Mise en place d'un service de géolocalisation au sein d'une plateforme d'exploitation d'un réseau de capteurs sans fil.

Anthony Deroche

Rapport de stage

Mise en place d'un service de géolocalisation au sein d'une plateforme d'exploitation d'un réseau de capteurs sans fil

Anthony Deroche

Année 2013–2014

Stage de 2e année réalisé dans l'entreprise INRIA

Maître de stage : Thibault CHOLEZ

Encadrant universitaire : Bertrand PETAT

Déclaration sur l'honneur de non-plagiat

Je soussigné(e),

Nom, prénom : Deroche, Anthony

Élève-ingénieur(e) régulièrement inscrit(e) en 2e année à TELECOM Nancy

Numéro de carte de l'étudiant(e) : 31010052

Année universitaire : 2013–2014

Auteur du document, mémoire, rapport ou code informatique intitulé :

Mise en place d'un service de géolocalisation au sein d'une plateforme d'exploitation d'un réseau de capteurs sans fil

Par la présente, je déclare m'être informé(e) sur les différentes formes de plagiat existantes et sur les techniques et normes de citation et référence.

Je déclare en outre que le travail rendu est un travail original, issu de ma réflexion personnelle, et qu'il a été rédigé entièrement par mes soins. J'affirme n'avoir ni contrefait, ni falsifié, ni copié tout ou partie de l'œuvre d'autrui, en particulier texte ou code informatique, dans le but de me l'accaparer.

Je certifie donc que toutes formulations, idées, recherches, raisonnements, analyses, programmes, schémas ou autre créations, figurant dans le document et empruntés à un tiers, sont clairement signalés comme tels, selon les usages en vigueur.

Je suis conscient(e) que le fait de ne pas citer une source ou de ne pas la citer clairement et complètement est constitutif de plagiat, que le plagiat est considéré comme une faute grave au sein de l'Université, et qu'en cas de manquement aux règles en la matière, j'encourrais des poursuites non seulement devant la commission de discipline de l'établissement mais également devant les tribunaux de la République Française.

Fait à Villers-lès-Nancy, le 24 août 2014

Signature :

Rapport de stage

Mise en place d'un service de géolocalisation au sein d'une plateforme d'exploitation d'un réseau de capteurs sans fil

Anthony Deroche

Année 2013–2014

Stage de fin de deuxième année dans l'entreprise INRIA

Anthony Deroche
10, Avenue Jean Jaurès
54500, Vandoeuvre-lès-Nancy
+33 (0)6 87 58 04 75
anthony.deroche@telecomnancy.eu

TELECOM Nancy
193 avenue Paul Muller,
CS 90172, VILLES-LES-NANCY
+33 (0)3 83 68 26 00
contact@telecomnancy.eu

INRIA
615, rue du Jardin Botanique
54600, VILLERS-LES-NANCY CEDEX
+33 (0)3 83 59 30 00

Maître de stage : Thibault CHOLEZ

Encadrant universitaire : Bertrand PETAT

Remerciements

Je remercie dans un premier temps l'équipe MADYNES de l'institut de recherche INRIA pour m'avoir accueilli chaleureusement durant ces 10 semaines de stage.

Je tiens à remercier particulièrement Thibault Cholez et Emmanuel Nataf pour leurs précieux conseils, ainsi que pour la confiance et le temps qu'ils m'ont accordés. Je remercie également Thierry Duhal avec qui j'ai collaboré sur la réalisation du projet qui m'a été confié.

Pour m'avoir permis de participer au Best Student Recognition Event 2014 à Prague, organisé par IBM du 30 juin au 2 juillet, j'aimerais également remercier Rémi Badonnel.

Enfin, je tiens aussi à remercier l'ensemble du personnel du LORIA qui a contribué au bon déroulement de ce stage, notamment l'équipe s'occupant de la restauration pour leur sympathie.

Résumé

Dans les réseaux de capteur sans-fil, les nœuds sondent l'environnement et perçoivent des grandeurs physiques telles que la température ou l'humidité. A ce titre, la position de ceux-ci est une information critique souvent utilisée à des fins de corrélation. A travers l'étude de trois algorithmes (moyenne pondérée, trilatération, multilatération), nous avons mis en place un module de géolocalisation basé sur l'indication de force du signal (RSSI). Cette fonctionnalité est intégrée dans une plateforme web JEE et le système embarqué est Contiki. Dans un premier temps, résultant d'une communication pair à pair, les RSSI sont collectés depuis le réseau, et routés jusqu'au sink. Par la suite, une passerelle transfère ces données au serveur qui se charge de leur traitement et enregistrement. Les résultats obtenus à partir des algorithmes de géolocalisation sont envoyés à l'interface utilisateur qui peut ainsi consulter les résultats et voir la topologie du réseau sur une carte. La comparaison des algorithmes nous amène, dans le meilleur des cas et pour différentes topologies, à une précision autour de 3m. Dans un but de réduction de la consommation d'énergie, nous avons mis en place un module de commandes. Celles-ci sont envoyées depuis l'interface web et nous permettent de désactiver certains modules tels que la géolocalisation. S'inscrivant dans la tendance des Open Data, les données collectées sont mises à disposition via une API. Enfin, une interface d'administration permet la gestion de la plateforme web et la supervision du réseau de capteurs par l'intermédiaire d'envoi de commandes au sink relayées dans le reste du réseau.

