minuscule, chaque nouveau mot avec une majuscule. Ex : *fonctionEpoustouflante*
- **Variables** commençant par un caractère de soulignement puis minuscule à chaque mot, chaque nouveau mot séparé par un caractère de soulignement. Ex : *_variable_extraordinaire*
- Les **constantes** respectent les mêmes règles qu'évoqués ci-dessus à l'exception qu'elles doivent être écrites en majuscules.

Le code devra être suffisamment commenté pour pouvoir être compris facilement par une personne qui reprendrait le projet, ainsi chaque rôle de méthode et fonction devront être soigneusement commentés de même que chaque module, classe, variables ou attributs dont le nom ne suffit pas à comprendre son utilité ainsi que chaque bloc algorithmique complexe.

Première partie

Description Générale

5

Environnement du projet

Le projet se base uniquement sur de la documentation scientifique, ainsi il n'y a pas d'environnement existant dans lequel intégrer le projet. De plus, le projet étant presque exclusivement de la recherche opérationnelle, de la modélisation et de la simulation, aucun matériel n'aura besoin d'être intégré.

6

Caractéristiques des utilisateurs

Les utilisateurs ciblés du logiciel maîtrisent l'informatique. Ils ont les connaissances nécessaires sur les réseaux de capteurs ainsi que sur les graphes pour l'utiliser. L'application doit être suffisamment ergonomique pour pouvoir être prise en main intuitivement à la première utilisation. En plus de cela, il sera important d'élaborer une documentation précise et complète.

7

Fonctionnalités du système

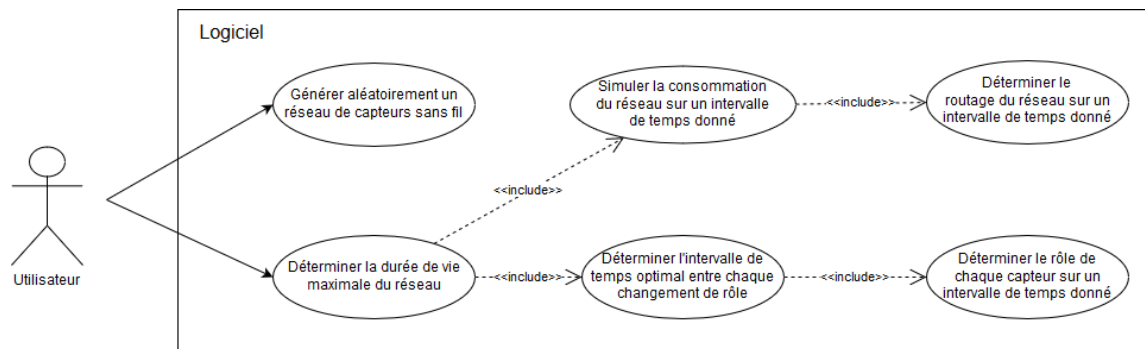


Figure 1 – Diagramme des cas d'utilisations (Source : Thibaud Beaufiles)

L'utilisation du logiciel est très ciblée et implique un seul utilisateur par instance. Il y a six cas généraux d'utilisation :

- L'utilisateur peut demander la génération aléatoire d'un réseau en fournissant divers paramètres.
- Le logiciel peut déterminer la durée de vie T maximale du réseau.
- Le logiciel peut déterminer le routage du réseau pour chaque intervalle de temps ΔT .
- Le logiciel peut déterminer le rôle de chaque capteur pour chaque intervalle de temps ΔT .
- Il peut déterminer l'intervalle de temps ΔT optimal entre chaque changement de rôle.
- Enfin il peut simuler la consommation du réseau sur un intervalle de temps ΔT et plus généralement sur la durée de sa vie T .

Structure générale du système

La figure 1 présente le diagramme de classes du projet. La version détaillée est disponible en annexes.

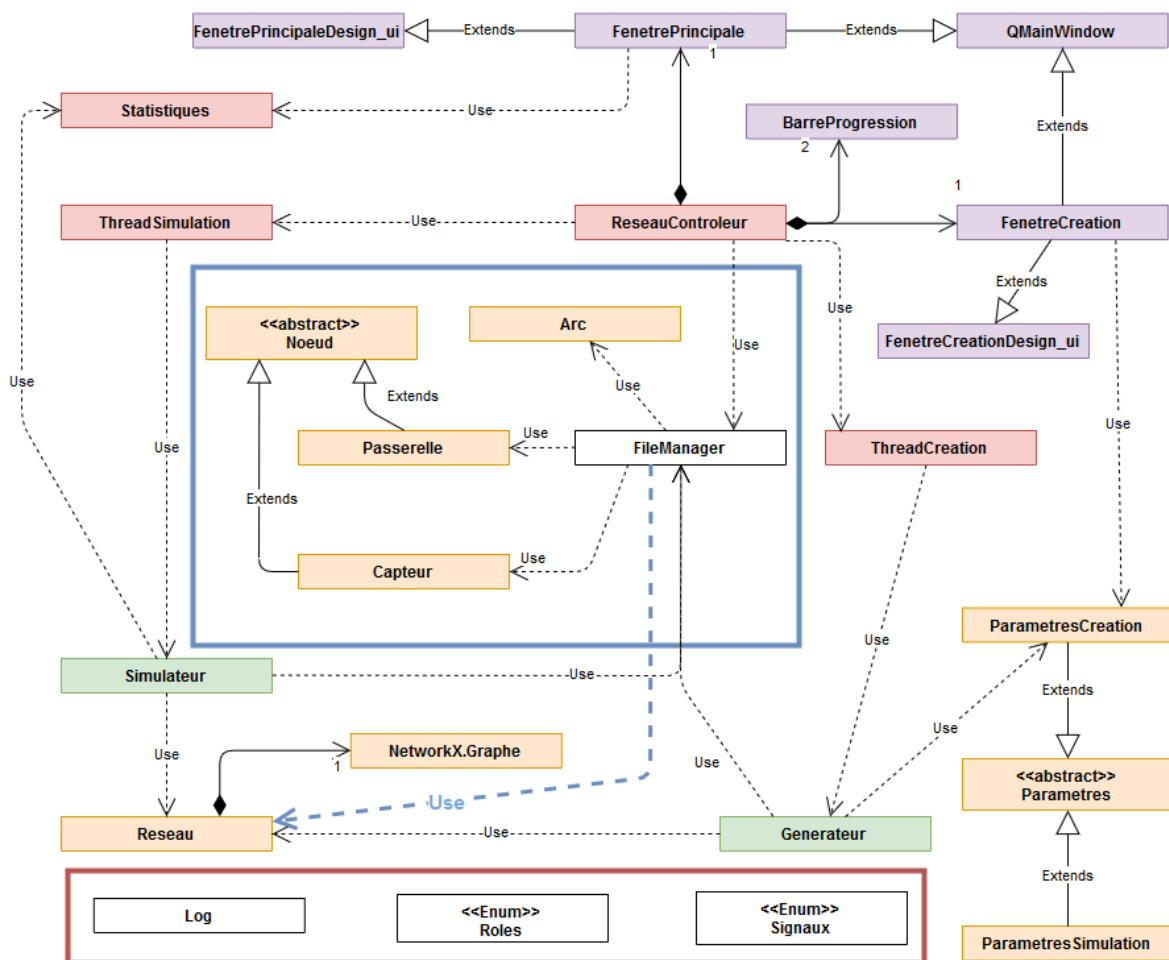


Figure 1 – Diagramme de classes simplifié (Source : Thibaud Beaufils)

Le système se décompose en cinq parties.

1 Le modèle

Sur le diagramme, en orange, cette partie inclue deux classes principales : `Reseau` et `Parametres`. Le premier modélise un réseau en utilisant les éléments fournis par la bibliothèque `NetworkX` dont nous reparlerons plus loin. Le second contient les paramètres saisis par l'utilisateur à travers les vues.

2 La vue

En violet sur le diagramme, se retrouve deux fenêtres qui héritent des fonctionnalités de la classe `QMainWindow` de la bibliothèque `Qt`. La première correspond à la fenêtre principale, la seconde permet à l'utilisateur de saisir les paramètres nécessaires à la création d'un réseau. Cette partie inclue également la gestion de barre de chargement.

3 Le contrôleur

En rouge, la classe qui fait le lien entre la vue et le modèle, `ReseauControleur`, fait office de contrôleur dans le système. Elle lance la simulation ou la génération de réseau à travers les `Threads` correspondants. Cette partie inclue également la classe `Statistique` qui compile les données récoltées lors de la simulation afin qu'elles puissent être exploitables.

4 Le moteur

Cette partie, en vert sur le diagramme, inclue deux classes : `Simulateur` et `générateur`. Ils contiennent respectivement l'ensemble algorithmique pour simuler la vie d'un réseau de capteurs et pour en créer un.

5 Utilitaires

Hormis ces quatre premières divisions, il y a deux autres regroupements de classes possibles. La première correspond à l'ensemble des classes contenues dans le rectangle rouge. S'y retrouve l'utilitaire `FileManager` permettant d'enregistrer le réseau, les statistiques de la simulation mais également l'ensemble des états intéressants dans lequel le réseau est passé lors de la simulation. `FileManager` permet également d'importer ces données. Il utilise en intermédiaire les classes `Arc`, `Passerelle` et `Nœuds` entre les données brutes et la création d'un objet de type `Reseau`.

La seconde contient la classe `Log`, utilisée par l'ensemble des classes afin de pouvoir vérifier le déroulement du programme en production, et également deux énumérations utilisées par le moteur, le contrôleur et le modèle.

Deuxième partie

État de l'art

Dans ce projet, l'état de l'art a pour premier objectif de fixer le contexte et d'orienter les objectifs par l'étude des travaux proches. Son second rôle est de permettre de choisir des solutions à des problématiques techniques.

Le travail qui se rapproche le plus de notre situation est la thèse de Tifenn Rault (17 janvier 2017) : « Energy-efficiency in wireless sensor networks » (Optimisation de la consommation énergétiques des réseaux sans fils). Le résumé commenté est disponible Annexe. Cette étude a permis de comprendre de façon globale le fonctionnement d'un réseau sans fil, ses avantages et les problématiques qu'il implique. Elle nous a également permis d'avoir des bases sur l'optimisation de la consommation énergétique et nous a ouvert à de nombreuses solutions.

Outre cet étude, l'état de l'art a ensuite l'objectif de répondre à des questionnements sur des choix techniques. Dans notre cas nous pouvons en dénombrer trois principaux :

1. La génération de topologies de réseaux sans fil, incluant la recherche de modèles existant permettant de représenter fidèlement notre définition de réseau et de mettre en place nos contraintes fonctionnelles.
2. Le calcul de la consommation énergétique d'un capteur en fonction de l'action effectuée et ainsi, de son rôle.
3. Le routage des données sur une durée ΔT , incluant la recherche de différents tests de connexité et de détermination d'ensemble dominant afin de pouvoir déterminer le rôle des capteurs.

Il est primordial que des solutions soient choisies pour ces trois problématiques avant le début de la conception détaillée des lots les concernant (c'est-à-dire les lots 1 et 2). Cette phase de recherche a permis de découvrir une bibliothèque Python qui propose des solutions aux problématiques précédente. NetworkX contient en effets un panel très complet concernant la modélisation de réseaux à travers la notion de graphes.

Troisième partie

Analyse et conception

De la phase de recherche a suivie une phase de conception. En a découlé un ensemble algorithmique complet illustrant le fonctionnement interne du logiciel à produire. Ce chapitre expose les diagrammes de classes, illustrations de cette recherche algorithmique.

9

Génération d'un réseau

La génération d'un réseau correspond à un algorithme principal `creerReseau` et à un sous-ensemble algorithmique qu'il est nécessaire de détailler pour la compréhension de la complexité de l'algorithme.

Le sous-ensemble, `connecteur`, permet de rendre connexe (cad formant un ensemble unique) un multigraphe en déplaçant ses sous-graphes un à un vers le plus grand.

L'algorithme principal utilise ce sous-algorithme de sorte à rendre connexe le graphe généré à l'étape « `generationReseau` », juste après la génération des positions détaillée Figure 1. C'est seulement à l'avant dernière étape que le graphe prend le sens de réseau en récupérant les paramètres fournis. La dernière étape, `configurationTopologique`, prépare le réseau à être utilisé pour la simulation. Son fonctionnement est détaillé plus loin.

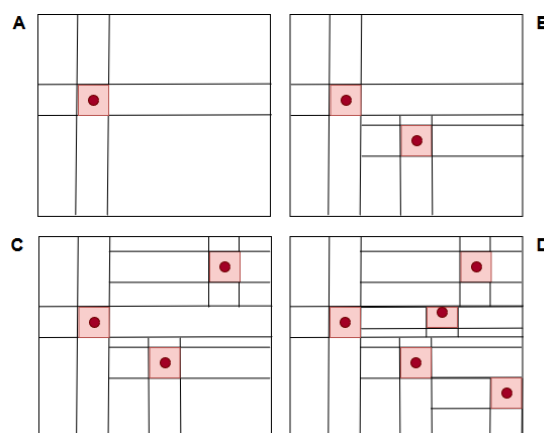


Figure 1 – Algorithme de répartition des points sur une surface fixe à deux dimensions.

A - Le premier point est placé aléatoirement sur la surface. En rouge est représenté la surface où aucun autre point ne peut être placé. Le reste de la surface est divisé en 8 nouvelles surfaces.

B - Le second point est placé aléatoirement dans la plus grande surface. Cette dernière est redivisée en 9 surfaces. Puis un autre point est placé de la même manière jusqu'à ce que tous les points soient placés.

D - Un point peut être placé sur le bord d'une surface et ainsi voir sa zone tronquée. Ce choix a été effectué pour simplifier l'algorithme mais permet à certains points de se retrouver proches entre eux.

(Sources : Thibaud Beauvils)

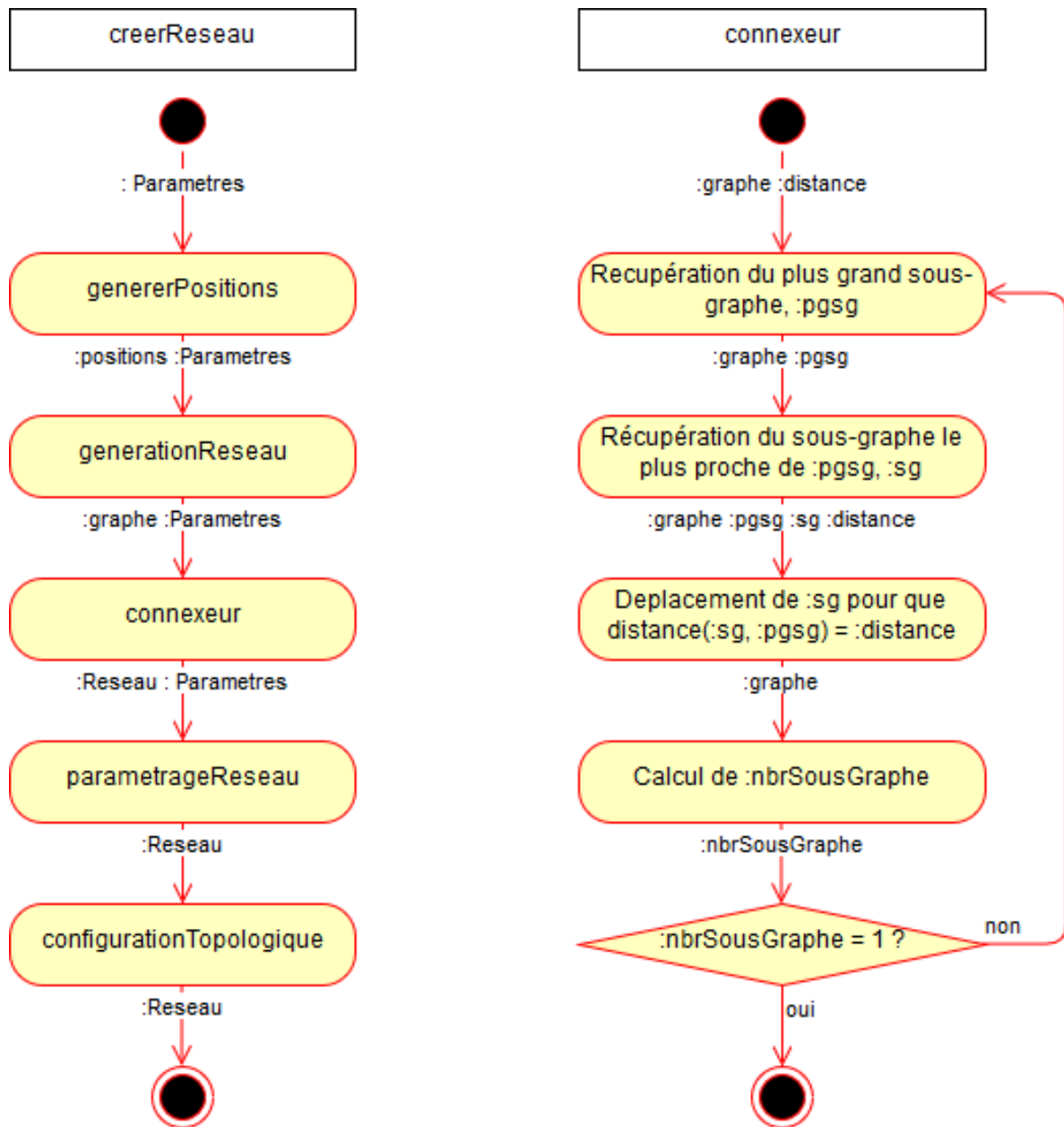


Figure 2 – Diagramme d'activité. A gauche diagramme pour la création d'un réseau, à droite pour rendre un graphe connexe. (Sources : Thibaud Beaufiles)

10

Simulation du réseau

L'algorithme principale est décomposé en deux boucles. La première correspond à un cycle. A chaque cycle un nouvel intervalle de temps est déterminé et le réseau est réinitialisé. La seconde permet de simuler la consommation énergétique du réseau tout en changeant le rôle des capteurs le moment voulu. Il peut être divisé en cinq sous-ensembles algorithmiques (Figures 2 (Chapitre 9), 1 et 2).

1 Détermination de l'intervalle de temps

La détermination de l'intervalle de temps est l'étape clef pour la résolution du problème initiale, c'est-à-dire calculer l'intervalle de temps optimal. L'objectif est d'explorer le maximum de possibilités tout en convergeant vers la meilleure.

Dans l'objectif de cette exploration, l'algorithme se rapproche d'un raisonnement dichotomique. En effet, après avoir passé les deux cycles initiaux, la détermination de l'intervalle se fait toujours après deux cycles d'exécution, la première en explorant la solution utilisant un intervalle de temps moitié moins grand que le précédent ; la seconde avec un intervalle moitié plus grand.

2 Configuration de la topologie du réseau

Cette partie permet au simulateur d'exploiter le réseau. Elle définit le rôle des capteurs à partir de son ensemble dominant et détermine le routage du réseau.

3 Détermination de l'ensemble dominant du réseau

L'ensemble dominant correspond à l'arbre couvrant du réseau. C'est-à-dire l'ensemble minimal de capteurs tels les autres capteurs fassent partie de l'ensemble ou y soient connectés.

Dans ce but, il est nécessaire d'exclure les capteurs sans énergie avant d'extraire les nœuds de l'ensemble dominant pour ensuite les relier entre eux et former un graphe finalement transformé en arbre dont la racine est la passerelle.

La figure 2 (Chapitre 11) montre le résultat obtenu après application de l'algorithme.

4 Détermination du routage des capteurs

Une fois l'arbre dominant déterminé, il est possible d'indiquer aux capteurs vers qui envoyer les informations récoltées ou reçues. Pour cela l'arbre est remonté à partir de la racine en assignant chaque nœud précédent comme routage des capteurs parcourus. Ensuite les autres capteurs sont routés vers le capteur dominant ayant le plus d'énergie.

5 Détermination si la meilleure durée de vie a été atteinte

Cette étape, aussi importante que le calcul de l'intervalle de temps suivant, permet de déterminer si la simulation doit continuer. Pour cela le sous-ensemble algorithmique fait écho aux cycles évoqués précédemment. Les deux premiers cycles sont ignorés sauf si le second résultat n'est pas meilleur que le premier. Sinon tout les deux cycles l'algorithme vérifie que l'un des deux résultats obtenus est meilleur que le précédent. Si c'est le cas l'algorithme continue de creuser dans cette direction, sinon la simulation se conclue.

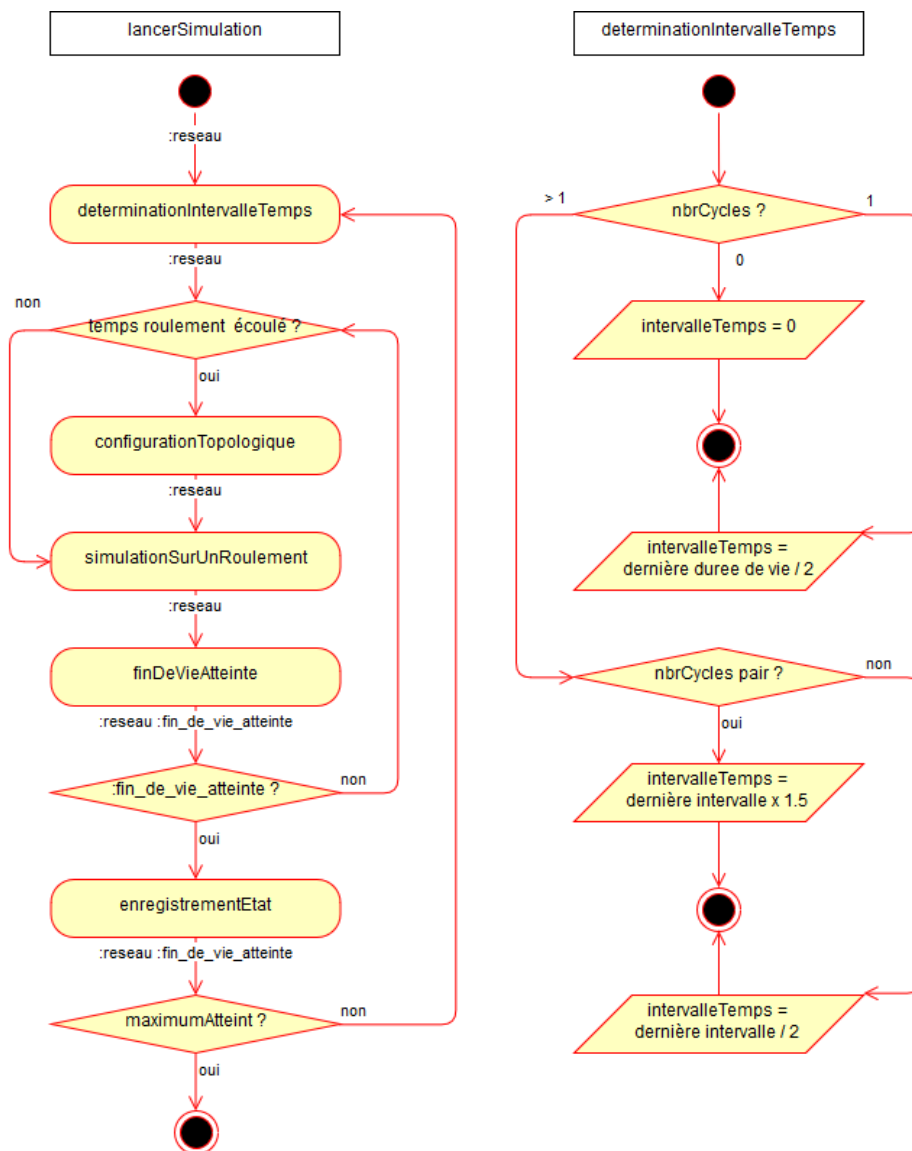


Figure 1 – Diagramme d'activité. À gauche diagramme pour la simulation du réseau, à droite pour la détermination de l'intervalle de temps. (Sources : Thibaud Beaufils)

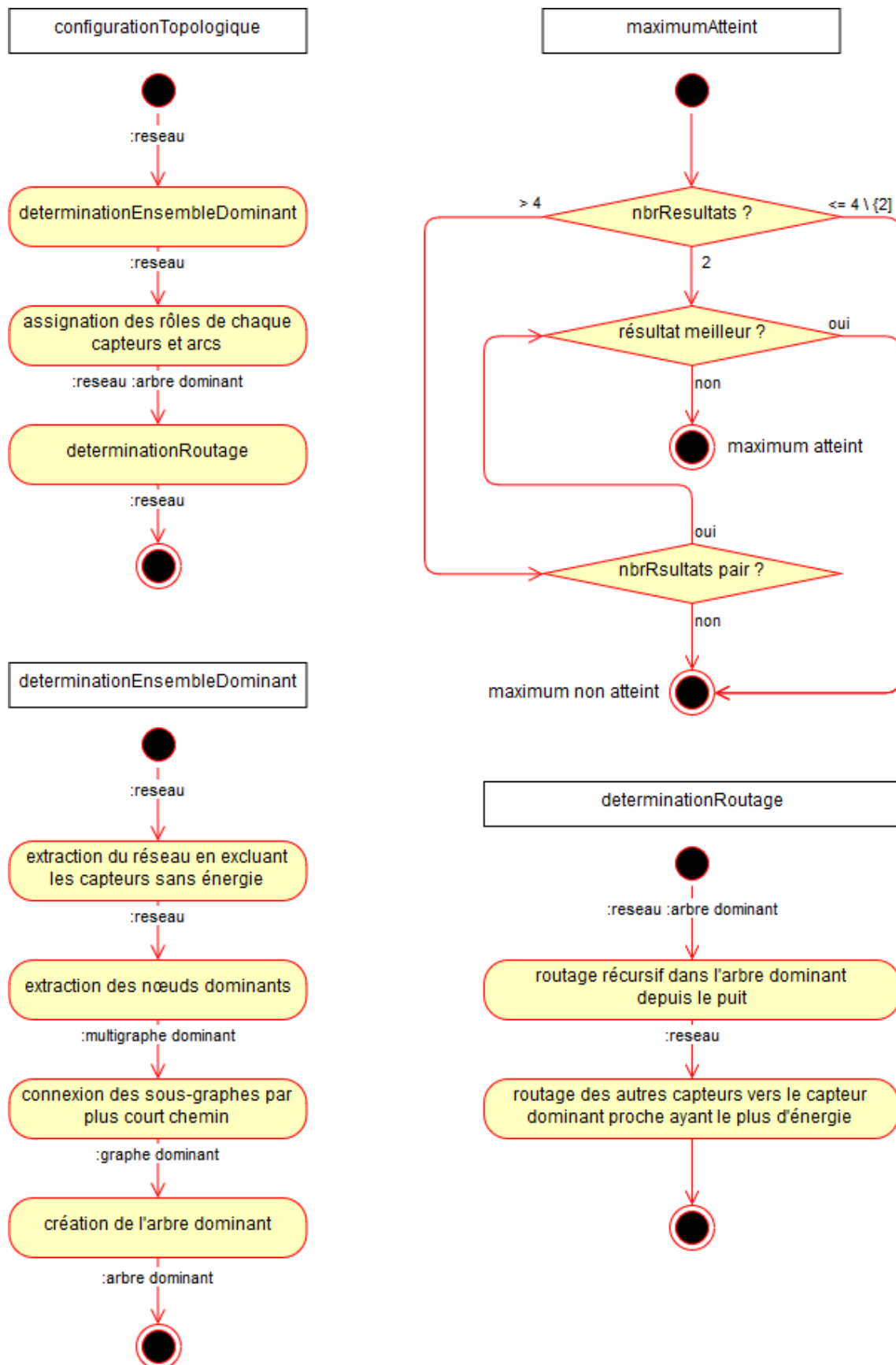


Figure 2 – Diagramme d'activité. De gauche à droite et de haut en bas : le diagramme pour configurer topologiquement le réseau, le mécanisme pour déterminer si la durée de vie maximale a été atteinte, la détermination de l'arbre couvrant et l'assignation du routage aux noeuds (Sources : Thibaud Beaufils)

Quatrième partie

Mise en oeuvre

Dans ce chapitre nous aborderons l'application de la solution envisagée. Pour le détail du fonctionnement de l'application en tant qu'utilisateur ainsi que l'interprétation des résultats se référer au manuel d'utilisateur en annexe.

11

Outils et librairies utilisées

1 NetworkX

Comme il a été évoqué précédemment, la bibliothèque NetworkX a été utilisée pour ses fonctionnalités de modélisation de Réseau et de manipulation de graphes. Dans le logiciel ont notamment été incorporés les algorithmes nécessaires à la détermination du plus court chemin et de l'ensemble dominant (voir Figure 5)

2 Plotly

Un avantage de la bibliothèque NetworkX est que son modèle est compatible avec la bibliothèque Plotly. Cette bibliothèque permet l'affichage avancée de graphiques scientifiques, que ce soit de l'affichage 2D, 3D de graphiques, de graphes mais aussi de cartes de chaleurs ou géographiques. Plotly génère son rendu dans un fichier html scripté permettant ainsi d'y inclure de nombreuses animations et fonctionnalités.

Dans le cadre de notre projet cette bibliothèque permet d'afficher le réseau avec différentes couleurs suivant de rôles de chaque élément et diverses fonctionnalités de visualisation.



Figure 1 – Logo de Plotly (*Source*)

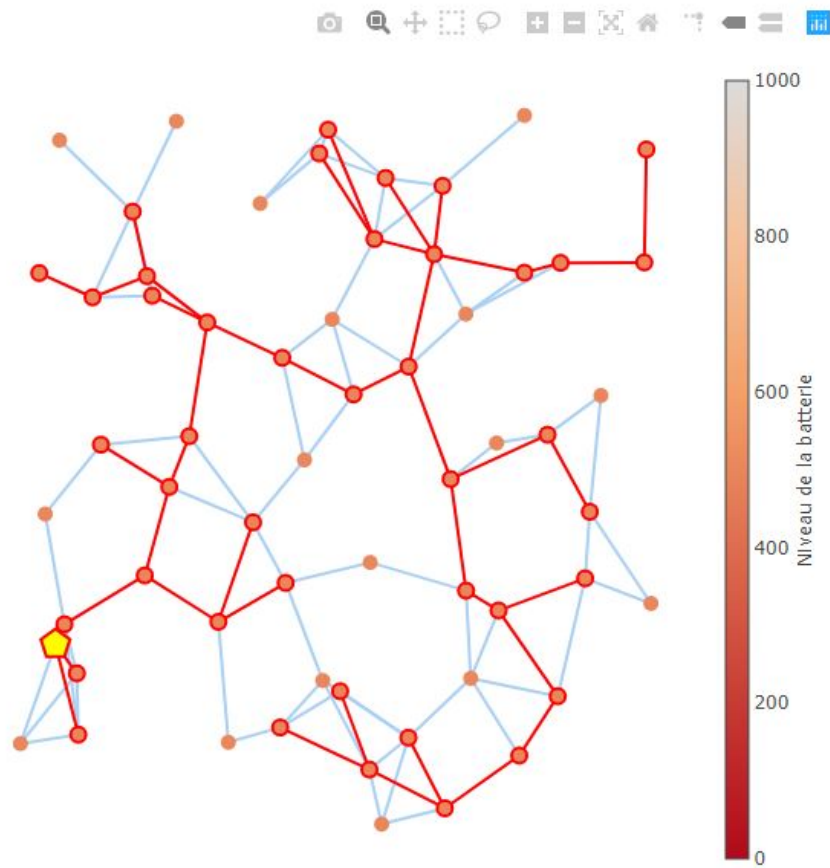


Figure 2 – Un réseau de 59 capteurs représenté avec plotly. En haut à droite les fonctionnalités de visualisation (zoom, déplacement dans la vue, etc.). À droite l'échelle du niveau de batterie des capteurs. Au centre le réseau : l'hexagone jaune correspond à la passerelle, les ronds orange aux capteurs, les traits bleus la représentation des connexions entre capteurs. Sont entourés en rouge les capteurs de l'arbre couvrant (c'est-à-dire les capteurs émetteurs/récepteurs) et les traits rouges correspondent aux connexions comprises dans l'arbre couvrant. (Source : Thibaud Beaufils)

3 Matplotlib

En plus de l'affichage du réseau, il est nécessaire d'afficher des graphiques représentant les résultats de la simulation en temps réel. Matplotlib est une bibliothèque connue et efficace de Python correspond à nos attentes.

4 PyQt

Après le choix des bibliothèques Plotly et Matplotlib, l'une des difficultés a été de déterminer comment réaliser une interface utilisateur ergonomique et intuitive. L'enjeu était donc de combiner astucieusement les deux rendus générés.

Les bibliothèques graphiques performantes et rapides d'utilisation sous python sont peu nombreuses, c'est pour cela que le choix a rapidement été orienté vers la bibliothèque PyQt, l'extension python de Qt. En effet, PyQt donne la possibilité au développeur d'incorporer une vue vers une fenêtre html (pour visualiser le rendu Plotly) de même que l'extension de Matplotlib conçue pour PyQt.

De plus, PyQt permet d'exploiter les fichiers créés par l'interface graphique de mise en page d'interfaces graphiques, QtCreator. Il suffit de compiler les dits fichiers en fichiers interprétables par python.

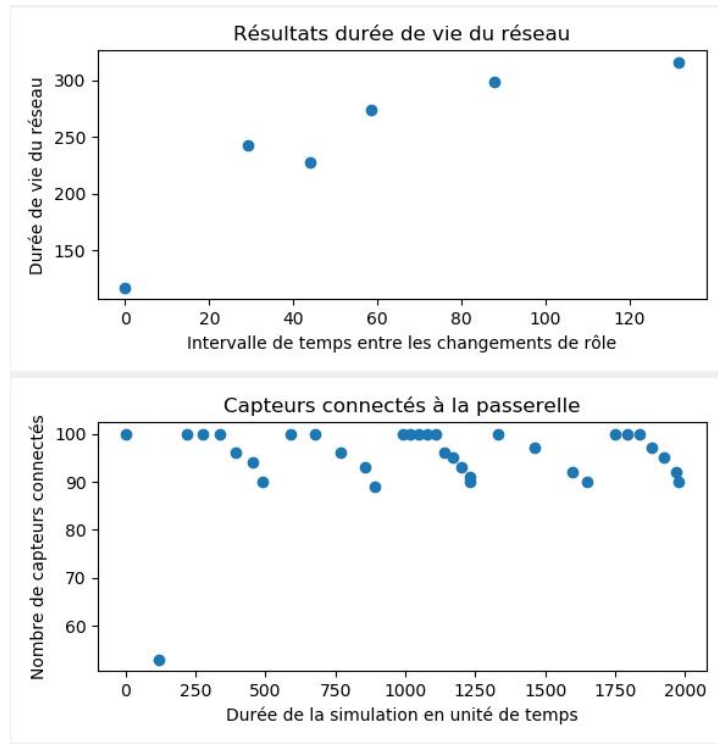


Figure 3 – Les deux graphiques affichés grâce à Matplotlib (Source : Thibaud Beaufils)

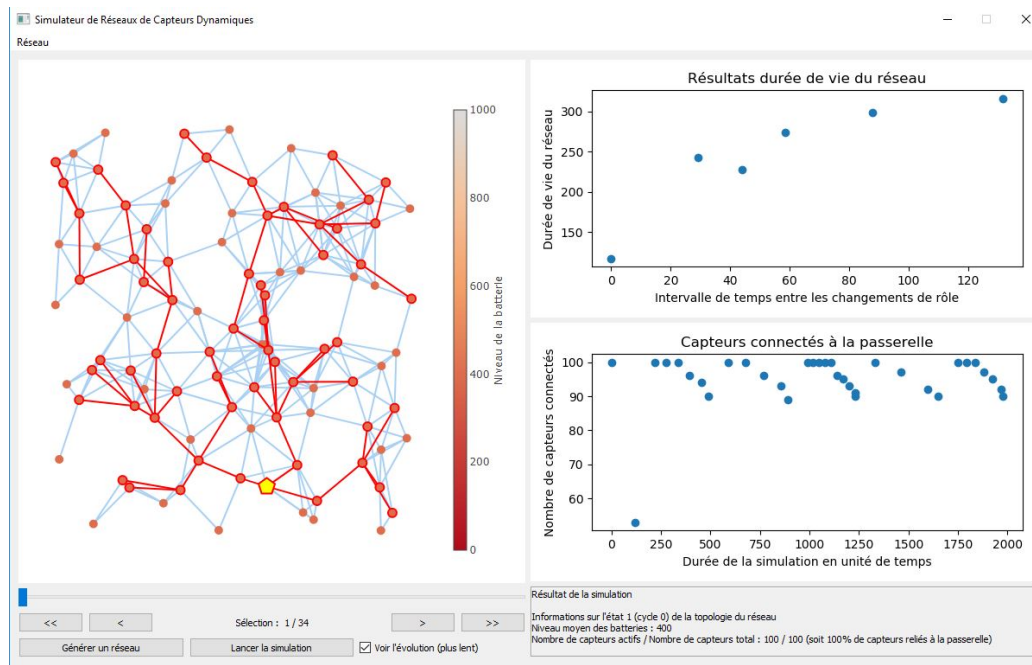


Figure 4 – Figure Rendu obtenu en incorporant le fichier html de plotly et les graphiques Matplotlib dans la même fenêtre. (Source : Thibaud Beaufils)

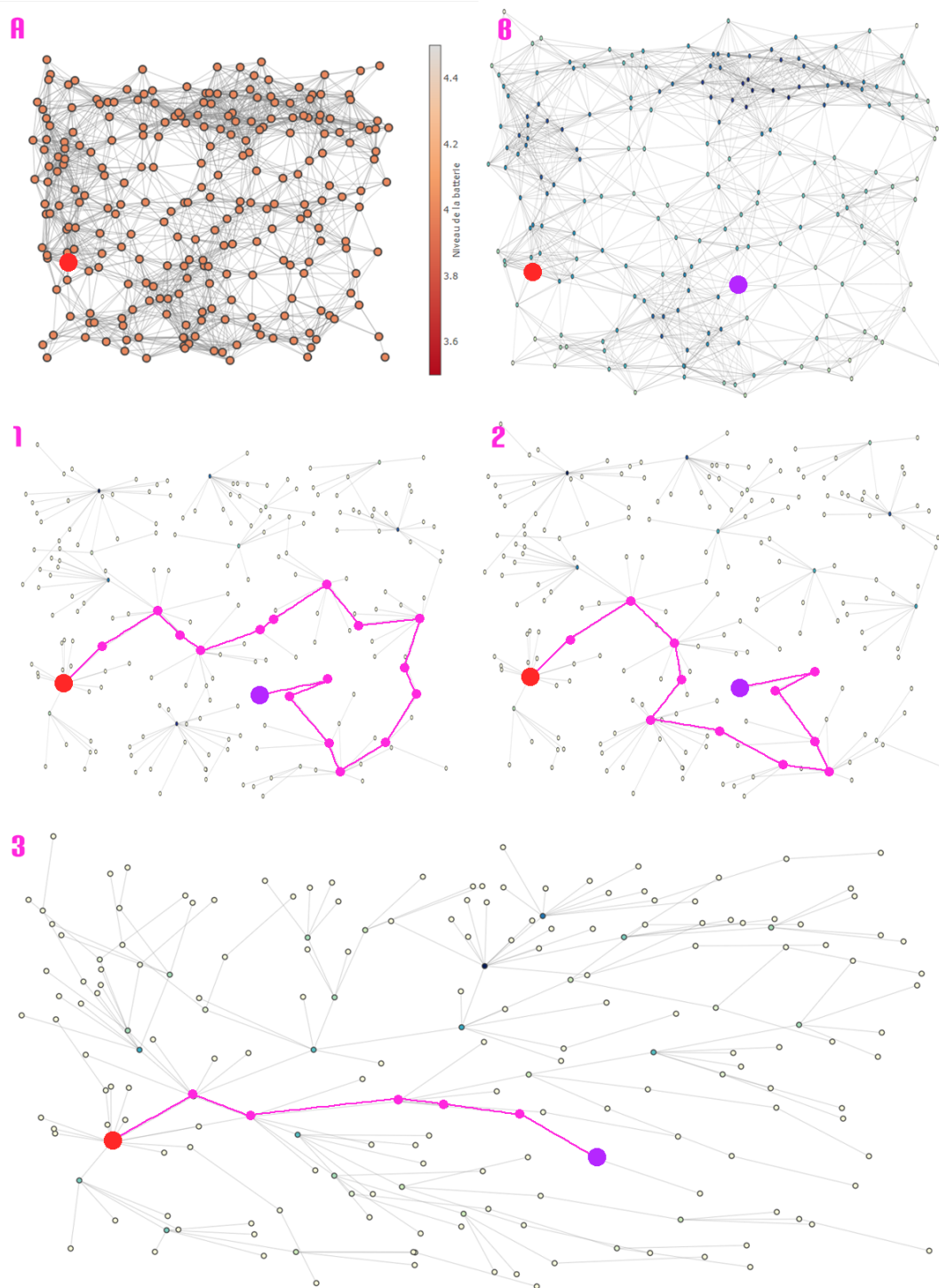


Figure 5 – Comparaison de différents algorithmes proposés par NetworkX pour la détermination d'un arbre couvrant A. Réseau initial B. Réseau après application de l'algorithme de détermination d'ensembles dominant et redondance des liens 1. Application de l'algorithme de Boruvka 2. Application de l'algorithme de Kruskal 3. Application de l'algorithme de Prim En rose le chemin unique d'un point violet vers la passerelle en rouge. On remarque que les deux premiers algorithmes sont similaires et moins performants en comparaison avec le troisième où la construction à partir de la racine est plus marquée. (Source : Beaufile Thibaud)

12

Limites

La première contrainte pour l'utilisateur du logiciel tel qu'il a été conçu est due à un problème de performance. En effet, suite à différents tests il a été conclu que la simulation ne permettait pas d'utiliser des réseaux de plus de 250 capteurs. Le graphique Figure 1 montre le temps d'exécution nécessaire suivant le nombre de capteurs, ce nombre s'envole au-delà de 200 capteurs pour atteindre une durée d'exécution supérieure à 30 minutes, dépassant ainsi largement les cinq minutes maximales estimées. De la même façon, en augmentant le nombre de connexions entre les capteurs le temps d'exécution augmente exponentiellement. Ce paramètre a ainsi également été limité. Un des enjeux d'une amélioration pourrait conduire à la réduction du temps d'exécution.

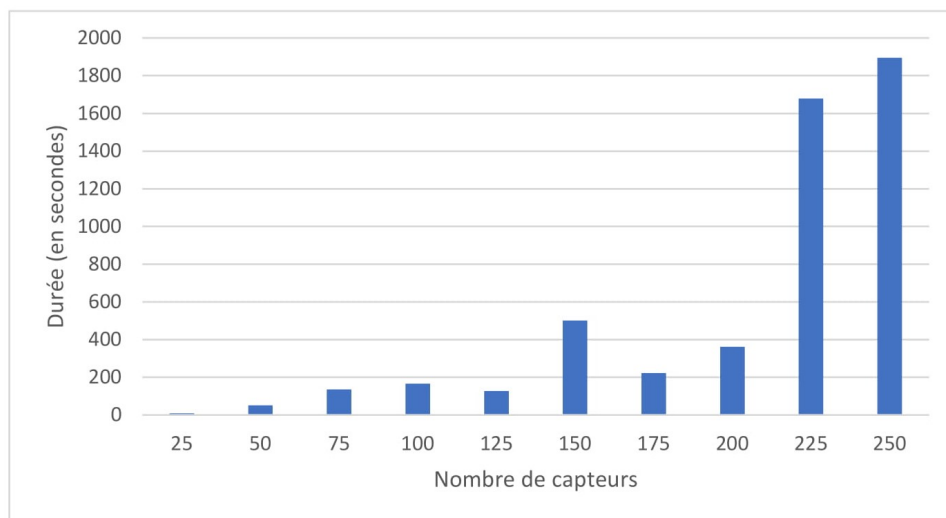


Figure 1 – Graphique illustrant le temps d'exécution de la simulation en fonction du nombre de capteurs contenus dans le réseau. (Source : Thibaud Beaufiles)

13

Analyse des performances

L'état de l'art n'ayant pas abouti à la découverte d'un simulateur similaire à notre solution il a donc fallu émettre une méthodologie différente de mesure des résultats. La simulation calcule toujours un premier cycle sans changer le rôle des capteurs, ce qui permet de toujours obtenir le résultat le plus faible de la simulation. En réalité ce résultat est la durée de vie la plus courte qui puisse être obtenu pour un réseau donné étant donné que celui-ci reste statique durant le premier cycle. Il était donc intéressant de pouvoir le meilleur résultat obtenu par la simulation avec le résultat initial. Le graphique figure 1 permet de visualiser cette amélioration qui peut dépasser les 100% d'amélioration avec une moyenne d'environ 50% d'amélioration.

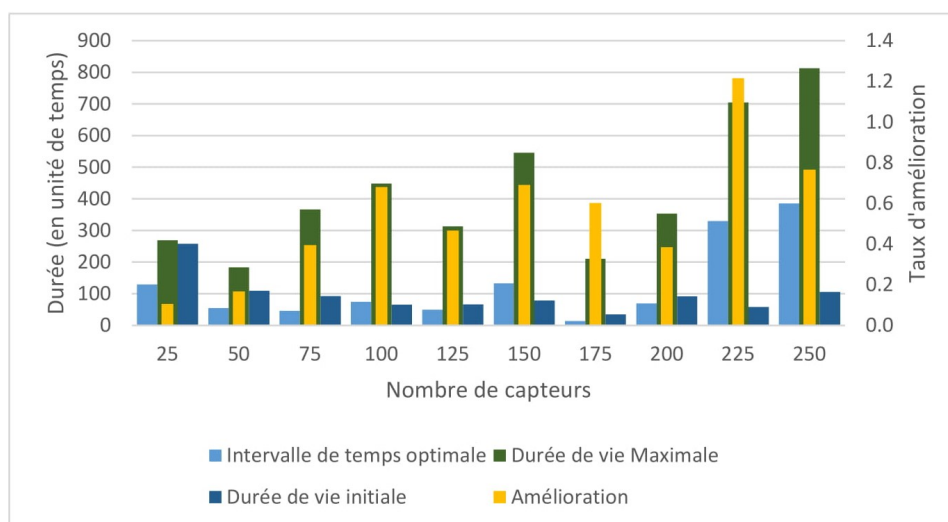


Figure 1 – Graphique illustrant l'amélioration obtenue avec le simulateur en fonction du nombre de capteurs contenus dans le réseau. (Source : Thibaud Beaufiles)

Cinquième partie

Bilan et conclusion

14

Bilan sur la recherche

L'étude de la littérature scientifique lors de l'état de l'art nous a permis de contextualiser le projet et, une fois rapporté à notre situation, d'en déduire des objectifs. Par la suite, il a été intéressant d'approfondir les recherches vers les références de la thèse de Mme Rault et de trouver des solutions sur les problématiques techniques exposées dans la partie de l'état de l'art. Le logiciel développé consiste en un simulateur du fonctionnement du réseau jusqu'à l'obtention de la durée de vie maximale et de l'intervalle de temps optimal pour le changement de rôle des capteurs. Finalement, nous répondons à différents besoins :

- La génération aléatoire d'un réseau de capteurs
- La détermination du rôle de chaque capteur lors de chaque changement
- La détermination du routage du réseau à chaque changement de rôle des capteurs
- La détermination de la fréquence optimale de changements de rôle
- La simulation de la consommation énergétique du réseau
- Le calcul de la durée de vie maximale du réseau

La soutenance orale du premier semestre marque la fin de la première étape du projet et le début de la seconde. Celle-ci a commencé par le rattrapage du léger retard pris sur la recherche technique dans l'état de l'art. En effet, il était prévu d'effectuer ces tâches en parallèle du cahier des spécifications, comme visible sur le diagramme de Gantt Figure 1, mais la recherche a nécessité d'avantage d'heures qu'initialement prévu.