Mots-clés : réseau de capteurs sans fil, géolocalisation, supervision

Abstract

In wireless sensor networks, nodes perceive the local environnement by sensing physical data such as temperature, humidity. As such, localization becomes a critical information and is frequently needed for correlation purposes. Through the study of three algorithms (weighted average, trilateration, multilateration), we set up a localization system based on the received strength signal indication (RSSI). This functionality is integrated into a JEE web service and we use the embedded OS Contiki. First of all, as a result of a peer to peer communication, relevant data such as RSSI are collected from the network and routed to the sink. Then a gateway forwards the data to the server which is in charge of processing and persistence. The results of localization algorithms are sent to the web UI and the user is able to check out results and view the network topology on a map. It turns out that the comparison of algorithms brings us to an accuracy of about 3m. Furthermore, to reduce energy consumption, we add the support of commands in the network which are actually sent from the web service and allow us to disable a module such as geolocation. In the trend of Open Data, we provide all the data collected from the network through an API. Finally, an administration interface allows administrators to manage the web platform, and to monitor the network by sending commands to the sink which relays them to the entire network.

Keywords : Wireless Sensor Network, geolocation, monitoring

Table des matières

Introduction

De nos jours, la plupart des objets qui nous entourent ont la vocation d'être interconnectés. Étant équipés d'une puce ou d'un capteur, ils se mettent ainsi à produire des données. Dans ce contexte de "l'Internet des objets", l'innovation est importante et c'est ainsi que de nombreuses applications voient le jour. On peut par exemple citer les domaines d'applications tels que la planification urbaine (environnement urbain durable, éclairage public), la gestion des déchets, la domotique (gestion du chauffage et de l'éclairage, alarmes de sécurité). Ces nouvelles applications font souvent appel à des réseaux de capteurs sans-fil, ces derniers faisant l'objet d'un travail de recherche important, notamment en termes de sécurité, d'optimisation et d'économie d'énergie.

Par ailleurs, la supervision, le contrôle de ces réseaux, et la consommation énergétique sont des enjeux importants. Dans ce contexte, les données issues des différents capteurs présents sur les nœuds peuvent être corrélés avec la position de ceux-ci. Cette information critique est utile pour beaucoup d'applications, et peut être également utilisée dans un but de tracer les objets connectés. C'est pourquoi ce sujet a suscité un intérêt important en termes de travaux de recherche.

Ce stage effectué chez INRIA comporte plusieurs objectifs. Tout d'abord, il convient de faire un état de l'art de la recherche dans ce domaine et par la suite une étude de différents algorithmes de localisation des capteurs. Le principal objectif poursuivi est de permettre la comparaison des performances de différents algorithmes de localisation dans un environnement réaliste de bureaux, à travers la collecte des données utiles au sein du système d'information et l'implantation des algorithmes considérés au sein d'une plateforme web expérimentale existante.

1. Présentation de l'institut de recherche

1.1 Présentation globale de Inria

Institut de recherche français public, Inria concentre sa recherche dans les domaines de l'informatique et des mathématiques appliquées. Il a été initialement créé le 3 janvier 1967 dans le cadre du Plan Calcul afin d'assurer à la France son indépendance en matière de super-ordinateurs. Inria est rattaché au ministère de l'enseignement supérieur et de la recherche et au ministère en charge de l'industrie. Son actuel président directeur général est Michel Cosnard. L'institut est composé de huit centres répartis sur le territoire français et son siège social est localisé à Rocquencourt dans le département des Yvelines. L'organigramme d'Inria est disponible en annexe.

Inria est composé de 168 équipes-projets parmi lesquelles sont rassemblés 1677 chercheurs et 1772 universitaires ou chercheurs d'autres organismes, avec un budget primitif de 235 millions d'euros. En partenariat avec les acteurs de la recherche publique et privée en France et à l'étranger, ces équipes mettent en œuvre leurs compétences et concentrent leurs savoirs afin d'inventer les technologies du numérique de demain. En plus d'être à l'origine de la création de plus de 110 start-up, le travail de ces équipes se traduit par la publication de près de 4500 articles en 2013.

Plus précisément, les activités de recherche d'Inria sont regroupées en cinq domaines de recherche :

— Santé, biologie et planète numériques
— Mathématiques appliquées, calcul et simulation
— Perception, cognition, interaction
— Réseaux, Systèmes et services, calcul distribué
— Algorithmique, programmation, logiciels et architectures

Pour ma part, j'ai effectué mon stage dans le centre de Nancy - Grand-Est au sein du LORIA, dans l'équipe MADYNES présentée par la suite.