Tout au long du développement trois phases critiques ont été rencontrées : les phases de conception détaillées de chaque lot. Si le détail n'est pas assez profond le risque est de passer plus de temps que prévu sur le développement ou sur les tests et de retarder l'ensemble du projet.

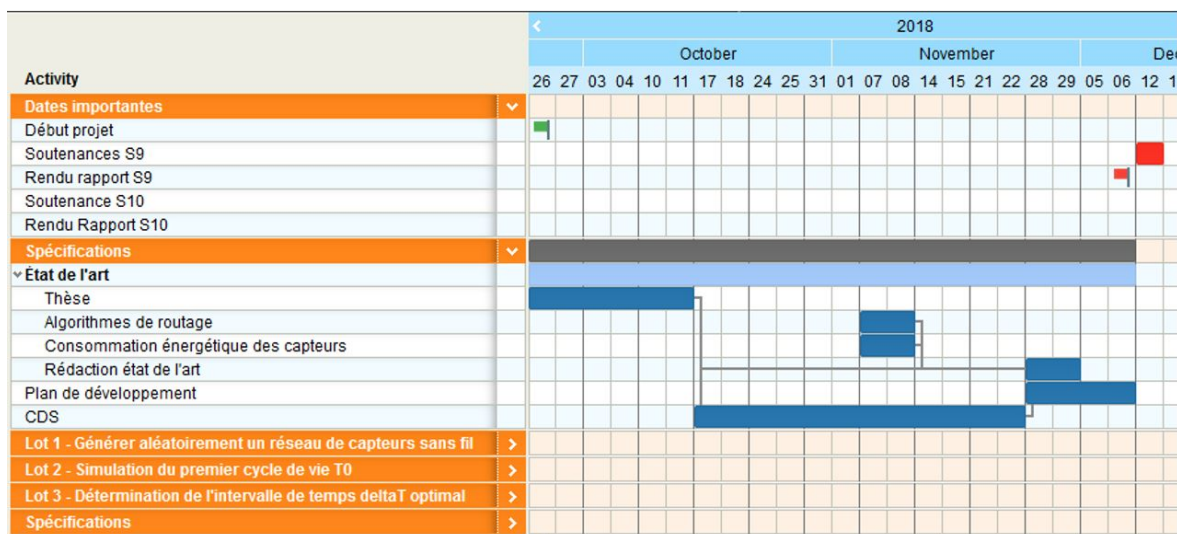


Figure 1 – Retroplanning du premier semestre. (Source : Thibaud Beaufiles)



Figure 2 – Retroplanning du second semestre. (Source : Thibaud Beaufiles)

15

Bilan sur le développement

La totalité des fonctionnalités attendues ont été développées, l'utilisateur peut :

- Générer un réseau aléatoire à partir de paramètres saisis
- Lancer la simulation du réseau
- Visualiser les résultats afin de :
 - Déterminer la durée de vie maximale du réseau
 - Déterminer le routage du réseau à un moment donné
 - Déterminer l'intervalle de temps optimal entre chaque changement de rôle
- Exporter ou importer des résultats obtenus

Cependant, un retard d'un mois pris lors de la phase de développement du premier lot a eu certaines répercussions sur le projet. Ainsi les tests n'ont pas pu être écrit et réalisés en parallèle du développement. Cette étape a été reportée à la fin ce qui a influencé le taux de couverture des tests et donc, la robustesse du programme.

De plus, il aurait été nécessaire d'allouer plus de temps à la phase d'analyse des résultats afin de pouvoir émettre des hypothèses sur les améliorations algorithmiques possibles à effectuer. Le tout dans l'objectif de réduire le temps d'exécution de la simulation et de permettre de générer des solutions pour des cas de graphes plus grands.

Outre le problème de performance à résoudre, quelques pistes d'améliorations sont envisageables concernant le logiciel.

- Il serait intéressant de proposer à l'utilisateur de saisir les paramètres de la simulation de la même manière qu'il lui est demandé de saisir ceux de la génération.
- Permettre à l'utilisateur d'annuler une action en cours (simulation ou génération).

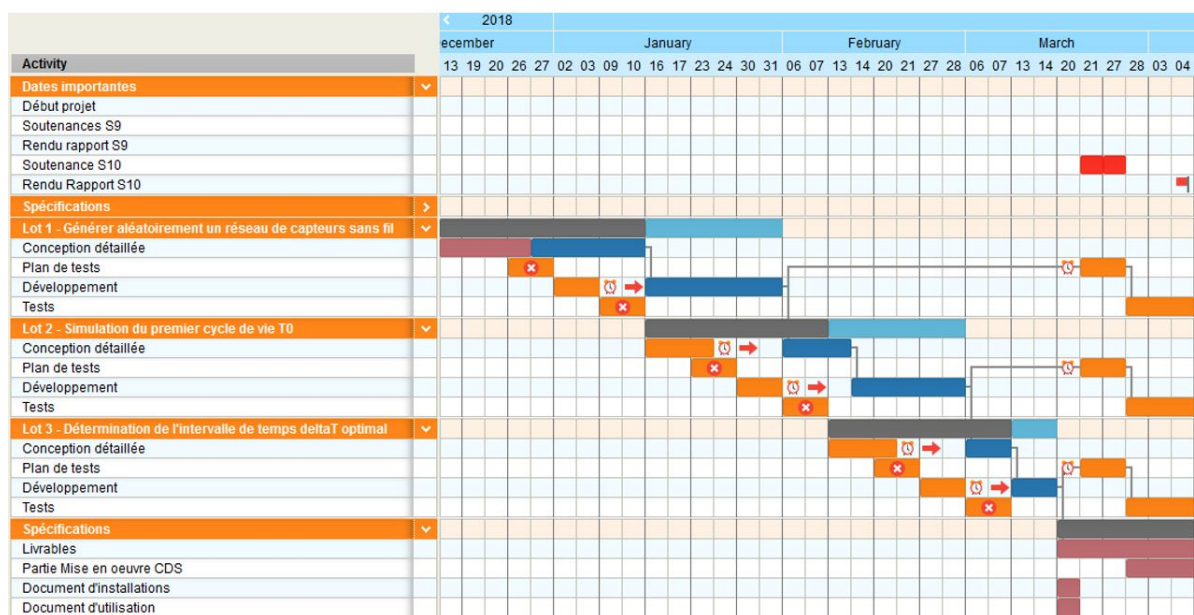


Figure 1 – Planning du second semestre. En orange avec une croix les étapes retardées puis reportées à la fin du développement. En horaire avec une horloge fléchée les étapes qui ont été reportées. En bleue les étapes du projet reportées telles qu'elles se sont déroulées. (Source : Thibaud Beaufils)

16

Conclusion

Ce projet m'a permis d'être dans les conditions d'un projet professionnel de longue durée, première dans mon expérience. Il a donc été intéressant de mettre en place les connaissances que j'ai acquies pendant ces trois années d'études et découvrir que, malgré mes acquis, il me reste encore du chemin à parcourir avant d'arriver à un niveau de maîtrises des outils à ma disposition.

J'ai notamment pu remarquer qu'un retard peu avoir de lourdes conséquences sur un projet et qu'il est parfois indispensable de travailler en dehors des horaires aménagés. Si je devais reprendre ce projet de recherche et développement depuis le début j'envisagerais moins de fonctionnalités et prendrais davantage le temps de valider mon avancement auprès du client afin de progresser sur des bases plus solides.

ANNEXE 1 : Spécifications Non Fonctionnelles

1 Contraintes de développement et conception

Le logiciel est développé sous le système d'exploitation Windows 10.

Le langage utilisé est le langage python 3.7 avec l'IDLE PyCharm. Le détail des bibliothèques utilisés est détaillé dans le guide d'installation.

2 Contraintes de développement et conception

2.1 Performances et capacités

Étant donné qu'il s'agit d'une simulation, sa durée doit ne pas dépasser la dizaine de minutes, et ce quelle que soit la taille de la topologie, mais l'intervalle de temps ΔT ou la durée de vie du réseau T ne doivent pas avoir d'influence sur le réseau, hormis le fait que le nombre d'itération soit plus grand.

En ce qui concerne l'espace de stockage nécessaire, il est de l'ordre du mégaoctet afin de pouvoir enregistrer l'ensemble des états du réseau lors de la simulation, ce qui correspond à un nombre important d'états du réseau.

2.2 Contrôlabilité

Comme évoqué précédemment, chaque étape d'une action est référencée dans un fichier de log. A chaque erreur survenue, l'utilisateur est invité à visualiser le fichier. Une fois la simulation lancée, l'utilisateur doit attendre la fin de son déroulement. Cependant, chaque état à chaque instant est stocké dans un fichier, l'utilisateur peut donc revenir à un instant demandé.

2.3 Sécurité

ANNEXE 2 : Spécifications fonctionnelles

1 Contrôler les paramètres - *controleParametres*

Cette fonction fait office de contrôleur, elle permet de s'assurer que les paramètres saisis par l'utilisateur correspondent bien à ceux attendus.

Entrées : La liste des paramètres à tester. *Sorties* : La confirmation ou non que les paramètres soient corrects.

2 Fonction Génération d'un réseau - *generationReseau*

La fonction a pour objectif de générer aléatoirement un réseau de capteurs sans fil à partir des paramètres saisis par l'utilisateur. La fonction commence par générer aléatoirement un graphe puis lui donne le sens de réseau.

Une exception est levée si les entrées ne correspondent pas aux critères demandés.

2.1 Entrées

Ensemble de paramètres, on retrouve cinq nombres entiers :

- La capacité de la batterie des capteurs
- L'intervalle de distance entre les capteurs (borne inférieure et borne supérieure)
- Le nombre de capteurs
- La taille de la surface à couvrir

Sorties : Le réseau généré. *Préconditions* : Les nombres ont des valeurs cohérentes. *Postconditions* : Un réseau de capteurs connexe a été généré.

3 Fonction sauvegarde réseau - *sauvegardeReseau*

La fonction sauvegarde le réseau passé en paramètre dans un fichier XML, unique, écrasé à chaque nouvel enregistrement.

3.1 Entrées

Le réseau à sauvegarder, le chemin vers lequel enregistrer le réseau. *Sorties* : La validation, ou non, du bon déroulement de la sauvegarde. *Préconditions* : Le réseau transmis est correcte. *Postconditions* : Le réseau a été enregistré.

4 Fonction charger réseau - *chargerReseau*

La fonction charge le réseau depuis le fichier XML unique prévu à cet effet
Une exception est levée si aucun fichier n'est trouvé.

4.1 Entrées

Le chemin vers le fichier contenant les informations du réseau *Sorties* : Le réseau créé à partir des données contenues dans le fichier XML *Postconditions* : Le réseau a été créé et retourné

5 Fonction de détermination de l'intervalle de temps - *determinerIntervalleTemps*

La fonction a pour objectif de déterminer l'intervalle de temps ΔT entre chaque changement de rôle des capteurs (émetteur ou émetteur/récepteur).

Voir le chapitre Analyse et Conception pour plus de détails sur la détermination de ΔT .

Sorties : Le nouvel intervalle de temps. *Postconditions* : L'intervalle de temps est inférieur à la durée de vie.

6 Détermination de la configuration topologique - *configurationTopologique*

La fonction détermine quelle sera le prochain rôle de chacun des capteurs en fonction de leurs anciens rôles et de leur énergie de sorte que chacun des capteurs émetteurs puisse se connecter à un émetteur/récepteur et que chaque émetteur/récepteur soit à la portée d'un autre ou de la passerelle. La fonctionnalité *ensembleDominant* est utilisée pour déterminer un sous-graphe de capteurs émetteur/récepteur connexe.

6.1 Entrées

Le réseau à configurer et l'ensemble dominant. *Sorties* : Le réseau avec les rôles modifiés
Préconditions : L'ensemble dominant a été déterminé à partir du réseau passé en paramètres
Postconditions : L'ensemble des capteurs émetteurs/récepteurs couvre la totalité du réseau, c'est-à-dire que chaque capteur émetteur est relié à l'ensemble dominant ou au nœud puit.

7 Détermination de l'ensemble dominant - *ensembleDominant*

Détermine l'arbre dominant de capteurs récepteur/émetteur dont la racine est la passerelle.

7.1 Entrées

Le réseau *Sorties* : Graphe, arbre dominant de l'ensemble des capteurs/récepteur dont la racine est la passerelle. *Postconditions* : L'ensemble dominant est un arbre de racine la passerelle

8 Détermination du routage des données - *routage*

Détermine le routage des données sur l'ensemble du réseau, elle crée un lien entre chaque capteur et l'ensemble dominant.

8.1 Entrées

Le réseau et son arbre dominant. *Sorties* : Le réseau avec les instructions de routage stockées dans les nœuds. *Préconditions* : L'arbre dominant correspond au réseau. *Postconditions* : Chaque nœud sait vers quel nœud envoyer ses données.

9 Lancer la simulation - *simulation*

Cette fonction permet de simuler la consommation électrique de l'ensemble du réseau sur l'intervalle de temps ΔT .

Voir le chapitre Analyse et Conception pour plus de détails sur le déroulement de l'algorithme.

9.1 Entrées

Le réseau et l'intervalle de temps. *Sorties* : La liste des nœuds qui n'ont plus d'énergie. *Préconditions* : Le routage du réseau a été déterminé et stocké dans ses nœuds. *Postconditions* : L'énergie totale du réseau a diminué proportionnellement à son utilisation.

10 Simulation de la consommation énergétique - *consommationEnergie*

La fonction simule la consommation énergétique du réseau pour un instant donné et met à jour le niveau de batterie de chaque capteur. L'algorithme déduit la consommation chaque récolte, réception et transmission d'information.

10.1 Entrées

Le réseau. *Sorties* : Le réseau modifié. *Postconditions* : L'énergie totale du réseau a diminué proportionnellement à son utilisation

11 Sauvegarde du résultat de la simulation - *sauvegardeResultats*

Cette fonction permet d'ajouter dans plusieurs fichiers XML l'ensemble des états dans lequel le réseau est passé pendant la simulation

11.1 Entrées

Le chemin du dossier où copier les états *Sorties* : La validation, ou non, du bon déroulement de la sauvegarde.

12 Chargement du résultat d'une simulation - *chargerResultats*

Cette fonction permet de charger depuis un dossier contenant un ensemble de fichier XML l'ensemble des états dans lequel le réseau est passé pendant la simulation

12.1 Entrées

Le chemin du dossier où récupérer les états *Sorties* : La validation, ou non, du bon déroulement du chargement.

13 Afficher le réseau - *afficherReseau*

Cette fonction permet d'afficher le réseau dans l'interface graphique tel qu'il est dans son état au moment de l'appel à la fonction.

Se référer au Guide d'utilisation pour plus de détails sur l'affichage.

13.1 Entrées

Le réseau. *Postconditions* : Le réseau s'affiche.

14 Afficher les statistiques du réseau - *afficherStatistiques*

Cette fonction permet d'affiche dans deux graphiques les statistiques relatives aux résultats obtenus lors de la simulation.

Postconditions : Toutes les informations s'affichent correctement

15 Détermination de la fin de vie du réseau - *determinationFinDeVie*

La fonction détermine, en fonction du nombre de capteurs reliés à la passerelle par rapport au nombre total de capteurs, si le capteur a atteint sa fin de vie. Pour cela il compare ce ratio avec un seuil fixé à 10

Entrées : Le réseau et la liste des nœuds qui n'ont plus d'énergie. *Sorties* : Vrai si la durée de vie a été atteinte, Faux sinon.

16 Comparaison des durées de vie - *comparaisonDureeDeVie*

La fonction détermine si la nouvelle durée de vie est plus longue que l'ancienne meilleure.

16.1 Entrées

Les deux couples : durée de vie/intervalle de temps ($T, \Delta T$), précédent et actuel. *Sorties* : Vrai si la durée de vie est meilleure, Faux sinon, ainsi que le couple avec la durée de vie la plus longue.

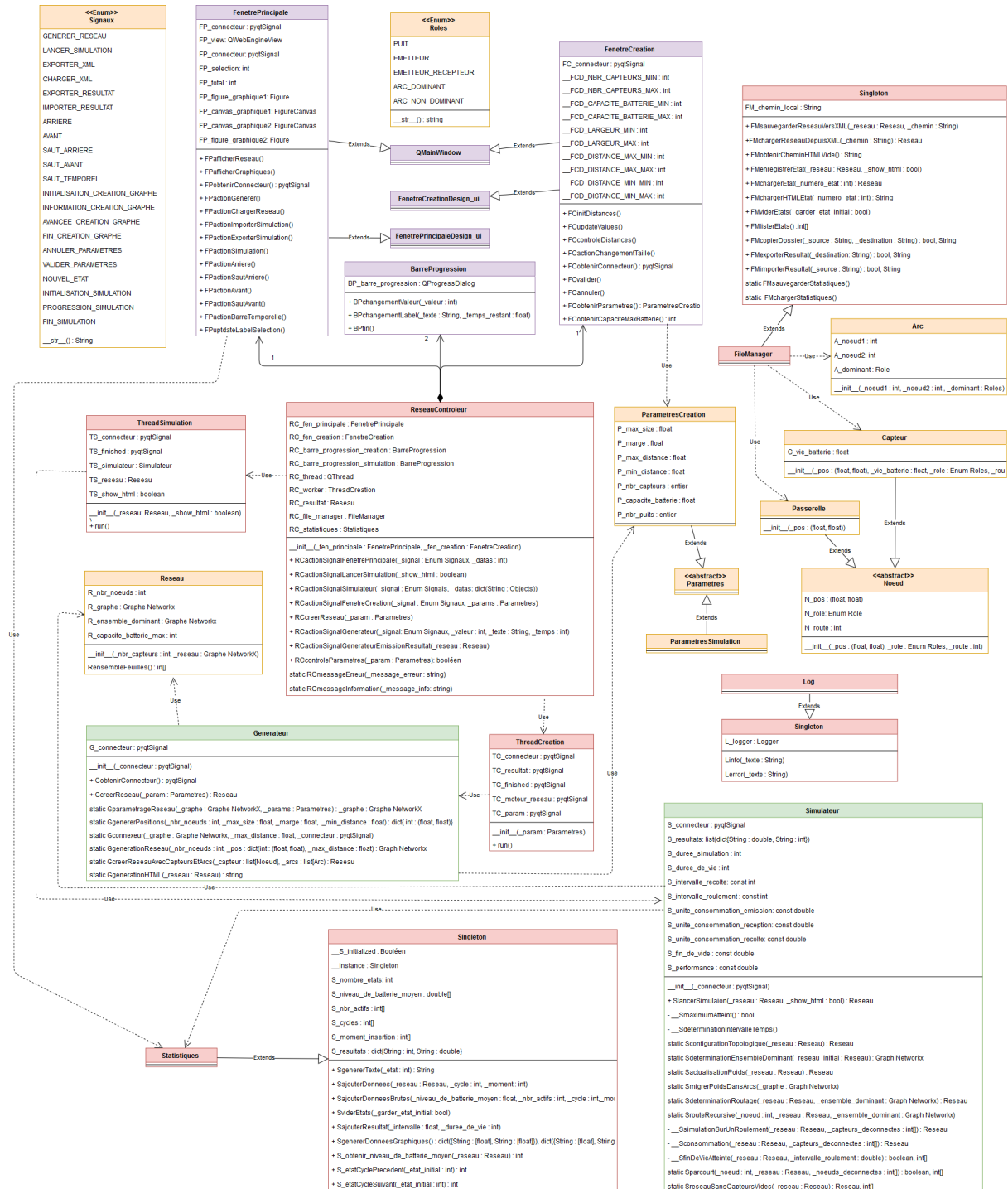
17 Récupération de la liste des états du réseau - *listeEtats*

Permet de récupérer le nombre d'états dans lequel est passé le réseau

Sorties : Le nombre d'états



ANNEXE 3 : Diagramme de classes



ANNEXE 4 : Évaluation des risques

		Gravité			
		1 Faible	2 Moyenne	3 Grave	4 Très Grave
Probabilité	4 Très probable			La simulation est beaucoup plus lente que la durée souhaitée	
	3 Probable			Compréhension des algorithmes plus complexes que prévu	
	2 Improbable	Arrêt maladie de l'enseignant encadrant	Arrêt maladie de l'étudiant		Perte ou endommagement des données en ligne
	1 Très improbable				Perte des données en ligne et de la sauvegarde

Dispositifs mis en place :

- Perte des données en ligne : sauvegarde des données sur un disque externe
- Complexité de l'algorithme plus grande prévue : travail de recherche finit avant implémentation
- La simulation est beaucoup plus lente que la durée souhaitée : allocation d'un temps dans le plan de développement pour des tests d'optimisation



ANNEXE 5 : Guide d'installation

V 1.0

Polytech Tours

Simulateur de réseaux de capteurs dynamiques

Guide d'installation



Beaufils Thibaud
Projet R&D 2019



Table des matières

I – Installation utilisateur.....	2
II – Installation développeur.....	3
1. Environnement de développement	3
2. Bibliothèques à importer.....	3
3. Utilisation de Qt	5

Commencer en téléchargeant la version 3.7 de Python disponible en suivant le lien suivant :



<https://www.python.org/downloads/release/python-370/>

Pendant l'installation, sélectionner l'option « Ajouter aux variables d'environnement ».

I – Installation utilisateur

La suite du guide d'installation utilisateur sera illustré sous l'environnement Windows.

Une fois Python installé, ouvrir un Invite de Commandes et y saisir « python ». Un résultat similaire à la figure suivant devrait apparaître. Vérifier que la version est bien de la forme 3.7.x.

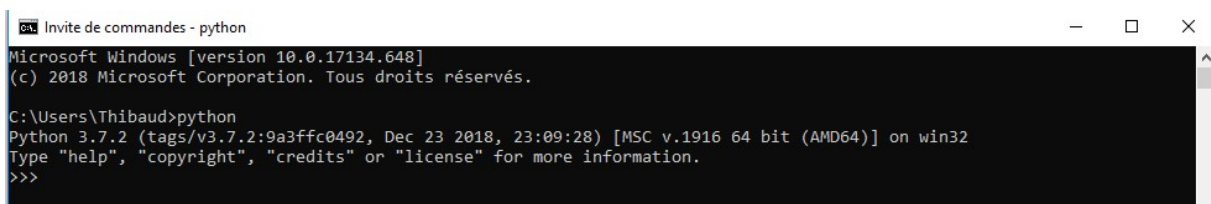
A screenshot of a Windows Command Prompt window titled "Invite de commandes - python". The window shows the output of the 'python' command: "Microsoft Windows [version 10.0.17134.648] (c) 2018 Microsoft Corporation. Tous droits réservés. C:\Users\Thibaud>python Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit (AMD64)] on win32 Type "help", "copyright", "credits" or "license" for more information. >>>". The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

Figure 1: Confirmation de l'installation de Python

Décompresser dans un nouveau dossier le contenu de l'archive nommée « Simulateur de Reseaux de Capteurs Dynamique.zip ».

Double cliquer sur le fichier « Installation.bat » pour installer les dépendances de l'application. « ===== Installation terminée ===== » doit s'afficher et chaque bibliothèque doit s'être installée correctement (c'est-à-dire que aucune ligne rouge ne doit apparaître).

Enfin, lancer l'application en cliquant sur le fichier « Simulateur de Reseaux de Capteurs Dynamique – Démarrer.bat ». Se référer au Guide d'Utilisation pour plus d'informations sur le fonctionnement de l'application.

II – Installation développeur

1. Environnement de développement

Pour des conditions optimales de développement, il est conseillé d'utiliser le même IDE que lors du développement de l'application. La figure suivante présente les caractéristiques de celui-ci. A noter que la version professionnelle de PyCharm ferait tout autant l'affaire.



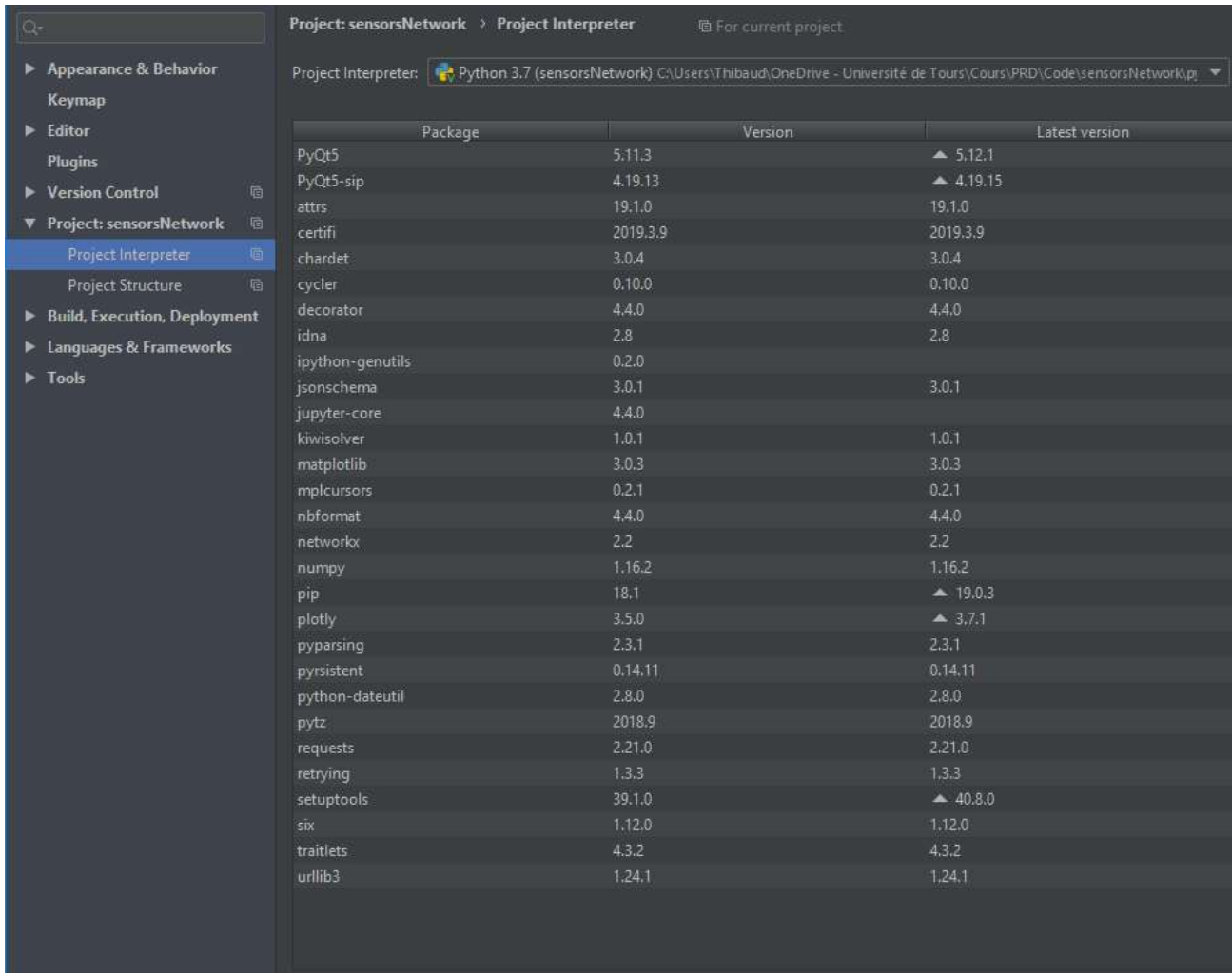
Figure 2 : Version de PyCharm utilisé lors du développement

2. Bibliothèques à importer

Les versions indiquées sont celles utilisées lors du développement. Il est conseillé d'utiliser les dernières versions mais d'utiliser celle-ci en cas d'incompatibilités.

plotly	3.5
PyQt5	5.11.3
PyQt-sip	4.19.13
networkx	2.2
mplcursors	0.2.1
matplotlib	3.0.3

Une fois l'ensemble de ces bibliothèques installées, elles-mêmes ayant installées des bibliothèques, la liste exhaustive des dépendances correspond à la suivante :



The screenshot shows the 'Project Interpreter' window in PyCharm for the project 'sensorsNetwork'. The window displays a table of installed packages, their current versions, and the latest available versions. The left sidebar shows the project structure with 'Project Interpreter' selected. The top bar indicates the interpreter is 'Python 3.7 (sensorsNetwork)' located at 'C:\Users\Thibaud\OneDrive - Université de Tours\Cours\PRD\Code\sensorsNetwork\p...'. The table lists 30 packages, including PyQt5, attrs, certifi, chardet, cyclur, decorator, idna, ipython-genutils, jsonschema, jupyter-core, kiwisolver, matplotlib, mplcursors, nbformat, networkx, numpy, pip, plotly, pyparsing, pypersistent, python-dateutil, pytz, requests, retrying, setuptools, six, traitlets, and urllib3.

Package	Version	Latest version
PyQt5	5.11.3	▲ 5.12.1
PyQt5-sip	4.19.13	▲ 4.19.15
attrs	19.1.0	19.1.0
certifi	2019.3.9	2019.3.9
chardet	3.0.4	3.0.4
cyclur	0.10.0	0.10.0
decorator	4.4.0	4.4.0
idna	2.8	2.8
ipython-genutils	0.2.0	
jsonschema	3.0.1	3.0.1
jupyter-core	4.4.0	
kiwisolver	1.0.1	1.0.1
matplotlib	3.0.3	3.0.3
mplcursors	0.2.1	0.2.1
nbformat	4.4.0	4.4.0
networkx	2.2	2.2
numpy	1.16.2	1.16.2
pip	18.1	▲ 19.0.3
plotly	3.5.0	▲ 3.7.1
pyparsing	2.3.1	2.3.1
pypersistent	0.14.11	0.14.11
python-dateutil	2.8.0	2.8.0
pytz	2018.9	2018.9
requests	2.21.0	2.21.0
retrying	1.3.3	1.3.3
setuptools	39.1.0	▲ 40.8.0
six	1.12.0	1.12.0
traitlets	4.3.2	4.3.2
urllib3	1.24.1	1.24.1

Figure 3 : Ensemble des bibliothèques nécessaires au projet

3. Utilisation de Qt

L'interface utilisateur (.ui) est à modifier avec Qt Creator.

L'illustration suivante indique quelle version a été utilisée lors du développement.



Figure 4 : Version de Qt Creator utilisée lors du développement

Pour compiler les fichiers .ui en fichier .py exploitable il faut s'assurer de posséder le script python pyuic5.exe (le télécharger sinon) et exécuter la liste de commande suivante :

```
> "chemin_absolu_vers_le_script\pyuic5.exe" fichier_a_compiler.ui -o fichier_compile_ui.py
```



ANNEXE 6 : Guide d'utilisation

V 1.0

Polytech Tours

Simulateur de réseaux de capteurs dynamiques

Guide d'utilisation



Beaufils Thibaud
Projet R&D 2019



Table des matières

Table des figures.....	1
Présentation de l'interface principale	2
I – Créer un réseau	4
II – Exporter un réseau	5
III – Importer un réseau.....	6
IV – Lancer une simulation	6
V – Sauvegarder le résultat d'une simulation	6
VI – Importer le résultat d'une simulation	7
VII – Interpréter le résultat d'une simulation.....	8

Table des figures

Figure 1 : Fenêtre Principale	2
Figure 2 : Le réseau vers la fin de sa vie	3
Figure 3 : Exemple d'affichage d'un réseau après sa génération	3
Figure 4 : Décomposition de la section D	3
Figure 5 : Fenêtre de paramétrage pour la création d'un réseau	4

Présentation de l'interface principale

Lors de l'ouverture de l'application, la première fenêtre se présente ainsi :

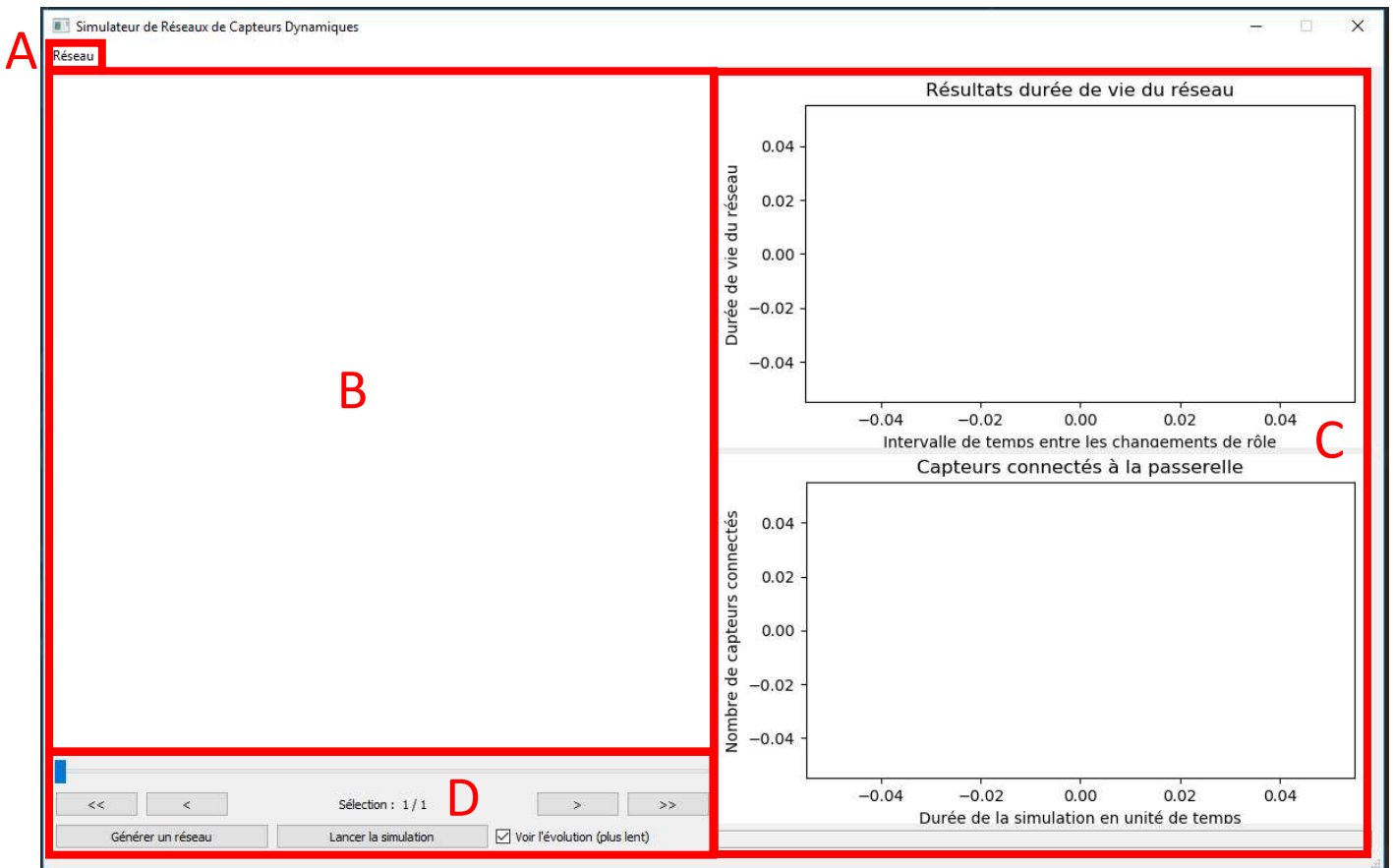


Figure 1 : Fenêtre Principale

A. Menu

Ce menu rend accessible quatre actions :

- L'exportation au format .XML du réseau affiché (cf. II)
- L'importation d'un réseau depuis le format .XML (cf. III)
- L'exportation dans un dossier du résultat de la simulation (cf. V)
- L'importation d'un résultat de simulation depuis un dossier (cf. VI)

B. Affichage du réseau

Cette section affiche le réseau dans un certain état. Le choix de l'état se fait dans la section partie D.

La Figure 3 présente un graphique tel qu'il peut être affiché. Au passage de la souris les informations suivantes sont affichées :

- Numéro du capteur
- Niveau de la batterie
- Numéro du capteur vers lequel transmettre ses données (route)

La couleur intérieure du capteur, associée à l'échelle à droite du graphe, correspond à son niveau de batterie. Les capteurs entourés en rouge sont les capteurs qui ont le rôle d'émetteur et de récepteur. Les autres capteurs ont seulement le rôle d'émetteur. Les arcs reliant les capteurs entre eux sont bleus sauf s'ils relient des capteurs avec le rôle d'émetteur / récepteur.

Le nœud pentagonal jaune représente la passerelle (ou puit). Toutes les données récoltées puis envoyés par les capteurs sont acheminées vers ce nœud.

Comme l'illustre la Figure 2, un capteur qui ne possède plus d'énergie est représenté en noir. Les nœuds qui ne sont plus reliés à la passerelle sont représentés en gris ainsi que les arcs reliant ces nœuds.

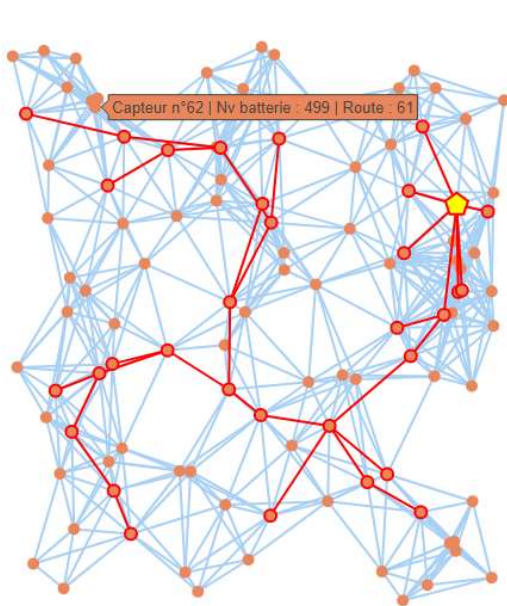


Figure 3 : Exemple d'affichage d'un réseau après sa génération

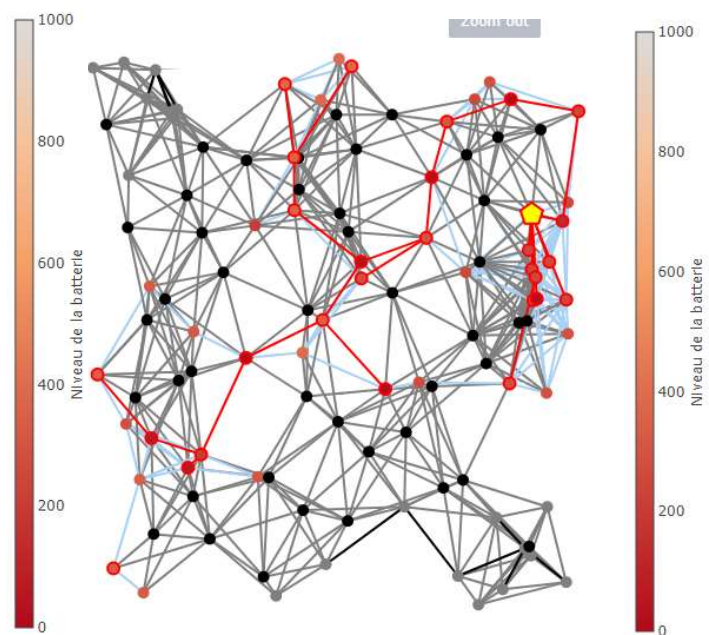


Figure 2 : Le réseau vers la fin de sa vie

A l'aide des différentes options accessibles sur la partie supérieure droite de la section, il est possible de zoomer et dézoomer sur la figure.

C. Graphiques et données

Se trouve dans cette section les graphiques représentant les résultats obtenus lors de la simulation. Leur interprétation est détaillée partie VII.

D. Options

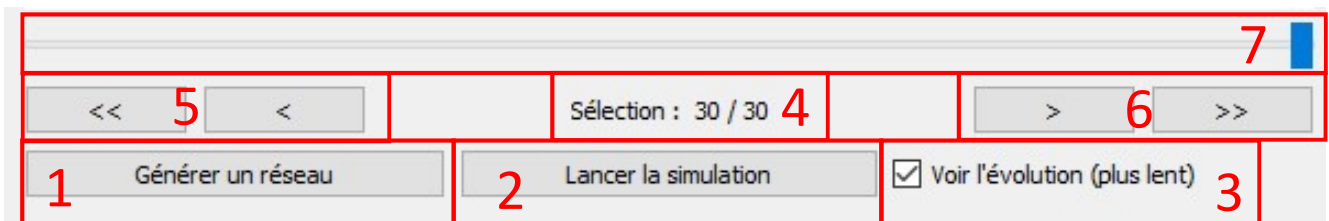


Figure 4 : Décomposition de la section D

Cette section rend accessible les actions suivantes :

1. Créer un réseau (cf I)
2. Lancer la simulation (cf IV)
3. Choisir si les étapes intermédiaires dans lequel passe le graphe doivent être affichées durant la simulation
4. Visualiser l'étape de la simulation sélectionnée et le nombre d'états transitoires dans lequel le réseau est passé.
5. Visualiser l'état précédent ou visualiser le premier état du cycle précédent.
6. Visualiser l'état suivant ou visualiser le premier état du cycle suivant.
7. Choisir manuellement quel état visualiser en utilisant la glissière.

I – Créer un réseau

La fenêtre de paramétrage de la création du réseau est accessible après avoir cliqué sur le bouton 1.

The screenshot shows a window titled "Paramétrage du réseau à générer" with a subtitle "Veuillez saisir les paramètres nécessaires". It contains several sliders and input fields for configuring network parameters:

- Nombre de capteurs sans fils à déployer :** A slider ranging from 3 to 122, with the current value set to 110.
- Capacité de la batterie des capteurs (Ampère.heure) :** A slider ranging from 1 to 1000, with the current value set to 499.
- Largeur de la zone carrée à couvrir :** A slider ranging from 10 to 500, with the current value set to 245.
- Distance à partir de laquelle deux capteurs peuvent établir une connexion :** A slider ranging from 1 to 49, with the current value set to 49.
- Distance minimum à respecter entre deux capteurs :** A slider ranging from 0 to 5, with the current value set to 2.

At the bottom of the window, there are two buttons: "Annuler" and "Générer le réseau".

Figure 5 : Fenêtre de paramétrage pour la création d'un réseau

Elle comporte les paramètres suivants :

- Le nombre de capteurs à inclure dans le réseau
- La capacité de la batterie des capteurs
- La largeur de la surface carrée à couvrir
- La distance maximale pour que deux capteurs établissent une connexion
- La distance minimum que doivent respecter deux capteurs entre eux

Attention Créer un nouveau réseau effacera l'ancien ainsi que le résultat de la simulation précédente. Il est conseillé d'exporter vos résultats préalablement.

II – Exporter un réseau

L'exportation de l'état d'un réseau est possible à travers le menu A. Un dossier où exporter le fichier est demandé à l'utilisateur. Uniquement le format XML est possible.

La structure du fichier est la suivante :

```
<reseau> <- Balise contenant l'ensemble des données du réseau
  <meta> <- Contient des données diverses
    <nbrnoeuds> <- Le nombre de nœuds dans le graphe
    </nbrnoeuds>
    <capbatteriemax> <- La capacité de batterie maximale des capteurs
    </capbatteriemax>
  </meta>
  <graphe> <- Contient les nœuds et les arcs
    <noeuds> <- Contient les nœuds
      <noeud> <- Un noeud
        <numero></numero> <- Le numéro du nœud
        <role></role> <- Le rôle du nœud
        <batterie></batterie> <- Le niveau de sa batterie
        <posx></posx> <- Sa position absolue en abscisse
        <posy></posy> <- Sa position absolue en ordonnée
        <route></route> <-Le nœud vers lequel envoyer ses données
      </noeud>
      ...
    </noeuds>
    <arcs> <- Contient les arcs
      <arc> <- Un arc
        <noeud1></noeud1> <- Le premier nœud de l'arc
        <noeud2></noeud2> <- Le second nœud de l'arc
        <dominant></dominant> <- Si l'arc est de l'ensemble
dominant
      </arc>
      ...
    </arcs>
  </graphe>
</reseau>
```

III – Importer un réseau

A travers le menu A il est possible de charger l'état d'un réseau depuis un fichier .XML dont la structure est détaillée en II.

Pour que l'importation soit une réussite, les caractéristiques suivantes doivent être respectées :

- Le nombre de nœuds doit être un entier positif et doit correspondre au nombre de balises « nœud »
- Dans cette version la passerelle doit être unique
- La capacité de la batterie doit être un entier positif ou -1 dans le cas d'un puit
- Chaque numéro de nœud doit être différent
- Le code du rôle doit être une des valeurs de la liste suivante :
 - « 0 » pour un nœud puit
 - « 1 » pour un nœud émetteur
 - « 2 » pour un nœud émetteur/récepteur
- Ses positions doivent être des réels positifs
- Sa route doit correspondre à un numéro de nœud existant ou son propre numéro pour un puit
- Les numéros des nœuds des arcs doivent correspondre à des nœuds existants.
- Le code de la balise « dominant » doit être une des valeurs de la liste suivante :
 - « 3 » pour un arc dominant
 - « 4 » pour un arc non dominant

IV – Lancer une simulation

Le lancement de la simulation se fait en cliquant sur le bouton 3. Cocher la case à droite du bouton prendre plus de temps mais permet de voir l'évolution du réseau en même temps que la simulation puis de revoir les états après la simulation.

Une fois la simulation lancée une barre de chargement apparaît. Y est indiqué l'intervalle de temps utilisé entre chaque changement de rôle. La valeur de la barre de chargement correspond au nombre de capteurs reliés à la passerelle. Elle se réinitialise à chaque nouveau cycle, c'est-à-dire à chaque nouvel intervalle de temps utilisé.

V – Sauvegarder le résultat d'une simulation

Une fois une simulation effectuée, il est possible d'en exporter le résultat, c'est-à-dire l'ensemble des états par lesquels est passé le réseau lors de la simulation en plus d'un fichier contenant les données utilisées pour l'affichage des graphiques.

Le nom des fichiers contenant les états est de la forme « etatX.xml » avec X numéro unique d'un l'état. Pour chaque état une version html y est également, utilisée pour afficher l'état dans l'application. Le contenu des fichiers est le même que celui présenté en II.

La structure du fichier contenant les données statistiques est la suivante :

```
<statistique> <- Balise contenant l'ensemble des statistiques de la simu.
  <nbretats> <- Contient le nombre d'état dans lequel est passé le rés.
  <nbrresultats> <- Contient le nombre de résultats=le nombre de cycles
  <etats> <- Contient l'ensemble des données des états
    <etat> <- Contient les données d'un état
      <numero_etat> <- Le numéro de l'état
      </numero_etat> <- Niveau moyen de la batterie des capteurs à
cet état
      <niveau_de_batterie_moyen>
      </niveau_de_batterie_moyen>
      <nbr_actifs> <- Nbr de capteurs reliés à la passerelle
      </nbr_actifs>
      <moment_insertion> <- Moment de création de l'état
      </moment_insertion>
      <cycle> <- Cycle au moment de l'insertion de l'état
      </cycle>
    </etat>
    <etat>
      ...
    </etat>
    ...
  </etats>
  <resultats> <- Contient l'ensemble des résultats obtenus
    <resultat> <- Un résultat
      <intervalle></intervalle> <-L'intervalle de temps utilisé
      <dureedevie></dureedevie> <-La durée de vie calculée associée
    </resultat>
  </resultats>
</statistique>
```

VI – Importer le résultat d'une simulation

Il est possible d'importer le résultat d'une simulation en sélectionnant le dossier tel qu'il a été exporté. Le dossier ne doit contenir aucun autre fichier et, en plus du format des états évoqué en II, le format des statistiques doit respecter les éléments suivants :

- Les numéros d'états doivent correspondre à des fichiers dans le dossier et sont uniques
- Le niveau moyen de batterie doit être un réel positif
- Le moment d'insertion est un naturel
- Le cycle doit être un nombre naturel
- L'intervalle est un réel positif
- La durée de vie est un naturel

VII – Interpréter le résultat d’une simulation

La fin de vie du réseau est considérée atteinte seulement quand au plus 20% des capteurs sont connectés à la passerelle. Ce ratio est visible dans la dernière ligne de la petite section relative aux informations du réseau affiché.