1.2 Le laboratoire d'accueil : LORIA

Le laboratoire dans lequel j'ai effectué mon stage est le laboratoire lorrain de recherche en informatique et ses applications (LORIA) situé à Villers-lès-Nancy. Formellement, le LORIA est une unité mixte de recherche créée en 1997, commune à plusieurs établissements que sont le CNRS, l'Université de Lorraine et Inria. Le LORIA regroupe 28 équipes au sein de 5 départements, dont 15 communes avec Inria, pour un total de plus de 500 personnes. Un organigramme du laboratoire est disponible en annexe B.

A travers ces équipes, le LORIA dispose de compétences variées dans les sciences et technologies de l'information et de la communication. Les équipes sont divisées en 5 départements :

— Algorithmique, calcul, image et géométrie
— Méthodes formelles
— Réseaux, systèmes et services
— Traitement automatique des langues et des connaissances
— Systèmes complexes et intelligence artificielle

Outre les bonnes conditions de travail, la vie au sein du laboratoire est très active et cela se manifeste par des présentations régulières d'intervenants des équipes ou extérieurs. Les sujets abordés sont très variés et j'ai pu assisté à une quinzaine d'entre elles, dont une soutenance de thèse. Ces présentations permettent aux équipes de suivre et voir le travail effectué par leurs collègues.

1.3 L'équipe MADYNES

L'équipe MADYNES [1] appartient au département Réseaux, systèmes et services, et se concentre sur l'étude expérimentale et appliquée des nouvelles méthodes de gestion de l'Internet de futur. Cela regroupe la configuration, la supervision, et la sécurité des services et protocoles de communication. La recherche de l'équipe se focalise sur l'étude de systèmes dynamiques tels que :

— Les réseaux de capteurs
— Les réseaux sans-fil auto organisés
— Les réseaux IPv6
— Les réseaux centrés sur le contenu
— Les réseaux pair-à-pair à grande échelle

L'équipe est composée de maîtres de conférence, de professeurs, d'ingénieurs, d'un chargé de recherche, et de doctorants. Le chef d'équipe est actuellement Isabelle Chrisment qui a récemment succédé à Olivier Festor.

1. Managing dynamic networks and services

2. Contexte matériel et logiciel

S'intégrant dans la continuité du travail de recherche effectué par l'équipe MADYNES sur les réseaux de capteurs, ce stage s'appuie sur du matériel mis à disposition ainsi qu'une plateforme web existante.

2.1 Le réseau de capteurs

2.1.1 Description d'un noeud du réseau

Le réseau de capteurs est composé de nœuds, aussi appelés "motes" dont voici une photo ci-dessous.

FIGURE 2.1 – Noeud du réseau

Conçus par l'université de Californie à Berkeley, ce mote est de type TelosB et se nomme MTM-CM5000-MSP. Il est équipé d'un micro-contrôleur Texas Instruments MSP430 16 bits avec un convertisseur numérique analogique 12 bits, d'une puce radio Texas Instruments CC2420, de différents capteurs (température, humidité, luminosité, infrarouge), d'un port USB mâle, de LEDs... Certains d'entre eux possèdent également un connecteur permettant l'ajout d'une antenne externe. La puce radio émet sur la bande de fréquence 2.4Ghz - 2.485Ghz, et dispose d'une puissance d'émission de 0 à -25 dBm configurable de manière logicielle permettant une portée de 20 à 30 mètres dans un environnement intérieur. Au niveau mémoire, un noeud possède 10Ko de mémoire vive et seulement 48Ko de mémoire flash, ce qui est relativement peu. De plus, le micro-contrôleur ne dispose pas d'une unité de calcul en virgule flottante.

Le port USB permet de connecter le nœud, lui donnant ainsi la possibilité d'envoyer des données sur l'interface série d'une machine. Ceci est d'autant plus utile lorsque le nœud fait office de puit (ou "sink"), car cela permet la collecte des données provenant de tout le réseau.

2.1.2 Le système d'exploitation embarqué : ContikiOS

Chaque nœud embarque le système d'exploitation libre ContikiOS. Ce système a été créé par Adam Dunkels en 2002 au sein d'une équipe de recherche suédoise. Étant conçu spécialement pour ce type de petits appareils en réseau, ContikiOS a l'avantage d'être léger et de favoriser une consommation énergétique minimale. De plus, il supporte les protocoles IPv6 et 6LoWPAN (IPv6 Low power Wireless Personal Area Network). Cela s'avère particulièrement utile dans la mesure où les nœuds communiquent en IPv6 et utilisent le standard 802.15.4 défini par l'IEEE (Institute of Electrical and Electronics Engineers).

Pour établir la topologie du réseau, les nœuds utilisent le protocole de routage RPL (Routing Protocol for Low power and Lossy Networks). Chaque paquet IPv6 du réseau doit passer par un chemin pour arriver jusqu'au sink. L'algorithme RPL construit un graphe orienté acyclique des nœuds et permet ainsi la découverte d'une route entre un nœud émetteur et le sink. RPL assure ainsi une topologie dynamique du réseau, dans le cas où un nœud capteur est ajouté, déplacé, retiré, ou défaillant. Chaque nœud doit donc être en mesure de relayer les paquets de ses voisins. Il doit alors pouvoir jouer à la fois le rôle d'émetteur et celui de récepteur.

Nous avons choisi le protocole de transport UDP (User Datagram Protocol) pour les communications entre les nœuds plutôt que TCP (Transport Control Protocol). En effet, bien que des pertes soient possibles avec UDP, les mécanismes assurant la fiabilité de TCP sont trop consommateurs d'énergie. Par ailleurs, puisque la principale consommation d'énergie du nœud provient de l'utilisation du module radio, le système d'exploitation provoque l'endormissement et le réveil régulier de celui-ci. La force du signal reçu par la puce radio (RSSI : Received Strength Signal Indication) peut être directement lu via l'interface de la couche réseau micro IP. La programmation d'applications dans ce système d'exploitation se fait dans le langage C et pour permettre la concurrence, Contiki utilise des protothreads qui sont en fait des threads légers sans pile spécialement conçus pour les environnements avec peu de mémoire comme les réseaux de capteurs.

2.2 Plateforme et logiciels existants

2.2.1 Un simulateur réseau pour Contiki : Cooja

Pour développer les programmes au sein de Contiki, le système met à disposition un simulateur réseau appelé Cooja. Le logiciel permet d'émuler des noeuds et de charger un programme compilé. Ceci est particulièrement utile pour tester les programmes avant de les mettre dans la mémoire flash des nœuds réels, puisque le logiciel simule les conditions d'exécution et de mémoire de la plateforme TI MSP430. Les données collectées provenant du sink via sa sortie standard peuvent être enregistrées dans des fichiers ou lus par des logiciels qui peuvent par la suite traiter et présenter les données à l'utilisateur. On peut par exemple citer le logiciel collect-view intégré dans Contiki qui permet de visualiser les

valeurs des capteurs et des données de supervision du réseau.

En revanche, Cooja a un intérêt limité si l'on veut tester des algorithmes de géolocalisation basés sur le RSSI. En effet, le logiciel utilise un modèle linéaire pour simuler ces valeurs en fonction de la distance, alors qu'en réalité, le modèle est logarithmique. De plus, l'environnement simulé ne reflète pas la réalité en raison de la non prise en compte des perturbations engendrées par les murs, sols...

Un aperçu du logiciel est disponible en annexe C.1.

2.2.2 Le système d'information initial : du réseau de capteurs à l'application serveur

Lors du projet de découverte de la recherche effectué dans le cadre de ma deuxième année à TELECOM Nancy en collaboration avec M. Thierry Duhal, nous avons développé une plateforme web avec la technologie JEE (Java Platform, Enterprise Edition) ainsi qu'une passerelle en Java.

Au niveau des nœuds du réseau, ils embarquaient une application de collecte présente dans Contiki et le flux de données résultant comportait les données des capteurs (température, humidité, luminosité).

Lisant le port USB où est connecté le sink, la passerelle permettait initialement le transfert des données provenant de celui-ci vers l'application serveur via des requêtes HTTP.

Concernant l'application serveur, elle centralisait les données dans un modèle relationnel et présentait graphiquement les valeurs des différents capteurs sur une interface web. L'accès aux ressources fournies par l'application serveur se fait via une API. Nous avions également une carte présentant la topologie du réseau établi par l'algorithme de routage. Une carte peut être visualisée en annexe D.1 et illustre un déploiement réalisé à TELECOM Nancy au mois de mai 2014.

Durant ce stage, je suis parti de ce système d'information en vue d'améliorer les composantes et de les rendre plus génériques afin de permettre l'intégration de modules tels que la géolocalisation ou encore un module de gestion du réseau.

3. Présentation des algorithmes de géolocalisation

Il existe plusieurs techniques de géolocalisation pour les réseaux de capteur sans fil. Bien que la localisation par GPS soit populaire, son utilisation reste cependant très limitée dans les environnements en intérieur. C'est pourquoi j'ai focalisé mon étude sur des techniques exploitables à l'intérieur des bâtiments. Dans ce contexte, comme décrit dans [1], il existe des algorithmes qui estiment la distance ou angles entre les nœuds : TOA (time of arrival), TDOA (time difference of arrival), AOA (angle of arrival), RSSI (received strength signal indication). Étant donné l'équipement des nœuds présentés précédemment, la seule technique que j'ai retenue est celle basée sur le RSSI. Les autres nécessitent des modules spécifiques absents sur nos nœuds tels que de la synchronisation précise.

3.1 Prise en compte de l'environnement

Pour cette étude, il est nécessaire d'utiliser un modèle mathématique afin d'estimer les distances en fonction des RSSI. Néanmoins, il n'est pas souhaitable de dépendre d'une intervention humaine longue et laborieuse de quadrillage de la zone afin de relever des mesures de référence, dans le but de déterminer les paramètres du modèle. En effet, une telle procédure de calibration rend en pratique impossible le déploiement d'un réseau à grande échelle. C'est pourquoi, j'ai décidé de mettre en place une calibration automatique du modèle, avec l'appui des documents [2] et [3]. Nous considérons quelques nœuds du réseau dont la position est connue (nœuds ancrés). Par ailleurs, nous ne prenons pas en compte les atténuations liées à la présence de murs et de sols entre les ancres dans le modèle car cela suppose une saisie manuelle de ces paramètres.