Le premier graphique s’actualise à chaque nouveau cycle, c’est-à-dire à la fin de chaque cycle de simulation. La simulation d’arrête lorsque aucun meilleur résultat n’est trouvé ou quand les résultats ne varient pas plus de 10%.

Le second graphique s’actualise plus fréquemment : à chaque changement de rôle des capteurs. Il est possible d’y retrouver visuellement l’ensemble des cycles comme illustré sur la figure 6 : chaque cycle est délimité dans un rectangle rouge.

Au passage de la souris les valeurs des points du graphique sont affichés et permet de relever le meilleur résultat.

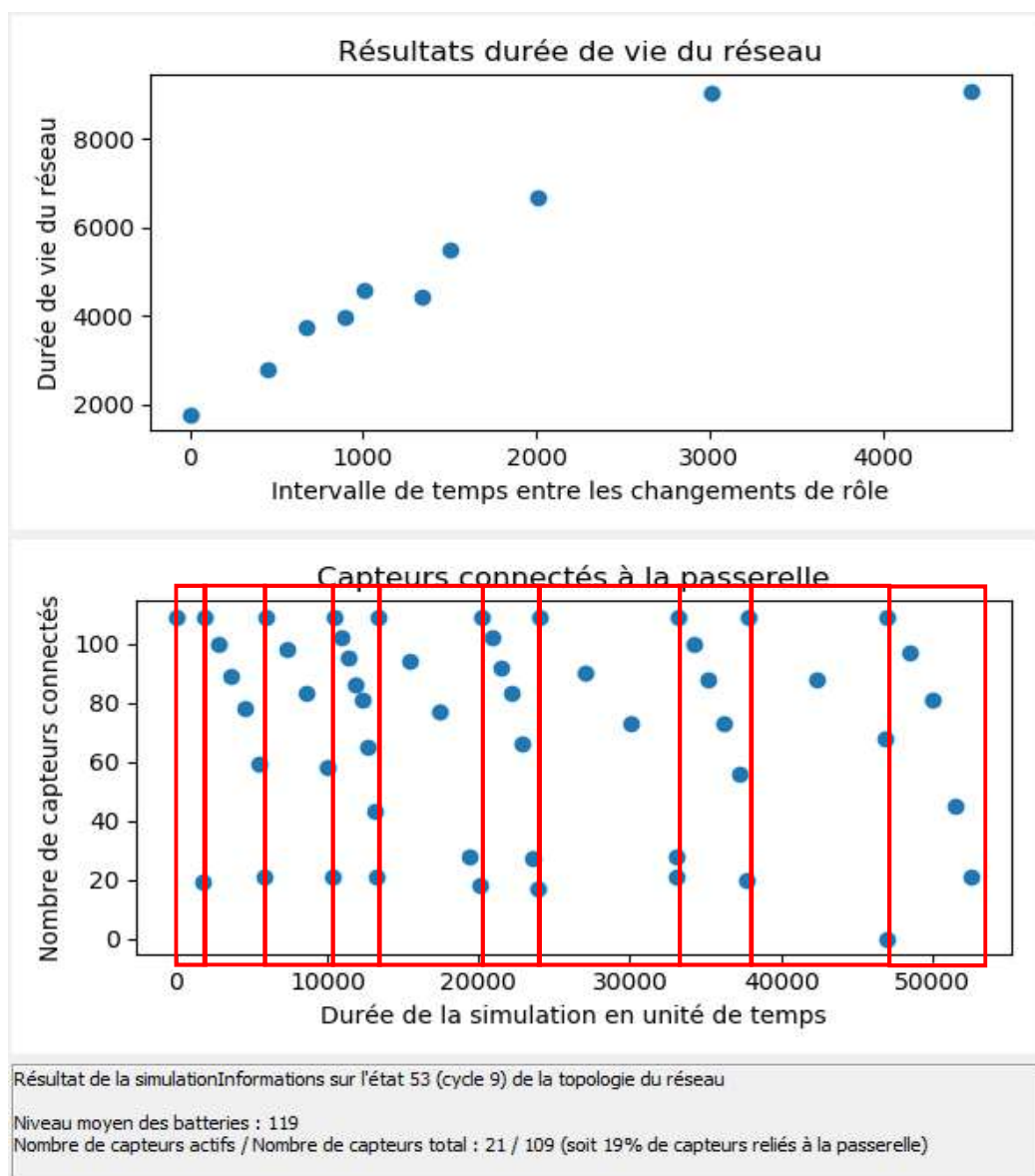


Figure 6 : Exemple de résultats obtenus après une simulation



ANNEXE 7 : Cahier de tests

ÉCOLE POLYTECHNIQUE DE L'UNIVERSITÉ FRANÇOIS RABELAIS DE TOURS

Spécialité Informatique

64 av. Jean Portalis

37200 TOURS, FRANCE

Tél +33 (0)2 47 36 14 31

www.polytech@univ-tours.fr

- CAHIER DE TESTS -

Projet : Développement Algorithmique et Conception d'un Simulateur pour un Réseau de Capteurs dynamiques

Emetteur :

A. SOUKHAL

MOA : EPU-DI

Date d'émission :

20/09/2018

Validation

Nom	Date	Valide (O/N)	Commentaires
-----	------	--------------	--------------

Historique des modifications

Version	Date	Description de la modification
00	06/03/2019	Version initiale

Table des matières

- Cahier de tests -	1
I. Introduction	3
1. Objectif du document	3
2. Portée du document	3
3. Références	3
II. Exigences à tester.....	4
III. Stratégie de tests	5
1. Types de tests	5
Tests unitaires	5
Tests Fonctionnels.....	5
Tests de données et d'intégrité.....	5
Tests de performances.....	6
Tests de configuration et d'installation.....	6
2. Outils utilisés.....	6
3. Ressources	6
Matérielles	6
Humaines	7
4. Les livrables.....	7
IV. Plan de tests.....	8
1. Tests fonctionnels	8
2. Tests unitaires	15
3. Tests de données et d'intégrité	18
4. Tests de performances.....	21
5. Tests de configuration et d'installation	24

I. INTRODUCTION

1. Objectif du document

Ce document présente la procédure à utiliser pour valider et recetter les différentes parties du projet « Développement Algorithmique et Conception d'un Simulation pour un Réseau de Capteurs ». Il permet de :

- Identifier les informations existantes du projet et les composants qui doivent être testés
- Énumérer les exigences d'évaluation à haut niveau
- Recommander et décrire les stratégies de test à employer
- Identifier les ressources nécessaires et fournir une estimation de l'effort de test
- Identifier les biens livrables pour les tests

2. Portée du document

Ce plan de recette est destiné au client et tuteur du projet ainsi que tout membre souhaitant continuer à élaborer d'autres stratégies pour la résolution du problème.

3. Références

- Cahier des spécifications fonctionnelles et non fonctionnelles
- Rapport final
- Cahier du développeur
- Guide d'installation
- Guide d'utilisation

II. EXIGENCES A TESTER

La liste suivante identifie les items, cas d'utilisation, exigences fonctionnelles et non fonctionnelles qui ont été choisies pour être testés.

- Générer aléatoirement un réseau de capteurs
- Déterminer le routage du réseau pour un instant donné
- Déterminer le rôle des capteurs pour un instant donnée
- Simuler la consommation du réseau jusqu'à la fin de sa vie
- Déterminer un maximum de la durée de vie du réseau et son intervalle de temps entre chaque changement de rôle des capteurs
- Naviguer à travers les solutions proposées par l'algorithme

III. STRATEGIE DE TESTS

1. Types de tests

Tests unitaires

Stratégies pour les tests unitaires

<i>Objectifs de tests</i>	Vérifier que chaque méthode fait son travail
<i>Technique</i>	Exécuter les méthodes de chaque fichier en utilisant des données valides et non valides tout en vérifiant : <ul style="list-style-type: none">• Les résultats attendus avec des données valides.• Les messages d'erreur et d'avertissement lorsque des données non valides sont utilisées.
<i>Considérations particulières</i>	Installation préalable de pytest obligatoire. L'exécution des tests est effectuée à partir de l'IDE PyCharm, le même environnement que celui de développement

Tests Fonctionnels

Stratégies pour les tests fonctionnels

<i>Objectifs de tests</i>	Vérifier que les fonctionnalités fournissent le résultat escompté
<i>Technique</i>	Exécuter les cas d'utilisation, chaque chemin du cas ou fonction en utilisant des données valides et non valides tout en vérifiant : <ul style="list-style-type: none">• Les résultats attendus avec des données valides.• Les messages d'erreur et d'avertissement lorsque des données non valides sont utilisées.
<i>Considérations particulières</i>	

Tests de données et d'intégrité

Stratégies pour les tests de données et d'intégrité

<i>Objectifs de tests</i>	Vérifier que les méthodes d'accès aux différents fichiers fonctionnent correctement et sans corruption de données récupérées ou enregistrées
<i>Technique</i>	Appeler chaque méthode et processus d'accès aux fichiers en les alimentant chacune de données valides et de données invalides. Inspecter les fichiers (d'entrée et de log) pour vérifier que les données ont été chargées comme prévu, que toutes les opérations sur les fichiers s'exécutent normalement et réviser les données produites afin de s'assurer que les bonnes données ont été extraites et pour les bonnes raisons.
<i>Considérations particulières</i>	Les fichiers d'entrées et sorties sont des fichiers XML et des dossier contenant des fichiers XML. La syntaxe de ces fichiers est lisible dans le guide d'utilisation

Tests de performances

Stratégies pour les tests de performances

<i>Objectifs de tests</i>	Vérifier la performance d'exécution des différentes parties du projet : la simulation et la génération : <ul style="list-style-type: none">- Temps d'exécution- Ressources utilisées
<i>Technique</i>	Utiliser des valeurs des tailles de réseaux variés et chronométrer à l'aide d'une application externe pour comparer les résultats annoncés avec l'application
<i>Considérations particulières</i>	La durée moyenne imposée est de 5 minutes mais la durée réelle dépendra de la taille du réseau.

Tests de configuration et d'installation

Stratégies pour les tests de configuration et d'installation

<i>Objectifs de tests</i>	Vérifier que la cible de test fonctionne correctement avec les configurations logicielle et matérielle définies.
<i>Technique</i>	Exécuter l'installation du logiciel par script sur des machines vierges de toute installation Python
<i>Considérations particulières</i>	

2. Outils utilisés

Les outils suivants seront utilisés lors du projet :

	Outils
<i>Gestion des tests</i>	Pytest
<i>Suivi des anomalies</i>	Fichier de log
<i>Tests fonctionnels</i>	
<i>Tests de performance</i>	Temps d'exécution affiché par le logiciel, Chronomètre externe
<i>Gestion de projet</i>	Github, Gantt

3. Ressources

Matérielles

	Outils
<i>Système d'exploitation</i>	Windows 10
<i>GPU</i>	Carte graphique NVidia compatible avec le système d'exploitation

Humaines

Ressources humaines

Fonction	Nom	Responsabilités - commentaires
Concepteur de test	Beaufils Thibaud	Identifier, prioriser et implémenter les cas de test Missions : <ul style="list-style-type: none">• Générer le plan de tests• Générer le modèle de tests
Testeur	Beaufils Thibaud	Exécuter les tests Missions : <ul style="list-style-type: none">• Exécuter les tests• Journaliser les résultats• Reprendre sur les erreurs• Documenter les anomalies et les demandes de changement.
Implémenteur	Beaufils Thibaud	Implémenter et exécuter les tests unitaires / fonctionnels / Performances / Configuration et d'Installation. Missions : <ul style="list-style-type: none">• Créer des fichiers de test et les paquetages de test du modèle de test.

4. Les livrables

Livrables	Date	Type
Cahier de recette	04/04/19	Document
Tests unitaires	02/04/19	Code source Python
Tests fonctionnels	27/03/19	Document
Tests de performances	27/03/19	Tableau des statistiques

IV. PLAN DE TESTS

1. Tests fonctionnels

Test Fonctionnel : Génération d'un réseau				
Objectif		Vérifier la cohérence entre les paramètres saisis et le réseau généré		
Eléments à tester		Génération d'un réseau		
Pré requis				
Scénario				
ID	Démarche	Données	Comportement attendu	OK ?
1	Générer un réseau avec les paramètres donnés	- 5 capteurs à déployer - 200 cap batterie - 125 largeur - 20 distance max - 2 distance min	Un réseau est généré et affiché	OK
2	Vérifier que l'ensemble des paramètres saisis se retrouve dans le réseau généré		- 4 capteurs sont affichés et un puit - chaque capteur a 200 cap. de batterie	OK
3	Vérifier les valeurs des différentes statistiques		- la sélection est indiquée « 1/1 » - La section « Résultat de la simulation » indique des informations relatives à l'état 1 (cycle 0). - Le niveau moyen des batteries est indiqué à 200 - Le nombre de capteurs actifs / total est de 4/4 avec un pourcentage de 100% reliés à la passerelle.	OK
4	Vérifier les valeurs des graphiques		- Sur le graphique supérieur aucun point n'est affiché - Sur le graphique inférieur un point est affiché de coordonnées (x = 0, y = 4)	OK
Rapport de test		Testé par Thibaud BEAUFILS	Le : 27/03/19	
Résultat				
Commentaire			Approbation	
Une anomalie a été détectée et réglée : lors de la génération il manquait un capteur car la passerelle était comptabilisée en tant que telle.			Validé	

Test Fonctionnel : Sauvegarder Réseau				
Objectif		Vérifier que l'exportation d'un réseau restitue le réseau dans son intégrité		
Eléments à tester		Fonctionnalité Exporter le réseau affiché		
Pré requis				
Scénario				
Id	Démarche	Données	Comportement attendu	OK ?
1	Générer un réseau de 5 capteurs.		Le réseau est généré et possède un ensemble dominant	OK
2	Sélectionner l'exportation dans le menu Réseau / Exporter ... / ... le réseau affiché (.XML)		La fenêtre de choix de fichier s'affiche	OK
3	Sauvegarder le réseau		Le réseau est sauvegardé et apparaît dans l'explorateur de fichier windows au format .XML avec le nom saisi	OK
4	Ouvrir le fichier en question avec un éditeur de texte. Ouvrir le fichier ../donnees/reseau/resultats simulation/etat0.xml contenu dans le dossier d'installation du simulateur		Le contenu des deux fichiers est identique	OK mais le niveau de la batterie est enregistré comme un réel (200.0) au lieu d'un entier initialement (200)
Rapport de test		Testé par Thibaud BEAUFILS		Le : 27/03/19
Résultat				
Commentaire			Approbation	
			Validé	

Test Fonctionnel : Charger Réseau				
Objectif		Vérifier que l'importation d'un réseau l'intègre dans sa totalité		
Eléments à tester		Fonctionnalité Import un réseau en tant qu'état initial		
Pré requis				
Scénario				
ID	Démarche	Données	Comportement attendu	OK ?
1	Sélectionner l'importation dans le menu Réseau / Importer ... / ... un réseau en tant qu'état initial (.XML)		La fenêtre de choix de fichier s'affiche	OK
2	Sélectionner le fichier à importer	reseau_10_capteurs.xml	Le réseau est importé	OK
3	Ouvrir le fichier précédemment importé avec un éditeur de texte. Ouvrir le fichier ../donnees/reseau/resultats simulation/etat0.xml		Le contenu des deux fichiers est identique	OK
Rapport de test		Testé par Thibaud BEAUFILS		Le : 27/03/19
Résultat				
Commentaire			Approbation	
			Validé	

Test Fonctionnel : Afficher Réseau				
Objectif		Vérifier que le réseau est affiché avec les bonnes informations		
Eléments à tester		Affichage		
Pré requis				
Scénario				
Id	Démarche	Données	Comportement attendu	OK ?
1	Générer un réseau avec les paramètres donnés	- 5 capteurs à déployer - 200 cap batterie - 125 largeur - 20 distance max - 2 distance min	Un réseau est généré et affiché	OK
2	Vérifier chaque information affichée dans le simulateur. : - Le nombre de capteurs affiché (sans compter le passerelle) - Le niveau de batterie - Chaque routage - La cohérence rôle / couleur - La cohérence couleur de lien / situation de lien		Toutes les informations sont cohérentes	OK
3	Générer un réseau avec les paramètres donnés	- 10 capteurs à déployer - 100 cap batterie - 50 largeur - 8 distance max - 1 distance min	Un réseau est généré et affiché	OK
4	Refaire la deuxième étape pour ce nouveau réseau			OK
5	Lancer la simulation pour ce réseau			OK
6	Appliquer l'étape 2 pour chaque étape dans lesquels le réseau est passé lors de la simulation		Chaque étape a des informations cohérentes	KO A la fin des cycles (hors cycle 0) capteurs non reliés ne sont pas grisés
Rapport de test		Testé par Thibaud BEAUFILS	Le : 27/03/19	
Résultat				
Commentaire			Approbation	
Correction de bug à effectuer. Gravité : moindre			Non	

Test Fonctionnel : Configuration Topologique				
Objectif		Vérifier que l'assignation des rôles des capteurs est cohérente		
Eléments à tester		Les rôles des capteurs		
Pré requis				
Scénario				
Id	Démarche	Données	Comportement attendu	OK ?
1	Générer un réseau avec les paramètres donnés	- 5 capteurs à déployer - 200 cap batterie - 125 largeur - 20 distance max - 2 distance min	Un réseau est généré et affiché	OK
2	Vérifier la cohérence des rôles		Chaque nœud avec le rôle « Émetteur » est connecté à au moins un nœud ayant le rôle « Émetteur / récepteur » Chaque nœud avec le rôle « Émetteur / récepteur » a au moins un nœud « Émetteur de connecté à lui.	OK
3	Vérifier la cohérence des liens		Les nœuds émetteurs / récepteur sont tous reliés au même ensemble marqué rouge sur l'affichage et relié à la passerelle	OK
4	Lancer la simulation pour ce réseau			OK
5	Appliquer l'étape 2 et 3 pour chaque étape dans lesquels le réseau est passé lors de la simulation		Chaque étape a des informations cohérentes	OK
Rapport de test		Testé par Thibaud BEAUFILS	Le : 27/03/19	
Résultat				
Commentaire			Approbation	
			Validé	

Test Fonctionnel : Routage				
Objectif		Vérifier que chaque capteur est relié avec la passerelle		
Eléments à tester		Le routage des capteurs		
Pré requis				
Scénario				
Id	Démarche	Données	Comportement attendu	OK ?
1	Générer un réseau avec les paramètres donnés	- 10 capteurs à déployer - 200 cap batterie - 125 largeur - 20 distance max - 2 distance min	Un réseau est généré et affiché	OK
2	Vérifier le routage de chaque capteur Émetteur		Chaque émetteur est routé vers un capteur Émetteur/Récepteur	OK
3	Vérifier la route de chaque capteur Émetteur/Récepteur		Chaque route inclus uniquement des Émetteur/Récepteur et mène jusqu'à la passerelle	OK
4	Lancer la simulation puis appliquer les étapes 2 et 3 pour chaque étape du réseau dans lesquels le réseau est passé lors de la simulation		Chaque étape a des informations cohérentes	OK
Rapport de test		Testé par Thibaud BEAUFILS	Le : 27/03/19	
Résultat				
Commentaire			Approbation	
			Validé	

Test Fonctionnel : Statistiques				
Objectif		Vérifier la cohérence des résultats affichés statistiquement		
Eléments à tester		Graphiques		
Pré requis				
Scénario				
Id	Démarche	Données	Comportement attendu	OK ?
1	Générer un réseau avec les paramètres donnés	- 10 capteurs à déployer - 200 cap batterie - 125 largeur - 20 distance max - 2 distance min	Un réseau est généré et affiché	OK
2	Vérifier la cohérence des informations affichées dans la section « Résultat de la simulation » et du point associé dans le graphique « Capteurs connectés à la passerelle »		Sont cohérents : Le numéro de l'état et la position du point par rapport aux autres, le niveau batterie moyen, le nbr de capteurs connecté, total et le ratio.	OK
3	Lancer la simulation puis appliquer l'étape 2 pour chaque état dans lesquels le réseau est passé lors de la simulation		Chaque étape a des informations cohérentes	KO
Rapport de test		Testé par Thibaud BEAUFILS	Le : 27/03/19	
Résultat				
Commentaire			Approbation	
A la fin du cycle 0, le nombre de capteurs comptabilisé est plus faible que le nombre de capteurs réel ayant encore de l'énergie. A la fin des autres cycles seuls les capteurs sans énergie sont comptabilisés. Les capteurs déconnectés sont considérés comme des capteurs actifs sur l'affichage du graphique et sur l'affichage du réseau			Non	

2. Tests unitaires

Tests unitaires	
Couverture des tests	38 / 54 (70.37%)
Complétion des tests	76,3%

Tests unitaires du module FileManager		
Couverture des tests	7 / 13 (53%)	
Complétion des tests	100%	
Tests		
Nom du test	Fonction associé	Exécution
testFMchargerReseauDepuisXML	FMchargerReseauDepuisXML	OK
testFMimporterResultat	FMimporterResultat	OK
testFMchargerEtat	FMchargerEtat	OK
testFMchargerHTMLEtat	FMchargerHTMLEtat	OK
testFMsauvegarderReseauVersXML	FMsauvegarderReseauVersXML	OK
testFMexporterResultat	FMexporterResultat	OK
testFMenregistrerEtat	FMenregistrerEtat	OK
Rapport de test	Testé par Thibaud BEAUFILS	Le : 02/04/19
Commentaire		

Tests unitaires du module FileManager		
Couverture des tests	7 / 8 (87.5%)	
Complétion des tests	100%	
Tests		
Nom du test	Fonction associé	Exécution
testConnexeur	Gconnexeur	OK
testCreerReseau	GcreerReseau	OK
testCreerReseauAvecCapteursEtArcs	GcreerReseauAvecCapteursEtArcs	OK
testGenerationHTML	GgenerationHTML	OK
testGenerationReseau	GgenerationReseau	OK
testGenererPositions	GgenererPositions	OK
testParametrageReseau	GparametrageReseau	OK
Rapport de test	Testé par Thibaud BEAUFILS	Le : 02/04/19
Commentaire		

Tests unitaires du module ReseauControleur		
Couverture des tests	9 / 10 (90%)	
Complétion des tests	0%	
Tests		
Nom du test	Fonction associé	Exécution
testActionSignalFenetrePrincipale	RActionSignalFenetrePrincipale	KO
testActionSignalSimulateur	RActionSignalSimulateur	KO
testActionSignalFenetreCreation	RActionSignalFenetreCreation	KO
testActionSignalGenerateur	RActionSignalGenerateur	KO
testActionSignalGenerateurEmissionResultat	RActionSignalGenerateurEmissionResultat	KO
testMessageInformation	RCmessageInformation	KO
testMessageErreur	RCmessageErreur	KO
testControleParametres	RCcontroleParametres	KO
testActionSignalGenerateurEmissionResultat	RActionSignalGenerateurEmissionResultat	KO
Rapport de test	Testé par Thibaud BEAUFILS	Le : 02/04/19
Commentaire		
Je n'ai pas réussi à lancer les tests car la création d'un objet FenetrePrincipale en dehors de l'environnement classique		

Tests unitaires du module Simulateur		
Couverture des tests	10 / 14 (71%)	
Complétion des tests	100 %	
Tests		
Nom du test	Fonction associé	Exécution
testReseauSansCapteursVides	SreseauSansCapteursVides	OK
testParcourt	Sparcourt	OK
testRouteRecursive	SrouteRecursive	OK
testMigrerPoidsDansArcs	SmigrerPoidsDansArcs	OK
testLancerSimulation	SlancerSimulation	OK
testFinDeVieAtteinte	SfinDeVieAtteinte	OK
testDeterminationRoutage	SdeterminationRoutage	OK
testDeterminationEnsembleDominant	SdeterminationEnsembleDominant	OK
testConfigurationTopologique	SconfigurationTopologique	OK
testActualisationPoids	SactualisationPoids	OK
Rapport de test	Testé par Thibaud BEAUFILS	Le : 02/04/19
Commentaire		

Tests unitaires du module Statistiques		
Couverture des tests	5 / 9 (55.5%)	
Complétion des tests	100 %	
Tests		
Nom du test	Fonction associé	Exécution
testObtenir_niveau_de_batterie_moyen	S_obtenir_niveau_de_batterie_moyen	OK
testGenererTexte	SgenererTexte	OK
testAjouterDonees	SajouterDonnees	OK
testAjouterDonneesBrutes	SajouterDonneesBrutes	OK
testAjouterResultat	SajouterResultat	OK
Rapport de test	Testé par Thibaud BEAUFILS	Le : 02/04/19
Commentaire		

3. Tests de données et d'intégrité

Test de données et d'intégrité : intégrité des données lors de l'importation d'un état				
Objectif		Vérifier que l'importation n'affecte pas les données d'un état		
Eléments à tester		Importation état, fichier XML		
Pré requis				
Scénario				
Id	Démarche	Données	Comportement attendu	OK ?
1	Importer le fichier XML	petit_reseau.xml	Le réseau est importé avec succès et apparaît dans l'application	OK
2	Ouvrir les deux fichiers dans un éditeur de texte et comparer les données	petit_reseau.xml petit_reseau - Copie.xml	Les deux fichiers sont identiques	OK
3	Vérifier la cohérence des données entre celles affichées dans l'application et le fichier importé		L'affichage et le contenu du fichier sont cohérents	OK
Rapport de test		Testé par Thibaud BEAUFILS Le : 28/03/19		
Résultat				
Commentaire			Approbation	
			Validé	

Test de données et d'intégrité : intégrité des données lors de l'importation d'un résultat				
Objectif		Vérifier que l'importation d'un résultat n'affecte pas ses données		
Eléments à tester		Importation résultat, fichier XML		
Pré requis				
Scénario				
Id	Démarche	Données	Comportement attendu	OK ?
1	Importer le dossier de résultats	Dossier resultat_petit_reseau	Le résultat est importé avec succès et apparaît dans l'application	OK
2	Ouvrir les deux dossiers et comparer leurs fichiers : les fichiers statistiques.xml, le premier et le dernier état ainsi qu'un état aléatoire	Dossier resultat_petit_reseau Dossier resultat_petit_reseau - Copie	Les deux dossiers ont des résultats identiques	OK
3	Vérifier la cohérence des données entre celles affichées dans l'application et celles contenues dans le dossier d'origine		L'affichage et le contenu du fichier sont cohérents	OK
Rapport de test		Testé par Thibaud BEAUFILS		Le : 28/06/19
Résultat				
Commentaire			Approbation	
			Validé	

Test de données et d'intégrité : restitution des données lors de l'exportation d'un état				
Objectif		Vérifier l'intégrité des données à l'exportation d'un état		
Eléments à tester		Exportation d'un état. Fichier XML		
Pré requis				
Scénario				
ID	Démarche	Données	Comportement attendu	OK ?
1	Générer un réseau avec les paramètres fournis	- 5 capteurs à déployer - 200 cap batterie - 125 largeur - 20 distance max - 2 distance min	Un réseau est généré et affiché	OK
2	Exporter le réseau au format XML			OK
3	Ouvrir le fichier XML avec un éditeur de texte et vérifier que TOUTES les données concordent en se basant sur les informations fournis dans le manuel d'utilisation		Les métadonnées, les nœuds et les arcs sont cohérents avec le réseau généré	OK
Rapport de test		Testé par Thibaud BEAUFILS	Le : 28/03/19	
Résultat				
Commentaire			Approbation	
			Validé	

Test de données et d'intégrité : restitution des données lors de l'exportation d'un résultat				
Objectif		Vérifier l'intégrité des données à l'exportation d'un résultat		
Eléments à tester		Exportation d'un résultat. Fichiers XML et HTML		
Pré requis				
Scénario				
Id	Démarche	Données	Comportement attendu	OK ?
1	Générer un réseau avec les paramètres fournis	- 5 capteurs à déployer - 200 cap batterie - 125 largeur - 20 distance max - 2 distance min	Un réseau est généré et affiché	OK
2	Lancer la simulation en vérifiant que la case « Voir l'évolution (plus lent) » est cochée		La simulation s'exécute correctement	OK
3	Exporter le résultat de la simulation			OK
4	Ouvrir le fichier statistiques.XML avec un éditeur de texte et vérifier que TOUTES les données concordent en se basant sur les informations fournis dans le manuel d'utilisation		Les données sont cohérentes avec celles affichées	OK
5	Dans un navigateur internet, ouvrir le premier fichier HTML, le dernier et un aléatoire et vérifier la cohérence d'affichage avec l'affichage de l'état correspondant dans l'application		Les deux affichages sont cohérents	OK
6	Ouvrir les fichiers de l'étape 5 en XML et vérifier la cohérence de l'ensemble des fichiers avec l'affichage dans l'application et le réseau généré		Les données sont cohérentes avec celles affichées	OK
Rapport de test		Testé par Thibaud BEAUFILS	Le : 28/03/19	
Résultat				
Commentaire			Approbation	
			Validé	

4. Tests de performances

Test Performances : Simulation avec graphe de 5 capteurs, connexité maximale				
Objectif		Tester le temps d'exécution de la simulation		
Éléments à tester		Le module simulation du programme		
Pré requis		Module génération de capteurs fonctionnel		
Scénario				
Id	Démarche	Données	Comportement attendu	OK ?
1	Importer un réseau	reseau_5_capteurs.xml	Le réseau s'importe avec succès	OK
2	Décocher la case « Voir l'évolution (plus lent) »			OK
3	Lancer la simulation		La simulation dure moins de 5 minutes	OK
Rapport de test		Testé par Thibaud BEAUFILS	Le : 28/03/19	
Résultat				
Commentaire			Approbation	
27s d'exécution			Validé	

Test Performances : Simulation avec graphe de 50 capteurs, connexité maximale				
Objectif		Tester le temps d'exécution de la simulation		
Eléments à tester		Le module simulation du programme		
Pré requis		Module génération de capteurs fonctionnel		
Scénario				
Id	Démarche	Données	Comportement attendu	OK ?
1	Importer un réseau	reseau_50_capteurs.xml	Le réseau s'importe avec succès	OK
2	Décocher la case « Voir l'évolution (plus lent) »			OK
3	Lancer la simulation		La simulation dure moins de 5 minutes	OK
Rapport de test		Testé par Thibaud BEAUFILS	Le : 28/03/19	
Résultat				
Commentaire			Approbation	
52s d'exécution			Validé	

Test Performances : Simulation avec graphe de 150 capteurs, connexité maximale				
Objectif		Tester le temps d'exécution de la simulation		
Eléments à tester		Le module simulation du programme		
Pré requis		Module génération de capteurs fonctionnel		
Scénario				
Id	Démarche	Données	Comportement attendu	OK ?
1	Importer un réseau	reseau_150_capteurs.xml	Le réseau s'importe avec succès	OK
2	Décocher la case « Voir l'évolution (plus lent) »			OK
3	Lancer la simulation		La simulation dure environ 5 minutes	KO 8min50s d'exécution
Rapport de test		Testé par Thibaud BEAUFILS		Le : 28/03/19
Résultat				
Commentaire			Approbation	
			Non	

Test Performances : Simulation avec graphe de 250 capteurs, connexité maximale				
Objectif		Tester le temps d'exécution de la simulation		
Eléments à tester		Le module simulation du programme		
Pré requis		Module génération de capteurs fonctionnel		
Scénario				
Id	Démarche	Données	Comportement attendu	OK ?
1	Importer un réseau	reseau_250_capteurs.xml	Le réseau s'importe avec succès	OK
2	Décocher la case « Voir l'évolution (plus lent) »			OK
3	Lancer la simulation		La simulation dure entre 5 et 10 min	KO 31m35s d'exécution
Rapport de test		Testé par Thibaud BEAUFILS		Le : 28/03/19
Résultat				
Commentaire			Approbation	
			Non	

Test Performances : Génération du graphe le plus grand possible : Affichage				
Objectif		Tester le temps d'exécution de la génération		
Eléments à tester		Le module génération du programme		
Pré requis				
Scénario				
Id	Démarche	Données	Comportement attendu	OK ?
1	Ouvrir la fenêtre de paramétrage de la création de réseau et saisir les paramètres demandés	- 250 capteurs à déployer - 1000 cap batterie - 500 largeur - 83 distance max - 5 distance min		OK
2	Ouvrir un chronomètre			OK
3	Lancer la génération en même temps que le chronomètre. Arrêter ce dernier uniquement		Le temps de création du réseau est de l'ordre de la minute	OK 11s dont environ 6s entre la notification de la création et l'affichage
Rapport de test		Testé par Thibaud BEAUFILS	Le : 28/03/19	
Résultat				
Commentaire			Approbation	
			Validé	

5. Tests de configuration et d'installation

Test de configuration et d'installation : Installation sur un environnement neutre				
Objectif		Tester l'installation sur une machine sans machine virtuelle Python		
Eléments à tester		« Installation.bat » et « Simulateur de Reseaux de Capteurs Dynamique.bat »		
Pré requis		Connexion internet fiable et sans restriction particulière. Environnement sans machine virtuelle Python		
Scénario				
Id	Démarche	Données	Comportement attendu	OK ?
1	Décompresser l'archive	« Simulateur de Reseaux de Capteurs Dynamique.zip »	Dossier contenant : « Installation.bat », « Simulateur de Reseaux de Capteurs Dynamique.bat », Guide d'installation, Guide d'utilisation et dossier « sources »	OK
2	Installer le logiciel en suivant les instruction sur le guide d'installation partie « Installation utilisateur »		L'installation est réussie	OK
3	Lancer « Simulateur de Reseaux de Capteurs Dynamique.bat »		La fenêtre principale de l'application s'ouvre	OK
Rapport de test		Testé par Thibaud BEAUFILS		Le : 28/03/19
Résultat				
Commentaire			Approbation	
			Validé	

Test de configuration et d'installation : Installation sur un environnement avec des configurations Pythons				
Objectif		Tester l'installation sur une machine avec machine virtuelle Python		
Eléments à tester		« Installation.bat » et « Simulateur de Reseaux de Capteurs Dynamique.bat »		
Pré requis		Connexion internet fiable et sans restriction particulière. Environnement avec machine virtuelle Python déjà utilisée		
Scénario				
Id	Démarche	Données	Comportement attendu	OK ?
0	S'assurer que Python est déjà installé en ouvrant un invite de commande et en tapant « python »		Accès au shell python	OK
1	Décompresser l'archive	« Simulateur de Reseaux de Capteurs Dynamique.zip »	Dossier contenant : « Installation.bat », « Simulateur de Reseaux de Capteurs Dynamique.bat », Guide d'installation, Guide d'utilisation et dossier « sources »	OK
2	Lancer « Installation.bat »		L'installation est confirmée avec aucune ligne rouge.	OK
3	Lancer « Simulateur de Reseaux de Capteurs Dynamique.bat »		La fenêtre principale de l'application s'ouvre	OK
Rapport de test		Testé par Thibaud BEAUFILS		Le : 28/03/19
Résultat				
Commentaire			Approbation	
			Validé	



ANNEXE 8 : État de l'art

État de l'art – Thèse de Tifenn Rault

« Energy-efficiency in wireless sensor networks »

- Optimisation de la consommation énergétiques des réseaux sans fils -

La thèse s'oriente autour des réseaux de capteurs sans-fils, l'objectif est d'optimiser l'utilisation de leur énergie afin d'augmenter leur durée de vie. L'application peut se faire à petite (capteurs dans le corps à des fins de contrôle de santé) comme à grande échelle (capteurs dans des champs pour améliorer les récoltes).

L'intérêt de cet état de l'art est la récupération d'informations pouvant être reporté à l'environnement du projet.

Part. I - Généralités sur les réseaux de capteurs sans fil

Dans cette partie, il est exposé la possibilité d'utiliser un puit mobile et un algorithme déterminant si le capteur doit envoyer les données en fonction de la distance avec le puit et de leur place libre en mémoire.

Ensuite une solution de rechargement de batterie à distance, grâce à un chargeur mobile et une technique de transmission d'énergie par plusieurs sauts inter-chargeur, est présentée. Les capteurs prennent alors nouveau rôle d'émetteur/récepteur d'énergie.

Chap. A - Etude des réseaux de capteurs sans fils

Ce chapitre commence par l'identifications des applications existantes, continue sur la proposition d'une nouvelle classification et finit avec l'étude des besoins des applications croisées avec la maximisation de la durée de vie.

La technologie des réseaux sans-fil est prometteuse mais implique un problème majeur : la contrainte énergétique de la batterie par rapport à la consommation. La principale solution est la prise en compte systématique de l'environnement pour des solutions adaptées aux besoins énergétiques qui diffèrent d'une application à l'autre.

Néanmoins on peut répertorier plusieurs catégories de solutions :

- *Duty cycling*, utilisation d'un cycle de sommeil/réveil des capteurs.
- *Data reduction or aggregation*, optimisation de la taille des données à envoyer.
- *Routing protocol*, utilisation de protocoles de routage particuliers.
- *Energy harvesting*, récupération d'énergie par recharge ou récolte.

Pour notre projet, le cas qui nous intéresse d'avantage est celui du protocole de routage.

1. Principe d'un Réseau de Capteur Sans Fil (RCSF)

Un réseau de capteurs sans fil est un réseau de capteurs nœuds formant un réseau connexe avec un puits qui fera le lien avec l'utilisateur, ce que nous explicite la Figure 1. L'avantage d'un tel réseau est notamment l'ajout simplifié d'un nœud. Un des enjeux est l'optimisation à faire sur le matériel en fonction des capteurs nécessaires.

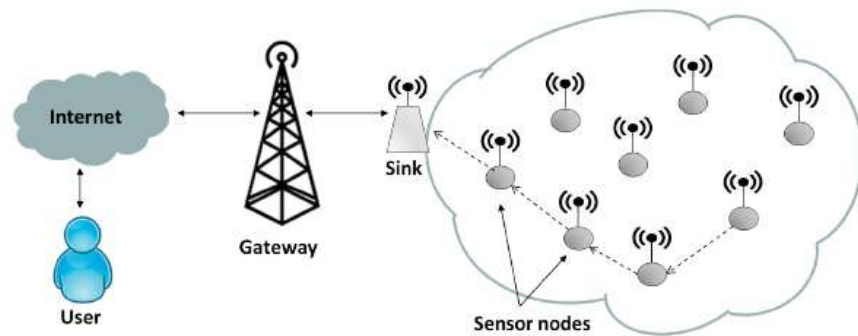


Fig. 1 - Architecture d'un simple RCSF

(Source : *Energy-efficiency in wireless sensor networks*, Tifenne Rault)

La Figure 2 correspond à un tableau de la consommation en énergie des composants récurrents d'un capteur spécifique. Nous remarquons que le matériel de communication sans fil est ce qui consomme le plus, suivi du matériel de calcul puis du principe de récupérations de données.

Module	Mode	Measured current
Radio	Receive	22.8 mA
Radio	Transmit (0 dB)	21.7 mA
Radio	Transmit (-25 dB)	12.1 mA
Radio	Idle (-25 dB)	2.4 mA
Microcontroller	CPU active	2.33 mA
Microcontroller	CPU idle	2.25 mA
Internal flash	Erase	1.35 mA
Internal flash	Write	0.9 - 1.34 mA
Internal flash	Read	0.68 mA
Microcontroller	CPU disable	1.80 μ A

Fig. 2 - Consommation énergétique d'une plateforme sensorielle TelosB

(Source : *Energy-efficiency in wireless sensor networks*, Tifenne Rault)

La durée de vie énergétique d'un tel capteur est de 95h avec la radio constamment allumée et 200h avec la radio allumée 25% du temps.

On peut en conclure que pour augmenter la durée de vie d'un réseau la communication doit être coupée le plus souvent possible.

Il est à noter que la durée de vie d'un RCSF est définie suivant une condition précise de fin de vie variant suivant les besoins (% de capteur en vie, aire de recouvrement minimum, etc.)

2. Étude des application existantes et de leurs besoins

Les domaines d'applications possibles sont nombreux, la Figure 3 détaille les utilisations possibles d'un RCSF dans cinq domaines : la santé, l'environnement et l'agriculture, la sécurité publique et les systèmes militaires, l'industrie et les transports.

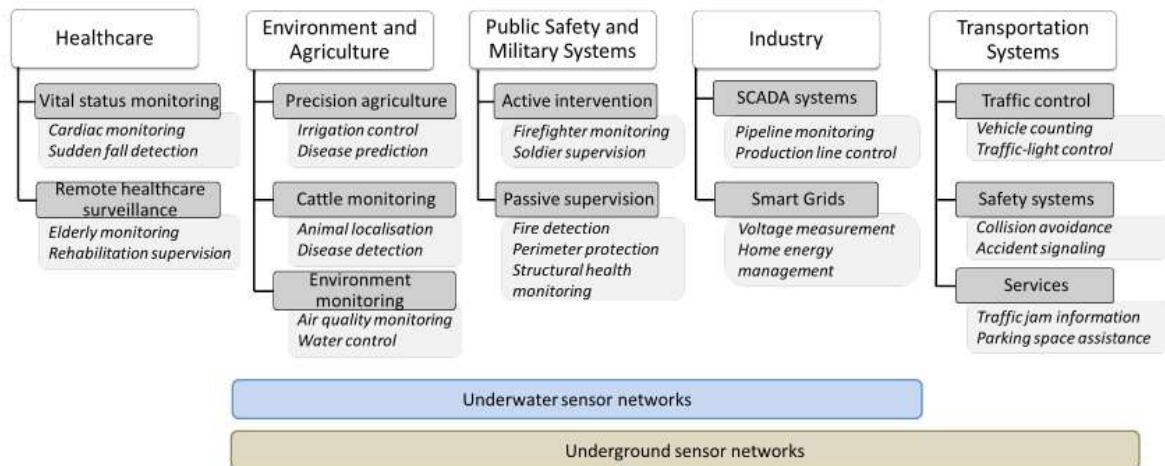


Fig. 3 – Taxonomie des applications d'un RCSF
(Source : *Energy-efficiency in wireless sensor networks*, Tifenne Rault)

La Figure 4 résume les besoins spécifiques par sous-domaines d'application.

		Scalability	Coverage	RT Delay	QoS	Security	Mobility	Robustness
Healthcare	Vital status monitoring	--	--	++	+	++	++	+
	Remote surveillance	--	--	+	+	++	++	-
Agriculture and Environment	Precision agriculture	++	++	--	--	--	--	+
	Cattle monitoring	++	-	-	--	--	+	-
	Environment monitoring	--	--	+	+	++	++	-
Public Safety & Military systems	Active intervention	--	--	++	+	++	++	++
	Passive supervision	--	+	++	+	++	--	-
Transportation systems	Traffic control	--	-	++	++	++	++	-
	Safety system	--	-	++	++	++	+	+
	Services	--	--	-	+	+	+	-
Industry	SCADA systems	--	-	++	+	++	--	++
	Smart grids	+	-	++	+	++	--	++
--								Very low
-								Low
+								High
++								Very high

Fig. 4 – Besoins d'un RCSF suivant ses applications.

Vocabulaire : Scalability = évolutivité, RT Delay = délais de réception-transmission, QoS = QdS (Qualité de services).

(Source : *Energy-efficiency in wireless sensor networks*, Tifenne Rault)

Notre environnement de projet ne prend pas en compte le domaine d'application du réseau.

3. Standards de communication à basse consommation d'énergie

Dans cette partie nous retranscrivons l'ensemble des protocoles évoqués dans la thèse pour la transmission de données sans fil adapté à des réseaux de capteurs.

- **IEEE 802.15.4**, couche physique et couche MAC, implémente un cycle composé en trois parties : 1) les nœuds utilisent le protocole CSMA/CA (protocole de communication à plusieurs accès) 2) certains nœuds prédéfinis bénéficient d'une période pour leurs communications 3) les nœuds dorment sur une troisième période
- **ZigBee**, couche communication par-dessus la couche IEEE 802.15.4., centaines à milliers d'appareils connectés.
- **WirelessHART**, utilise la couche IEEE 802.15.4, utilise TDMA (Time-division multiple access) les nœuds dorment fréquemment. Sécurisé, haute fiabilité, économe en énergie. Utilise un graphe, des recherches ont portées sur un routage des données par qualité de lien entre capteurs, possibilité d'exporter cette recherche sur le niveau de batterie.
- **ISA100.11** utilise aussi la couche IEEE 802.15.4., temps de réveil variable, permet de rendre des nœuds non récepteurs. Utilise un *system manager* qui permet également de choisir en fonction du niveau de batterie des capteurs.
- **Bluetooth Low Energy (BLE)**. Pas cher, faible consommateur d'énergie, faible portée. Conçu pour qu'un appareil dure un an
- **IEEE 802.15.6** utilisé pour des réseaux sur et dans le corps humain composé d'un puit et jusqu'à 64 nœuds. Utilise la couche MAC et physique. 3 niveaux de sécurité (max : authentification et encryptage)
- **6LoWPAN** s'ajoute à la couche IEEE 802.15.4, permet aux nœuds d'utiliser l'IPv6 afin de se connecter aux réseaux et appareils externes.
- **RPL** compatible IPv6. Permet de définir un routage suivant de nombreux critères
- **MQTT**, envisagé comme le futur de la communication des objets connectés à internet. Compatible avec le protocole TCP/IP. MQTT-S permet d'avoir plusieurs passerelles avec les réseaux externes, supporte les nœuds endormis. Compatible avec ZigBee

BLE et les protocoles basés sur IEEE 802.15.4 sont conçus pour les appareils fonctionnant sur batterie.

4. Mécanismes de sauvegarde d'énergie

Evoqués en introduction du chapitre, nous retrouvons, Figure 5 les axes d'optimisation exploitables pour l'augmentation de la durée de vie du réseau de capteur.