Par conséquent, en considérant L l'atténuation du signal en dB, l_0 l'atténuation à 1 mètre, et d la distance, on obtient le modèle suivant :

$$L(d) = l_0 + 10\alpha log_{10}(d)$$

En pratique, l_0 est une constante obtenue expérimentalement. En considérant des nœuds ancrés avec antenne, on a $l_0 = 20dB$. Il reste à déterminer α, qui correspond en fait à l'exposant de l'atténuation du signal. La méthode consiste à inférer le paramètre α à partir des couples $< d, RSSI >$ collectés à partir d'échanges de paquets entre les nœuds ancrés. En considérant deux ancres i et j, on a donc $RSSI_{(i,j)} = l_0 - 10\alpha log_{10}(d_{(i,j)})$ Ainsi, avec n couples de valeurs, on se ramène à un problème de moindres carrés et on cherche α tel que l'erreur soit minimisée :

$$\min_{n \in N} \sum_{i=0}^{n} e_i \text{ avec } e_i = |RSSI_i - (l_0 - 10\alpha log_{10}(d_i))|$$

Une fois ce paramètre déterminé, il est possible d'estimer la distance entre un nœud ancré et un nœud mobile à partir du $RSSI \in [-90, 0]$ du paquet échangé de cette manière :

$$d(RSSI) = 10^{(\frac{RSSI - l_0}{-10\alpha})}$$

En supposant que les nœuds s'échangent des paquets régulièrement, l'exposant d'atténuation α du modèle est ainsi déterminé de manière automatique. Sa valeur est dynamique et dépend de l'environnement du réseau de capteurs. Ce paramètre est important puisque les algorithmes de localisation présentés par la suite sont basés sur cette distance estimée.

Expérimentalement, j'ai constaté que $\alpha \approx 1.4$ avec des nœuds en ligne de vue équipés d'antennes et $\alpha \approx 2$ avec des nœuds dans un environnement de bureaux équipés d'antennes. J'ai vérifié la cohérence du modèle en prenant des mesures manuelles à travers une expérience menée sur un parking extérieur. On constate bien sur la figure 3.1 que le RSSI évolue de manière logarithmique en fonction de la distance. La courbe rouge représente les mesures réelles alors que les deux autres sont logarithmiques et bornent la courbe réelle.

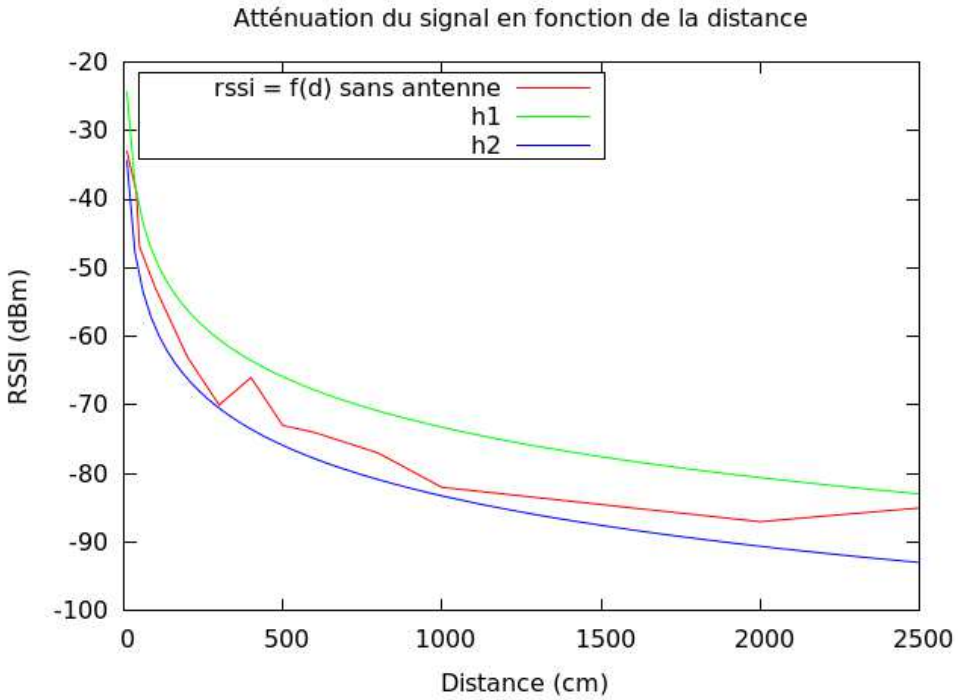


FIGURE 3.1 – Evolution du RSSI en fonction de la distance

3.2 Méthode de la moyenne pondérée (ou barycentre)

On considère n nœuds ancrés et un nœud mobile. On pondère les coordonnées des ancres avec l'inverse de la distance estimée entre le nœud mobile et l'ancre considérée. Cela

permet de donner plus d'importante aux faibles distances. En utilisant cette méthode, on obtient les coordonnées suivantes :

$$x_g = \frac{\sum_{i=0}^{n}(w_i * x_i)}{\sum_{i=0}^{n} w_i} \; ; \; y_g = \frac{\sum_{i=0}^{n}(w_i * y_i)}{\sum_{i=0}^{n} w_i} \text{ avec } w_i = \frac{1}{d_i}$$

Néanmoins, étant donné que les poids w_i sont tous positifs, le point résultant de cette méthode se trouvera toujours à l'intérieur du contour externe formé par les ancres. Autrement dit, si on considère 3 ancres, le résultat se trouve forcément dans le triangle formé par celles-ci.