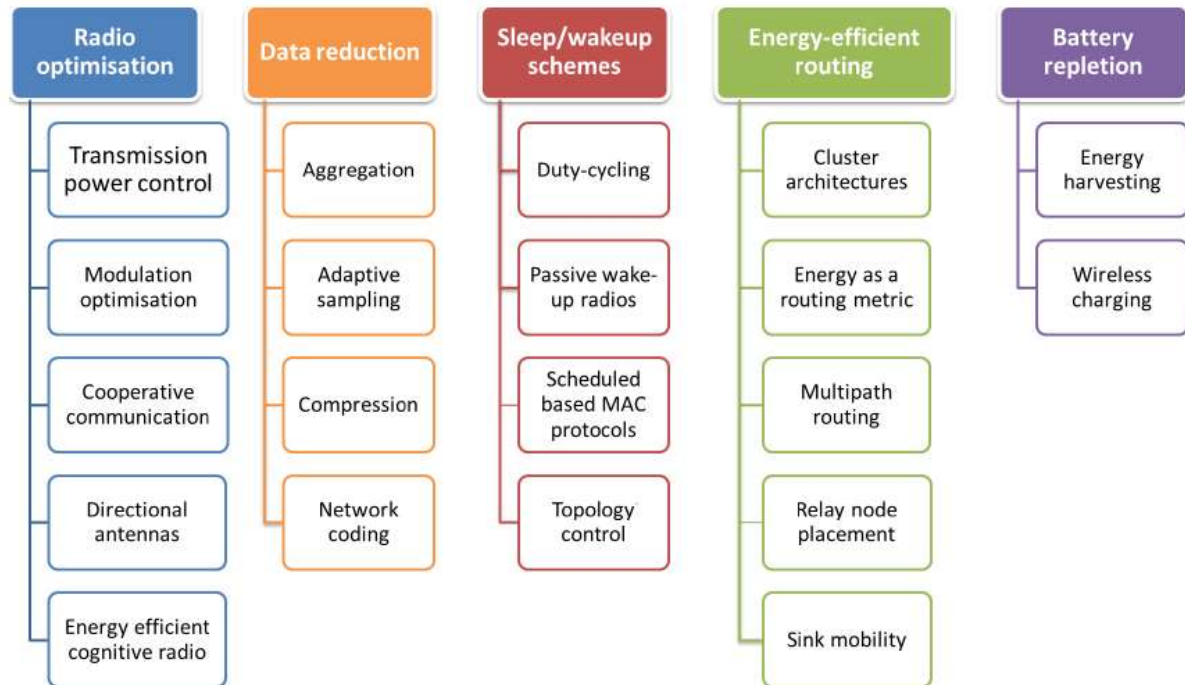


Fig. 5 – Classification des mécanismes d'économie d'énergie

(Source : *Energy-efficiency in wireless sensor networks*, Tifenne Rault)

Dans cette partie nous nous intéresserons uniquement à trois parties : la réduction des données, le schéma de cycles coucher/réveil et le routage des données.

Réduction des données

Deux axes sont envisageables pour cette solution : la limitation des extraits inutiles et celle du nombre de récupérations de données. Pour cela il existe différentes techniques :

- Agrégation : la fusion de données afin de réduire le total de données envoyées. L'avantage est la diminution des latences dues au trafic, l'inconvénient est la possible perte de données et d'informations.
- Ajustement de l'extraction : rendre intelligent l'activation des capteurs par des combinaisons de conditions. Par exemple, une caméra qui s'allume uniquement si le détecteur sonore relève quelque chose.
- Codage du réseau : combiner linéairement les paquets envoyés en broadcast.
- Compression des données. La plupart des algorithmes de compression ne sont pas compatibles avec le peu de ressources embarqués sur les capteurs. Certaines techniques ont cependant été développées.

Schéma coucher/réveil

Trois axes sont envisageables pour cette solution : l'utilisation de cycle de réveil, l'application d'un mode passif sur les radios des capteurs et une gestion par région du réseau.

- Cycle : cette méthode permet une grande économie d'énergie, cependant elle peut augmenter considérablement le temps de latence car un capteur a beaucoup de chance d'attendre qu'un autre se réveille. La Fig. 6 résume les avantages et inconvénients des trois schémas possibles : réveille sur demande, réveils asynchrones, ou par rendez-vous programmé.

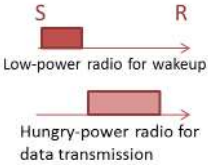
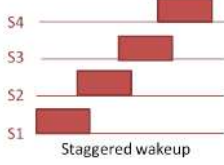
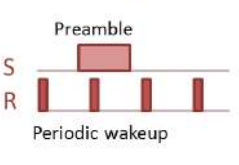
Type	On-demand	Schedule rendez-vous	Asynchronous
Principe	Wake up a node only when another wants to communicate with it.	Nodes wakeup at the same time as its neighbours according to a wakeup schedule. Then they go to sleep until their next rendez-vous.	Each node wakes up independently but its active period must overlap with its neighbours.
Broadcast	No	Yes	No
Synchronisation	No	Yes	No
Energy-efficiency	Nodes remain active only for the minimum time required.	More collisions because nodes wakeup at the same time after an inactive period.	Nodes need to wake up more frequently. Either the sender sends long preamble or the receiver remains awake longer.
Examples of applications	Even-driven application with low duty cycle.	Data-gathering application with possibility of aggregation.	Mobile applications when the neighbourhood is unpredictable.
Illustration	 <p>Low-power radio for wakeup</p> <p>Hungry-power radio for data transmission</p>	 <p>Staggered wakeup</p>	 <p>Preamble</p> <p>Periodic wakeup</p>

Fig. 6 – Les avantages et inconvénients de trois méthodes de cycle coucher/réveil

(Source : *Energy-efficiency in wireless sensor networks*, Tiffene Rault)

- Radio passives : dans cette situation les radios de transmission ont deux modes : actives, où des données sont transmises (coûteux en énergie), et passives, où aucune donnée n'est transmise, mais où le capteur attend d'être appelé à une communication, par un second nœud. Ce schéma nécessite néanmoins un certain temps aux données pour attendre le puit.
- Contrôle topologique : cette méthode se situe dans le cas où les capteurs couvrent une zone géographique. Cette zone est divisée en plusieurs plus petites dans lesquelles seul un capteur reste éveillé, à portée des autres capteurs éveillés. C'est ce capteur qui reçoit et transmet les données des capteurs de sa région.

La méthode de cycle associée à celle de contrôle topologique peuvent être celles qui nous intéresseraient le plus, transposé à notre projet.

Cinq axes sont envisageables pour cette méthode : la division du réseau en Clusters, la prise en compte de l'énergie comme critère de routage, l'utilisation de chemins multiples, le placement de nœuds relais et l'utilisation d'un puit mobile.

- **Clusters** : le réseau est divisé en groupes possédant un « nœud principal » qui coordonne les autres nœuds du groupe et communique avec les autres clusters. Cette technique a l'avantage de :
 - Réduire la portée des communications des nœuds non principaux
 - Limiter le nombre de communication
 - Permettre à deux nœuds d'être éteints
 - Distribuer la consommation énergétique grâce au roulement des nœuds principaux.
- **L'énergie comme critère de routage**. En plus du plus court chemin, l'algorithme prend en compte l'énergie local du réseau, il y a deux méthodes possibles :
 - Exponential and Sine Cost Function Based Route (ESCFR), qui prend en compte l'énergie restante du nœud.
 - Double Cost Function based Route (DCFR), qui prend en compte l'énergie restante ainsi que l'énergie nécessaire pour la transmission.
- **Chemins multiples** : alternent les chemins utilisés pour envoyer les données. L'avantage du multi-chemin est qu'une route peut être reconstruite facilement en cas de dysfonctionnement d'un nœud. Ici aussi il y a deux méthodes possibles :
 - Energy-Efficient Multipath Routing Protocol (EEMRP), qui prend en compte plusieurs chemins possibles avec le nombre de sauts et la quantité restante d'énergie des nœuds parcourus.
 - Energy-Efficient and Collision Aware protocol (EECA), qui construit deux routes disjointes pour aller d'un nœud à un puit afin de prévenir les collisions.
- **Placement de nœuds relais** : une optimisation possible est le placement de nœuds avec d'avantages d'énergie sur des points qui consomment davantage d'énergie. Cette solution est également liée au placement adapté du puit par rapport à l'ensemble des nœuds.
- **Mobilité du puit** : cette solution est la plus adaptée pour palier à la consommation plus rapide d'énergie des nœuds situés à proximité du puit.

Dans notre cas, la méthode des Clusters, le critère énergétique et la multiplicité des chemins sont les plus susceptibles de nous intéresser.

5. Économie d'énergie

Dans cette partie nous résumons l'approfondissement qui a été fait sur divers approches d'économie énergétique. Y est abordé en premiers les protocoles de routage à plusieurs critères, l'extension de la notion de sauvegarde d'énergie au niveau protocolaire réseau et l'approche de l'optimisation multi-objective.

- **Protocoles de routage à plusieurs critères** : un certain nombre de protocoles émergeant prennent en compte plusieurs critères comme la sécurité, la qualité des liens entre nœuds, ou la consommation énergétique. L'inconvénient de ces protocoles est qu'ils impliquent davantage d'échanges de messages de contrôle.
 - **ATSR (Ambient Trust Sensor Routing)** : Décision de routage faite localement à partir d'un poids calculé en fonction de l'énergie restante du voisinage, de la localisation et d'un indicateur de confiance calculé à partir de 7 critères de sécurité (réputation, authentification, intégrité des messages, etc.).
 - **ERTLD (Enhanced Real-Time routing protocol with Load Distribution)** : Utilisé pour les nœuds mobiles, prend en compte l'énergie restante, les délais de réception sur plus d'un saut et le RSSI (Received Signal Strength Indication). Évite le problème de trous de routage, possède un haut ratio de livraison et consomme peu d'énergie.
 - **PEMuR (Power Efficient Multimedia Routing)** : Sélectionne le chemin vers le puits sur lequel il restera le plus d'énergie après l'envoi des données. S'adapte également à la bande passante. Idéal pour transmettre des vidéos. Cependant chaque nœud envoie des informations sur son énergie restante et sa localisation.
 - **InROut** : Réservé principalement à l'industrie. Haut fiabilité, prend en considération le ratio d'erreur de paquets et l'énergie.
- **Approche de couches croisées**. C'est le principe de mise en place d'une prise en charge de la notion de sauvegarde d'énergie aux couches Physiques et MAC et non plus seulement dans le microprocesseur. Cette technique permet la réduction de latence et augmente l'économie d'énergie.
- **Optimisation multi-objective**. Obtient, pour un routage, plusieurs solutions optimisées à partir d'un algorithme évolutif ou d'un « Jeu théorique ».
 - Un algorithme évolutif est comparable à une évolution biologique : plusieurs solutions sont générées à partir de plusieurs critères, celle qui correspond le mieux est gardée, les autres éliminées. A la prochaine utilisation la survivante est modifiée pour créer une nouvelle génération qui est elle aussi évaluée, etc. Deux algorithmes ont été sélectionnés pour cette méthode : l'algorithme MODE prend en compte la consommation énergétique et la latence et l'algorithme DPAP optimise la consommation énergétique et la couverture géographique.
 - L'approche par « Jeu » est plus adaptée au cas où deux autorités se partagent un réseau car la relation entre nœud est basée sur le partage. En effet le routage prend en considération la réputation des nœuds.

Chap. B - Collection des données avec un puit mobile

Ce chapitre propose un nouveau schéma de collection de donnée en exposant son modèle de programmation puis en évaluant ses performances à travers des simulations.

Un des principaux problèmes d'un réseau de capteurs émetteur/récepteurs avec batterie est la durée de vie énergétique plus courte des capteurs proches des puits. En effet, ils se situent à des points de convergence des données et sont donc d'avantage sollicités. Une des solutions est la mobilité des puits.

Deux types de mobilités sont possibles : une mobilité contrôlée ou non contrôlée. La première implique un challenge d'optimisation de trajet pour le puit et un problème de routage vers le puit pour les nœuds.

La proposition est de placer une mémoire tampon sur les nœuds qui, seulement une fois pleine, se transmet vers le puit en multi-saut, et ce, si le puit n'est pas encore passé près du nœud.

1. Travaux relatifs

Il existe déjà plusieurs possibilités concernant cette solution :

- Par transfert multi-sauts, l'envoi des données est constant vers le puit ce qui implique de faibles latences mais une grande consommation d'énergie. Plusieurs approches sont possibles :
 - Détermination du temps de trajet et du taux de transfert entre les nœuds
 - Ajout de contraintes comme le temps maximal entre deux mouvements consécutifs, le temps minimum passé sur une localisation et le coût énergétique pour la construction d'un nouveau routage.
 - Approche évolutive où le déplacement du puit est revu à chaque tour en fonction de l'énergie des nœuds.
 - L'utilisation de plusieurs puits variablement mobiles.
- Par simple saut, les nœuds attendent d'être à portée des puits pour envoyer les données. Un grand gain de durée de vie est possible, au détriment de la latence et d'une forte capacité requise en mémoire tampon. Ici aussi l'approche avec un ou plusieurs puits est possible.
- Par routage hybride (multi et simple sauts), ici deux possibilités :
 - Envoi des données en multi-saut vers un nœud « tête » constant qui enverra les données vers le puit en saut simple. Ce nœud a donc plus de mémoire que les autres.
 - Envoi des données en multi-saut vers les nœuds qui seront sur la trajectoire prévue du puit et pourront lui envoyer les données en un saut.

2. Programmation linéaire

Cette partie explique le modèle linéaire présenté dans le chapitre.

Le modèle linéaire exposé détermine pour une topologie donnée : le temps que passe le puit à différente localisation, le flux de donnée dans le voisinage des nœuds et la mise en mémoire tampon des paquets.

$$\max \sum_{l_k \in K} t_{l_k} \quad (1)$$

$$\sum_{l_k \in K} \sum_{i: j \in S_i^{l_k}} e_{ij}^T q_{ij}^{l_k} t_{l_k} + \sum_{l_k \in K} \sum_{j: i \in S_j^{l_k}} e_{ji}^R q_{ji}^{l_k} t_{l_k} \leq E_i, i \in N \quad (2)$$

$$w_i^{l_k} = \sum_{j: i \in S_j^{l_k}} t_{l_k} q_{ji}^{l_k} + t_{l_k} Q_i - \sum_{j \in S_i^{l_k}} t_{l_k} q_{ij}^{l_k} + w_i^{l_{k-1}}, i \in N, k \in \{0, 1, \dots, |L|\} \quad (3)$$

$$\sum_{i \in N} Q_i t_{l_k} + \sum_{i \in N} w_i^{l_{k-1}} - \sum_{i \in N} w_i^{l_k} = \sum_{j: s \in S_j^{l_k}} t_{l_k} q_{js}^{l_k}, l_k \in K \quad (4)$$

$$w_i^{l_0} = 0, i \in N \quad (5)$$

$$t_{l_k} \geq 0, l_k \in K \quad (6)$$

$$q_{ij}^{l_k} \geq 0, i \in N, j \in S_i^{l_k}, l_k \in K \quad (7)$$

$$w_i^{l_k} \geq 0, i \in N, l_k \in K \quad (8)$$

$$q_{ij}^{l_k} \leq R_{ij}, i \in N, j \in S_i^{l_k}, l_k \in K \quad (9)$$

$$w_i^{l_k} \leq w_i, i \in N, l_k \in K \quad (10)$$

N , ensemble de nœuds et s , puit

$Q_i > 0, i \in N$, taux de génération de données du nœud i

L , ensemble des positions possibles de s

$k = (l_1, l_2, \dots, l_{|L|-1}, l_{|L|})$, liste ordonnée des visites de s

$t_{l_k} \geq 0$, temps que s passe à une position l_k

$S_i^{l_k} \subseteq N \cup \{s\}$, ensemble des nœuds qui sont dans la zone de transmission du nœud i et une position donnée l_k de s

$q_{ij}^{l_k} \geq 0$, taux de transmission du nœud i vers j quand s , est à une position l_k

$w_i \geq 0$, capacité de stockage du nœud i

$w_i^{l_k} \geq 0$, quantité de données stockées dans le nœud i quand s quitte la position l_k

$R_{ij} (i \in N, j \in S_i^{l_k})$, capacité du lien (i, j)

E_i , énergie initiale du nœud i

e_{ij}^T , énergie nécessaire à i pour envoyer à j

e_{ji}^R , énergie nécessaire à j pour recevoir de i

Détail des équations

Equation (1) : Maximisation du temps de vie de réseau, c'est-à-dire la somme du temps passé par le puit à tous les endroits.

Equation (2) : L'énergie totale dépensée par nœud ne doit pas dépasser leur énergie initiale. C'est-à-dire que la somme pour chaque lien et chaque nœud de l'énergie utilisé pour transmettre et recevoir ne doit pas dépasser l'énergie totale ; énergie calculée en fonction de l'énergie théorique nécessaire, le taux de transmission et le temps passé par le puit dans la zone.

Equation (3) : A un instant donné, la quantité de données stockées dans le nœud est égale à la somme :

- De la quantité de données reçue des autres nœuds
- De la quantité de données générées
- De la quantité de données déjà présentes

En lui enlevant la quantité de données envoyées vers les autres nœuds.

Equation (4) : Quand le puit quitte la position donnée, la somme des transmissions des nœuds vers le puit est égale à la somme de la quantité de données générée pendant ce temps avec la quantité de données précédemment stockée soustrait à la quantité restante (c'est à dire qui n'a pas le temps d'être transmise).

3. Résultats numériques

Pour vérifier l'efficacité de la modélisation, les résultats sont comparés avec ceux d'une solution qui fournit les routages et réveils programmés optimaux sans l'utilisation de mémoire tampon.

La modélisation est réalisée sur une aire de 350m² avec un réseau variant de 300 à 700 nœuds avec, pour chaque taille de réseau, l'utilisation d'une topologie différente. La distance transmission est de 22 à 35m. Sur la figure les carrés rouges représentent des points possibles de situation du puit. Tous les nœuds ont la même quantité d'énergie initiale et les mêmes limites de transmission.

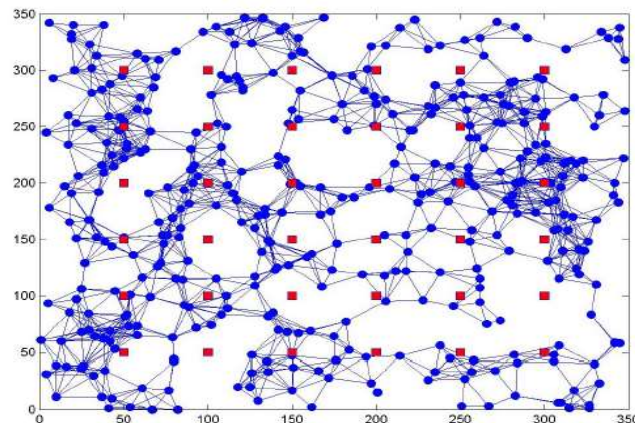


Fig. 7 – Topologie aléatoire d'un réseau de 400 nœuds sur une grille de 6x6 positions possibles du puit (Source : *Energy-efficiency in wireless sensor networks*, Tifenne Rault)

$ N $	REF [151]	OPT	Improvement
300	91 553	114 177	24.71%
400	71 335	88 294	23.70%
500	43 093	62 204	44.34%
700	28 080	41 035	46.13%

Fig. 8 – Durée de vie moyenne des deux modèles. De gauche à droite : le nombre de nœuds, la durée de vie du modèle de référence, la durée de vie du modèle proposé et la différence de performance. (Source : *Energy-efficiency in wireless sensor networks*, Tifenne Rault)

Le tableau Fig. 8 montre le résultat obtenu pour les deux méthodes. On peut également préciser que la nouvelle méthode consomme plus équitablement la totalité de l'énergie distribuée parmi les capteurs, avec une diminution de reste énergétique après la fin de vie du réseau, d'entre environ 5 et 7%.

Cette méthode est prouvée efficace, cependant elle a le défaut de nécessiter la programmation individuelle de chaque nœud, ce qui est compliqué à mettre en place et rigide une fois déployé.

4. Algorithme distribué

Cet algorithme permet de déterminer à quel moment le capteur doit envoyer ses données. Il fonctionne sur le principe du plus court chemin en mettant à jour la position du puit.

Afin de réduire le temps de latence et les possibilités de pertes de données, les nœuds ne traitent que celles qu'il produit. Les autres données sont directement redirigées.

L'algorithme proposé est plutôt basique :

Le nœud envoie un quart de ses données (pour ne pas congestionner le réseau) si le puit est proche à un certain nombre de sauts (limite identique pour tous les nœuds) ou que la quantité de données stockées dépasse un seuil. Ce seuil, différent pour chaque nœud, est une fonction linéaire de la distance entre le nœud et le puit.

5. Résultats de la simulation

Le simulateur utilisé se nomme « OM-Net++ simulator ». IEE 802.15.4 est le protocole utilisé, en plus d'un protocole de routage par plus court chemin. Chaque fois que le puit change de position il envoie un message en broadcast pour notifier les nœuds de sa nouvelle position.

Les résultats obtenus sont comparées avec un cas d'envoi en multi-saut continue vers le puit (sans utilisation de mémoire tampon) et un cas d'un envoi à simple saut (c'est-à-dire en attendant que le puit passe à côté). Le tableau Fig. 9 résume les caractéristiques des nœuds.

Parameter	Value
data generation rate	1 data every 30s
data/route/MAC-ack packet size	128/4/4 Byte
buffer capacity	16 KB
link capacity	250 Kbps
t_{min}	10-100 s
DIS K-Hops (a, b)	(3,40)

Fig. 9 – Paramètres de simulation

(Source : *Energy-efficiency in wireless sensor networks*, Tifenne Rault)

Résultats

Le premier graphique représente la consommation d'énergie en fonction du nombre de sauts de distance avec le puit avant transmission des données. La tendance de celui-ci, en opposition avec le second graphique représentant la latence, est d'augmenter en même temps que le nombre de sauts autorisés. Le dernier ensuite, montre le pourcentage de données arrivant à destination.

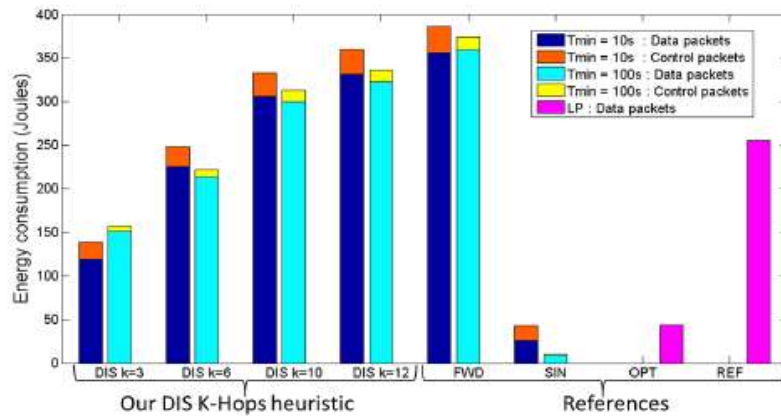


Fig. 10 – Consommation énergétique d'un réseau de 300 nœuds pour différentes solutions
(Source : *Energy-efficiency in wireless sensor networks*, Tifenne Rault)

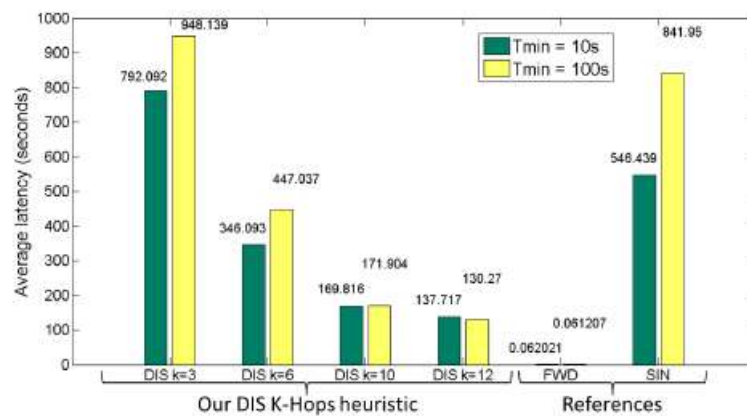


Fig. 11 – Latence moyenne d'un réseau de 300 nœuds pour différentes solutions
(Source : *Energy-efficiency in wireless sensor networks*, Tifenne Rault)

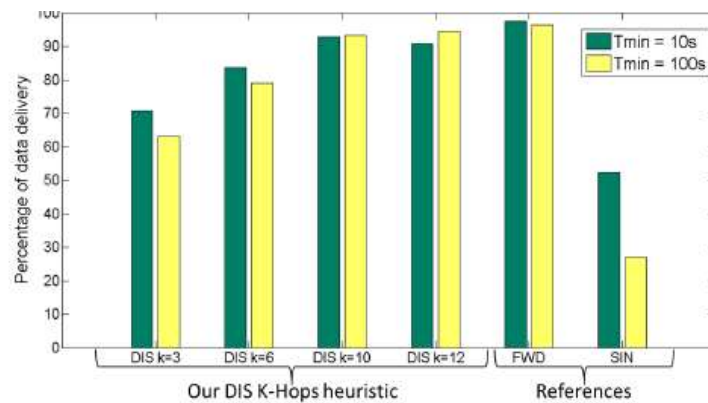


Fig. 12 – Ratio de données délivrées sur un réseau de 300 nœuds pour différentes solutions. (Source : *Energy-efficiency in wireless sensor networks*, Tifenne Rault)

Plus le nombre de sauts autorisé est élevé plus la consommation énergétique est forte. Cependant on observe une tendance inverse pour la latence. La variation du ratio de transmission/réception des données ne varie pas beaucoup, néanmoins une perte supérieure à 20% n'est pas négligeable. Si nous estimons que pour dix sauts et plus, la consommation énergétique est trop grande mais que pour trois sauts l'attente est trop grande, nous pouvons en conclure qu'il est préférable de choisir un nombre de sauts limité aux alentours de six.