3.3 Méthode de la trilatération

3.3.1 Principe général

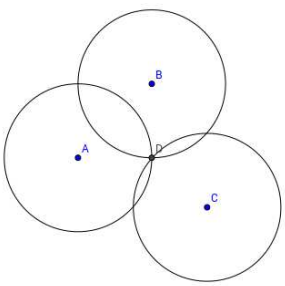
Contrairement à la triangulation qui repose sur les angles, la trilatération est un algorithme qui permet de déterminer une position à partir de distances en exploitant la géométrie des cercles, des sphères ou des triangles. Dans le cas présent, nous considérons 3 ancres et un nœud à localiser. Nous connaissons la position des ancres et une distance correspondant à un cercle, nous pouvons ainsi tracer 3 cercles centrés sur les ancres. Le rayon de ces cercles correspond à la distance entre le nœud mobile et l'ancre en question.

Dans le cas parfait, l'intersection des 3 cercles est un point et il suffit de résoudre le système suivant :

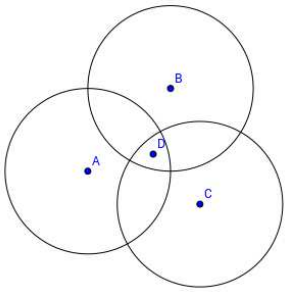
$$\begin{cases} {d_1}^2 = (x_1 - x)^2 + (y_1 - y)^2 \\ {d_2}^2 = (x_2 - x)^2 + (y_2 - y)^2 \\ {d_3}^2 = (x_3 - x)^2 + (y_3 - y)^2 \end{cases}$$

En réalité, si l'on suppose que les trois cercles ont une intersection, il s'agit d'une zone. Cela est illustré ci-dessous par la figure 3.2. Le point D correspond au point calculé dans les deux cas.

Cette méthode nécessite au minimum 3 points de référence. Cependant lorsque les trois cercles n'ont pas d'intersection, cela pose problème pour l'algorithme. Toutefois, il est envisageable d'adpoter des améliorations et des corrections d'erreurs afin d'obtenir des résultats même en absence d'intersection.



(a) parfaite



(b) classique

FIGURE 3.2 – Trilatération

3.3.2 Améliorations et correction des erreurs

Les corrections d'erreurs et illustrations exposées dans cette partie sont celles proposées dans [4].

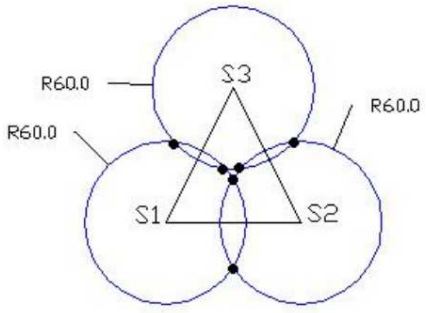
Une des erreurs les plus courantes observée avec la méthode de la trilatération est celle où les cercles ont des intersections deux à deux mais pas d'intersection commune. Il est suggéré dans ce cas de prendre le centre des 3 points les plus proches, comme illustré dans la figure 3.3a.

Une autre erreur courante se produit lorsque un des cercles a une intersection avec les deux autres alors que ces derniers n'en ont pas. Dans ce cas, il est suggéré de prendre, parmi les 4 points d'intersection, le milieu des deux points les plus proches ayant toutefois un seul cercle en commun. Illustré dans la figure 3.3b, le point finalement choisi est le milieu des deux points à l'intérieur du triangle.

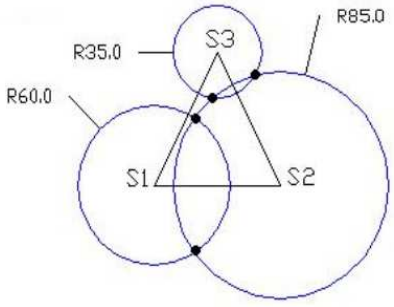
Le cas numéro 3 se produit lorsque seuls deux cercles ont une intersection. Le troisième est soit petit et se trouve à l'intérieur d'un de des autres, soit grand et englobe les deux autres. Il convient dans ce cas de repérer le cercle sans intersection et de le redimensionner en lui ajoutant ou retranchant un δ de sorte à obtenir une intersection avec l'un des des autres. La figure 3.3c illustre ce cas et montre qu'en augmentant légèrement la taille du petit cercle, on se ramène au cas précédent.

Sans intersection du tout comme illustré dans la figure 3.3d, il est conseillé d'augmenter ou diminuer la taille des cercles de manière cohérente jusqu'à temps d'obtenir une intersection et de se ramener à un cas précédent.

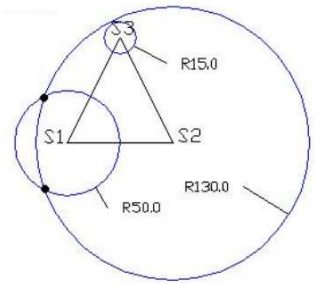
Néanmoins, lorsque les mesures ne sont pas précises ou ne reflètent pas la réalité des choses, ces améliorations peuvent ne pas donner un résultat consistant, notamment dans les cas 3 et 4. C'est pourquoi, ces résultats sont à prendre avec précaution.



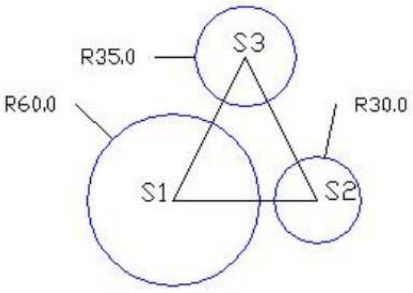
(a) Cas 1



(b) Cas 2



(c) Cas 3



(d) Cas 4

FIGURE 3.3 – Cas d'erreurs de la trilatération, images provenant du document [2]

3.4 Méthode de la multilatération

La multilatération consiste à trouver une position en corrélant les distances estimées par rapport à un ensemble de points de référence (ancres). Dans le cas de 3 ancres, cela revient à la trilatération.

On considère n ancres dont la position est connue, et n distances estimées entre le nœud à localiser, de coordonnées (x, y), et ces ancres.

L'équation correspondante pour l'ancre numéro i est la suivante : ${d_i}^2 = (x_i - x)^2 + (y_i - y)^2$

Il s'agit d'un problème de moindre carrées non linéaire à deux variables et cela revient donc à chercher x et y en minimisant la somme des erreurs de la manière suivante :

$$\min_{n \in N} \left[\sum_{i=0}^{n} e_i \right] \text{ avec } e_i = |{d_i}^2 - ((x_i - x)^2 + (y_i - y)^2)|$$

Ce genre de problème peut être résolu avec l'algorithme de Gauss-Newton ou encore celui de Levenberg Marquardt.

4. Implantation du service de géolocalisation et des modules annexes

Le module de géolocalisation ou plus généralement tout module implanté dans notre système d'information est divisé en plusieurs parties. On a dans un premier temps une partie embarquée dans les capteurs, qui inclut l'envoi des données en réseau jusqu'au sink. Dans un second temps, la partie serveur prend en charge le flux de données, son traitement, et le cas échéant, sa persistance. L'accès aux ressources que ce soit vers ou depuis le serveur se fait par l'intermédiaire d'une API. Nous avions également pour objectif que les différents acteurs composant le système d'information puissent communiquer de manière authentifiée et sécurisée.

4.1 Du réseau de capteurs à la passerelle

4.1.1 L'application embarquée dans les noeuds

Au sein du réseau, tous les nœuds embarquent le même programme sauf le ou les sinks qui rappelons le, mettent à disposition les flux de données reçus des autres capteurs en lecture sur le port USB.

Au sein du noeud, nous avons divisé l'application en modules, exécutés dans un thread principal, indépendants les uns par rapport aux autres. En plus du module de géolocalisation, nous en avons actuellement un qui gère l'acheminement des données provenant des capteurs. Par ailleurs, un autre module, faisant l'objet d'une sous partie ultérieure, permet la prise en charge de commandes afin de contrôler le nœud à distance. Chaque module dispose de ses propres flux réseaux (flux UDP dans notre cas). Ces derniers, utilisant différents ports, permettent au sink d'identifier le flux. Les chaines de caractères contiennent en fait des nombres entiers séparés par des espaces. Au niveau de l'ordonnancement, les modules lisent séquentiellement les événements récents (radio, bouton utilisateur...), et exécutent la portion de code appropriée.

Tout d'abord, la partie géolocalisation embarquée dans les nœuds a pour fonctions de récupérer les données utiles et permettre l'acheminement de celles-ci jusqu'au sink qui fera ainsi le nécessaire pour les transmettre au serveur via la passerelle. Il y a deux étapes principales récurrentes au sein de ce module. Dans un premier temps, les nœuds dialoguent et s'échangent ainsi leurs identifiants ainsi que les indicateurs de force de signal (RSSI) lus depuis la puce radio. L'ID d'un nœud est en fait les 16 derniers bits de son adresse MAC converti en un entier non signé. Ce dialogue se fait en UDP avec du broadcast dont le domaine est dans ce type de réseau la portée d'un nœud, étant donné qu'il faut le considérer comme routeur. Cette portée est conforme à ce que l'on souhaite puisque, pour

avoir des RSSI pertinents entre voisins, il ne faut pas de routage. Une fois qu'un noeud a stocké 10 couples $< ID, RSSI >$, il construit dans un second temps un paquet contenant son ID et ces couples qu'il adresse en unicast au sink dans un flux UDP sur un port défini. Ce paquet est cette fois routé par RPL au sein du réseau.