Chap. C - Rechargement sans fil des capteurs par sauts inter-capteurs

De récentes avancées technologies dans le rechargement sans-fil rendent très intéressantes son utilisation sur des systèmes à faible consommation d'énergie comme les RCSF. La problématique est de savoir où déployer ces chargeurs sur le réseau avec une approche multi-saut.

Il faut savoir que précédemment cette technique permettait seulement d'allonger la durée de vie du réseau sans pouvoir le recharger complètement. Il en est de même pour la récolte d'énergie au niveau du capteur (trop dépendant de l'environnement). Cependant, avec les récentes avancées il est désormais possible de rendre un RCSF opérationnel pour toujours, d'un point de vue énergétique, en plus, cette technologie est utilisable en multi-saut.

En termes d'exemple, la technique Witricity permet de fournir une lampe de 60W à 2m de distance et un ratio de 40%. La plupart des utilisations actuelles utilisent un chargeur mobile.

Le modèle qui sera exposé prend en compte la demande énergétique des nœuds, la capacité des chargeurs et la perte énergétique des transferts.

1. Travaux relatifs

En ce qui concerne le rechargement par saut unique, il existe de nombreux schémas :

- Les chargeurs mobiles qui font des rondes par région.
- Les chargeurs mobiles qui vont charger les nœuds qui en ont fait la demande.
- Les nœuds qui envoient leurs niveaux d'énergie à une station qui calcule le trajet que doit effectuer le chargeur mobile.
- Des tours de chargeur mobile (Qi-Ferry) qui sont effectués pour un capital énergétique. Afin d'optimiser le trajet, le chargeur réduit son nombre de visite par tour si toute son énergie n'a pas été consommée.
- Des chargeurs mobiles qui suivent les routages utilisés pour transmettre l'information. Le chargeur a ensuite trois possibilités : charger les nœuds avec le minimum d'énergie, les nœuds qui ont la durée de vie plus courte (prenant en compte la consommation) ou nœuds où le trafic est plus intense.
- Le calcul du chemin le plus rapide en incluant des capteurs qui sont les plus chargés, afin d'éviter les latences induites par le chargement (car la même fréquence que celle pour envoyer les données est utilisée)
- L'optimisation, pour le chargeur mobile, du ratio entre le temps passé à charger les nœuds et le temps passé à la station. Ce schéma suppose que le chargeur a assez d'énergie pour charger tous les nœuds le même tour.
- Utilisation de chargeur pouvant effectuer plusieurs rechargements simultanément. Les nœuds nécessitant d'être chargés sont regroupés par cluster ensuite visités par le chargeur mobile.

Pour le rechargement en plusieurs sauts, il se fait avec la possibilité de charger sur plusieurs sauts avec un ratio de 20% sur 8 sauts. Deux scénarios sont possibles :

- Le nœud qui n'a plus d'énergie envoie une demande aux autres pour qu'ils lui envoient une partie de leur énergie.
- Le graphe est séparé en groupe possédant un chef. Si un nœud nécessite de l'énergie il en fait la demande au chef de groupe. Le chef distribue son énergie ou bien renvoie un message groupé. Si tout le groupe n'a pas assez d'énergie pour répondre à la requête, le chef se fait recharger par un chargeur mobile puis transmet de l'énergie au premier nœud qui a demandé.

Le problème de charger les nœuds par multi-saut en minimisant le nombre de chargeur statique n'a pas encore été abordé dans la littérature scientifique.

2. Présentation de la solution proposée

Le réseau doit être couvert par un ensemble de batteries à capacités fixes. L'objectif est de déterminer le nombre minimum de chargeurs nécessaires et leur position en prenant en compte une schéma multi-saut.

En première étape un arbre de plus court chemin est construit pour chaque possible position des chargeurs. En second lieu une équation linéaire est utilisée pour déterminer le nombre minimum requis pour recharger tous les nœuds.

Les nœuds peuvent transmettre de l'énergie à plusieurs nœuds différents, mais un seul à la fois, et en recevoir d'un seul nœud, en correspondance avec un arbre.

Etape 1 : Pour chaque (i, j) , calcul de π_{ij} , p_{ij}^* et $l(p_{ij}^*)$ nombre de nœuds entre i et j avec le chemin p

Etape 2 : Application de l'équation linéaire avec préalablement le calcul :

$$B_{ij}(i \in N, j \in Z_i^h) = \begin{cases} 1 & \text{si } i \in T_j \\ 0 & \text{sinon} \end{cases}$$

$$B_{jj} = \begin{cases} 1 & \text{si l'arbre avec la racine } j \text{ existe} \\ 0 & \text{sinon} \end{cases}$$

$$\min \sum_{j \in N} B_{jj} \quad (1)$$

$$B_{ij} \leq B_{jj}, i \in N, j \in Z_i^h \quad (2)$$

$$\sum_{j \in Z_i^h} B_{ij} = 1, i \in N \quad (3)$$

$$\sum_{i \in Z_j^h} E_i \pi_{ij} B_{ij} \leq C, j \in N \quad (4)$$

$$\sum_{x \in P_{ij}} B_{xj} \geq l(p_{ij}^*) B_{ij}, i \in N, j \in Z_i^h, i \neq j \quad (5)$$

N , ensemble de nœuds

S , chargeur à capacité C

$E_i > 0$, demande énergétique du nœud i

$k_{ij}(= k_{ji}) \geq 1$, coefficient de perte énergétique par simple saut

P_{ij} , ensemble des chemins de i vers j

$K_{ij}^p = \prod_{(x,y) \in p} k_{xy}, p \in P_{ij}$, coefficient de perte énergétique par multi-saut

$\pi_{ij} = \min_{p \in P_{ij}} K_{ij}^p = K_{ij}^{p_{ij}^*}$, coefficient minimum de perte énergétique par multi-saut

p_{ij}^* , chemin qui minimise π_{ij}

T_j , arbre de racine j

h , taille maximale de l'arbre

Z_i^h , ensemble des nœuds les plus loin de h -sauts de i

Détail des équations

Equation (1) : Minimise le nombre de nœuds chargeurs, c'est-à-dire la somme des nœuds racines d'arbre.

Equation (2) : Permet de s'assurer qu'un nœud chargeur est racine de l'arbre contenant un nœud donné.

Equation (3) : Un nœud ne peut appartenir qu'à un arbre, donc être relié à un chargeur

Equation (4) : L'ensemble de l'énergie demandée par les nœuds de l'arbre de la batterie doit être inférieur à sa capacité

Equation (5) : Le nombre de nœuds du plus court chemin pour aller d'un nœud au chargeur doit être inférieur ou égal au nombre de points contenus dans l'arbre d'origine le chargeur. S'assure que tous les nœuds du chemin sont contenus dans l'arbre.

3. Évaluation des performances

Le réseau évalué est composé de 100 nœuds sur un carré de 25m². La demande énergétique de chaque nœud est définie à 1KJ et la capacité de la batterie entre 20KJ et 2000KJ. Le nombre de saut a également été considéré comme une variable. La perte de transmission a été fixée à 20%. Enfin, la modélisation a été résolue avec CPLEX.

Le graphique Fig. 13 expose le nombre optimal du nombre de chargeur en fonction du nombre de sauts. Etant donné que l'amélioration n'est pas très marquante entre un chargeur de 500 et 2000KJ, la solution optimale retenu est de 8 chargeurs de 500KJ.

Le graphique Fig. 14 montre le nombre optimal de chargeur en fonction de la capacité du chargeur et pour plusieurs nombres de sauts.

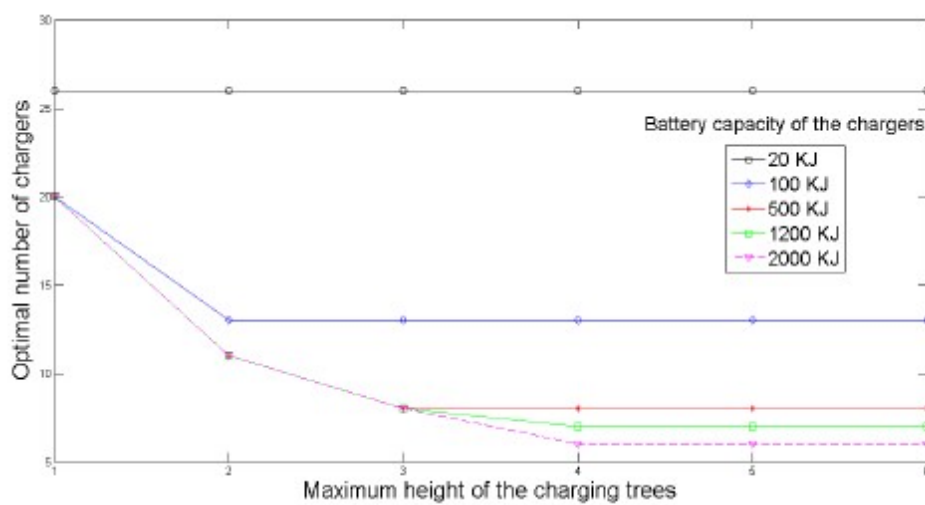


Fig. 13 – Nombre minimum de chargeurs suivant le nombre maximal de saut pour une transmission d'énergie et la capacité de la batterie

(Source : *Energy-efficiency in wireless sensor networks*, Tifenne Rault)

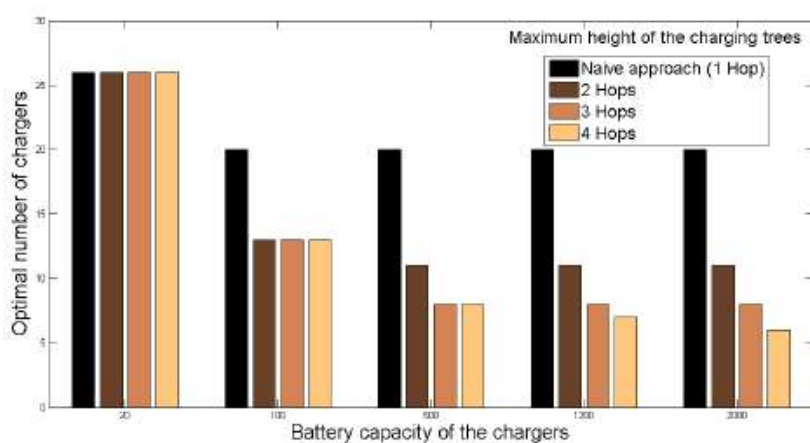


Fig. 4.3 The minimum number of chargers for our multihop energy transfer scheme compared to a naive single hop energy transfer approach.

Fig. 14 – Nombre minimum de chargeurs pour une approche multi-saut en comparaison avec un transfert en saut unique et suivant la capacité de la batterie.

(Source : *Energy-efficiency in wireless sensor networks*, Tifenne Rault)

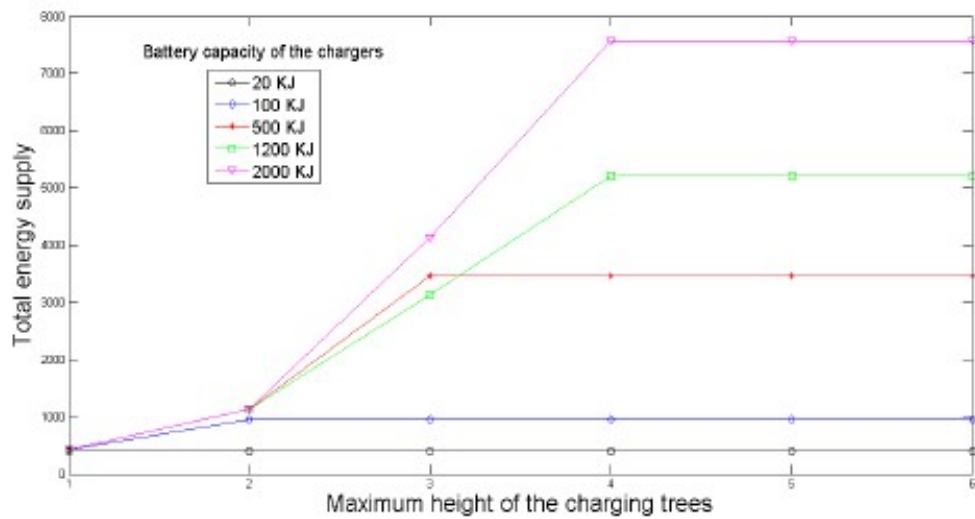


Fig. 15 – L'énergie totale distribuée, en fonction du nombre maximal de sauts et de la capacité de la batterie (Source : *Energy-efficiency in wireless sensor networks*, Tifenne Rault)

Finalement ce troisième graphique montre l'énergie maximale fournie par un chargeur pour un arbre suivant le nombre de saut. On remarque que ce total est plafonné à 4 sauts de la même façon qu'il y a un plafonnement figure 4.2 du nombre de chargeur optimal pour 4 sauts.

En prenant en compte le plafonnement du premier et troisième graphique nous pouvons conclure qu'il est plus intéressant de prendre un maximum de 3 sauts et une capacité de 500KJ. Au-delà le gain ne serait pas assez intéressant par rapport au coût de la batterie et au coût en énergie

Part II - Application sur un RCSF, domaine de la santé

Cette partie développe le cas d'utilisation de leur solution de RCSF appliqué au contrôle de l'état de santé du corps humain. Elle se décompose en trois chapitres :

- **Chapitre 5** - Etude des solutions actuelles des systèmes de suivi médicaux humains (état de l'art des solutions de capteurs intégrés à l'individu, classification des méthodes de sauvegarde d'énergie, comparaison qualitative des solutions existantes, identification des possibilités d'orientation de recherche)
- **Chapitre 6** - Schéma de sélection de passerelles pour économiser de l'énergie (proposition d'une nouvelle architecture économe en énergie pour un réseau sur le corps humain, analyse numérique, proposition d'un algorithme de distribution, évaluation des performances à travers des simulations)
- **Chapitre 7** - Prototype de réseau de capteurs sur individus (création d'un prototype de capteurs extra-corporels, implémentation du schéma de sélection de passerelles, développement d'une application de contrôle)

Cette partie traitant d'un cas particulier éloigné de l'objectif du projet, nous ne la développerons ici.



Bibliographie

- [1] Rault, Bouabdallah, Challal. (18 Juin 2014). *Energy Efficiency in Wireless Sensor Networks : atop-down survey*. [Lien](#)
- [2] Hagberg, Schult, Swart. (19 Septembre 2018). *NetworkX Reference*. [Lien](#)
- [3] *Plotly Reference*. [Lien](#)
- [4] *PyQt5 Reference Guide*. [Lien](#)
- [5] *Matplotlib User's Guide*. [Lien](#)

Développement Algorithmique et Conception d'un Simulateur pour un Réseau de Capteurs dynamiques :

Thibaud Beaufls

Encadrement : Ameer Soukhal

Contexte

Les capteurs d'un réseau consomment de l'énergie en récoltant, transmettant ou relayant l'information. Les enjeux :

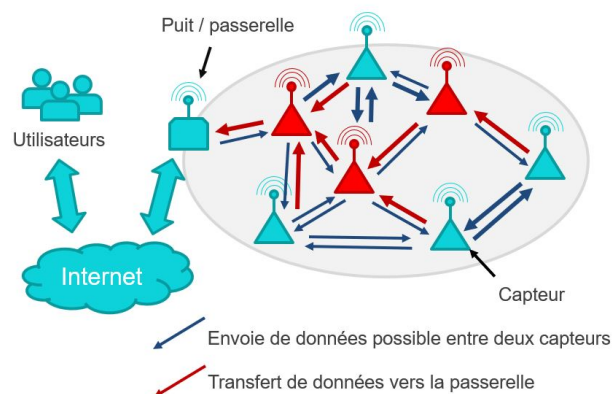
- déterminer le routage optimal ;
- déterminer les capteurs relais ;
- déterminer l'intervalle de temps entre les changements de rôle des capteurs.

Solution

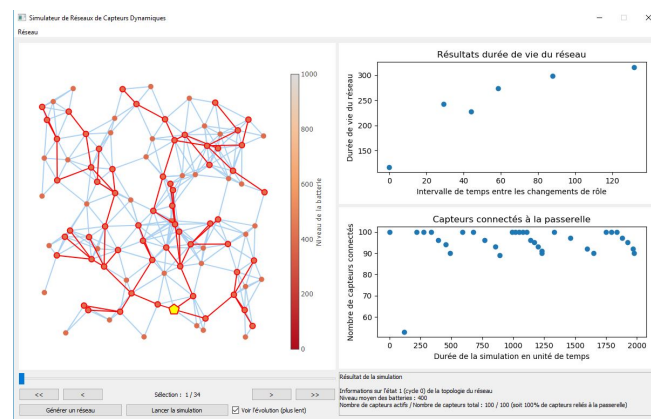
La conception d'un simulateur à partir d'une recherche algorithmique préalable a été retenue. Le simulateur explore différentes valeurs possibles de l'intervalle de temps entre les changements de rôle des capteurs, pour déterminer la durée de vie maximale.

Résultats

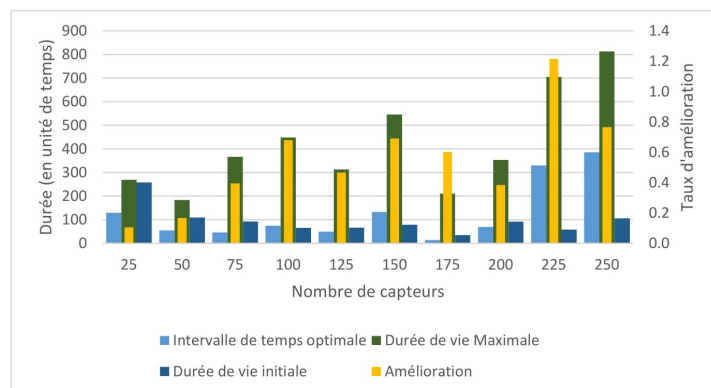
Le simulateur développé permet d'obtenir des résultats intéressants avec une moyenne d'amélioration de 50%. Néanmoins le temps d'exécution s'envole aux alentours de 200 capteurs avec un temps d'exécution pouvant dépasser la demi-heure.



Réseau de capteurs. Les capteurs rouge ont le rôle de relayer l'information, les bleus le rôle de l'émettre.



Simulateur développé. Affichage du réseau et des graphiques affichant les résultats exploitables.



Résultats obtenus en fonction du nombre de capteurs dans le réseau.

Développement Algorithmique et Conception d'un Simulateur pour un Réseau de Capteurs dynamiques :

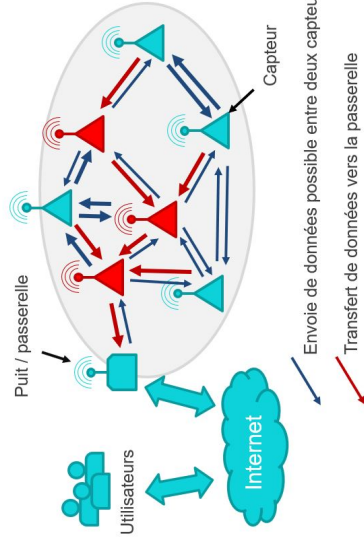
Thibaud Beaufils

Encadrement : Ameer Soukhal

Contexte

Les capteurs d'un réseau consomment de l'énergie en récoltant, transmettant ou relayant l'information. Les enjeux :

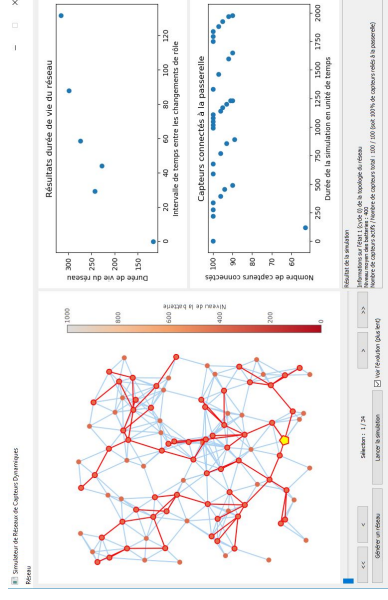
- déterminer le routage optimal ;
- déterminer les capteurs relais ;
- déterminer l'intervalle de temps entre les changements de rôle des capteurs, pour déterminer la durée de vie maximale.



Réseau de capteurs. Les capteurs rouge ont le rôle de relayer l'information, les bleus le rôle de l'émettre.

Solution

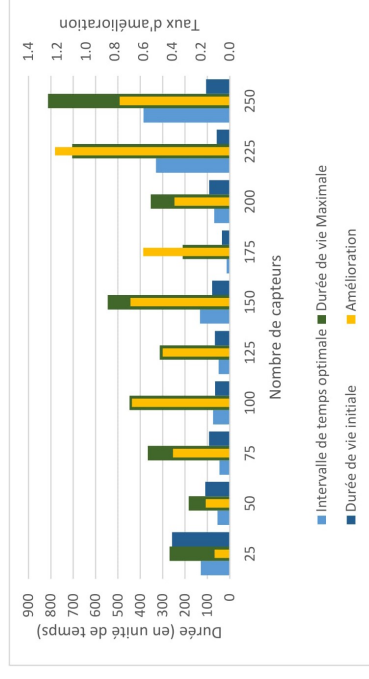
La conception d'un simulateur à partir d'une recherche algorithmique préalable a été retenue. Le simulateur explore différentes valeurs possibles de l'intervalle de temps entre les changements de rôle des capteurs, pour déterminer la durée de vie maximale.



Simulateur développé. Affichage du réseau et des graphiques affichant les résultats exploitables.

Résultats

Le simulateur développé permet d'obtenir des résultats intéressant avec une moyenne d'amélioration de 50%. Néanmoins le temps d'exécution s'envole aux alentours de 200 capteurs avec un temps d'exécution pouvant dépasser la demi-heure.



Résultats obtenus en fonction du nombre de capteurs dans le réseau.

Développement Algorithmique et Conception d'un Simulateur pour un Réseau de Capteurs dynamiques

Résumé

Le projet concerne les réseaux de capteurs, l'objectif est de maximiser leurs durée de vie en minimisant leur consommation énergétique. Les capteurs d'un réseau consomment de l'énergie en récoltant, transmettant ou relayant l'information. Les enjeux qui en découlent et sur lesquels nous nous focaliserons sont :

- la détermination du routage optimal ;
- le choix du rôle des capteurs entre émetteur de l'information ou relayeur ;
- la détermination de l'intervalle de temps entre les changements de rôle des capteurs pour mieux répartir la consommation énergétique.

La solution du problème est exprimée sous la forme d'un simulateur permettant d'explorer différentes valeurs possibles de l'intervalle de temps afin de déterminer la durée de vie maximale.

Mots-clés

Réseau, Simulateur, Capteurs, graphes, Python, NetworkX, PyQt

Abstract

The project is about sensors networks. The aims is to get the best life time with the less energie consupction. Sensors consumes energie by harvesting, sending or receiveing data. We will focus on :

- the dertermination of the best route ;
- the choice of wich sensors would be sender or relay runner ;
- the calculation of the delay between the reconsideration of the sensors role to split energie consupction.

The solution take the form of a simulator that calculate several delay and keep the one giving the best living time.

Keywords

Network, Simulator, Sensors, Graphs, Python, NetworkX, PyQt