La chaîne de caractère envoyée sur le réseau a ainsi le format suivant :
`id_noeud nb_couples voisin_1 RSSI_1 voisin_2 RSSI_2 ... voisin_10 RSSI_10`

Par exemple le noeud dont l'ID est 27054 peut envoyer la chaîne suivante :
`27054 10 10008 -55 22234 -68 ... 22005 -60`

Le dialogue entre tous les nœuds voisins permet de couvrir la topologie rapidement. Par ailleurs, nous voulions obtenir du temps réel ou du pseudo temps réel. Ainsi, le temps entre deux broadcasts a été fixé à 1 seconde. Cela permet une assez haute fréquence d'envoi des paquets contenant 10 couples $< ID, RSSI >$. Cependant, si pour diverses raisons (pertes, défaillance) au bout de 30 secondes, un nœud a reçu moins de 10 paquets provenant de ses voisins, on envoie tout de même les couples qu'il a stockés. Il est également possible de solliciter l'envoi de broadcasts des voisins d'un mote en appuyant sur son bouton blanc.

Le sink recevant les données identifie le flux utilisé avec le port local utilisé. Il ajoute au début de cette chaine de caractère un identifiant de flux (10 pour la géolocalisation) et dispose celle-ci sur sa sortie standard qui est ensuite lue par la passerelle.

4.1.2 La passerelle

La passerelle est une application Java et résulte de l'allègement et de la transformation de l'application Collect-view intégré dans Contiki. Créée lors d'un projet précédent, elle permettait initialement le transfert des informations lues sur le port USB sur une URL via des requêtes HTTP. Disposant de plusieurs flux de données, nous l'avons ainsi transformée afin qu'elle puisse en gérer un nombre indéterminé.

La passerelle nécessite les droits super-utilisateur, lit et écrit sur le port USB grâce à des fichiers binaires fournis avec collect-view (compilés à partir de C et de Python). Après lecture d'une chaîne de caractère envoyée par le sink, elle ajoute au début le timestamp et identifie le flux avec l'identifiant ajouté par le sink. Grâce à un fichier de configuration externe, elle fait le lien entre un type de flux et un URL. Elle poste ensuite les données sur l'URL correspondant au type via une requête HTTP. Dans notre cas, les URL du fichier de configuration appartiennent à l'API et le serveur se charge ensuite du traitement de ces flux. Cette nouvelle version de la passerelle permet ainsi la gestion d'un nombre infini de types de flux. Dans cette partie du système d'information, seuls la passerelle et le sink ont la connaissance d'une gestion des types de flux du réseau.

Ainsi, quittant la passerelle dans une connexion TCP, la chaîne de caractère correspondant à la géolocalisation est sous cette forme :

```
timestamp type_flux(=10) chaine_géolocalisation
```

4.1.3 Problèmes rencontrés

J'ai rencontré quelques problèmes lors de la programmation de ce module à bas niveau dans les capteurs. Tout d'abord, la contrainte mémoire m'a forcé à réduire au maximum le code et à enlever tout ce qui n'est pas essentiel. En effet, lors du processus qui consiste à flasher la mémoire morte du nœud, il m'est arrivé plusieurs fois d'avoir un dépassement de mémoire, que ce soit virtuellement dans le simulateur ou réellement. Dans le cadre du travail collaboratif mené lors de ce stage, nous avons également fait face à un dépassement mémoire dû cette fois à l'utilisation de deux versions différentes du compilateur gcc-msp430 [1]. Malgré l'utilisation du même code source, la nouvelle version du compilateur (4.6) ajoutait près de 800Ko en plus par rapport à l'ancienne (4.5). Nous sommes cependant parvenus à condenser le code pour pouvoir compiler avec la nouvelle version.

Par ailleurs, suite à un oubli de programmation dans un autre module, nous avons eu à faire face à une tempête de broadcasts, ce qui engendrait une saturation du réseau et un nombre très important de collisions, et donc de pertes. Pour détecter ceci, j'ai utilisé un Atmel AVR RZ USBstick (cf figure 4.1 ci-dessous) qui est en fait une clé USB équipée d'une puce radio 2.4Ghz et qui, avec un programme spécifique, peut être utilisé comme sniffer sur le réseau IEEE 802.15.4. Ensuite, le logiciel wireshark permet de nous présenter les trames capturées. Nous avons pu identifier, grâce à celui-ci, le module qui posait problème et corriger l'erreur.

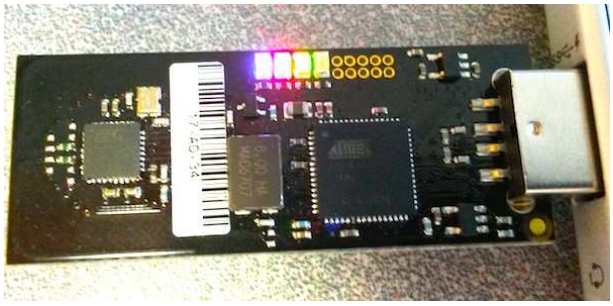


FIGURE 4.1 – Atmel AVR RZ USBstick

4.2 La géolocalisation au niveau de l'application serveur

L'application serveur est exécutée dans un environnement JEE sur le serveur Glassfish 4, et utilise une base de données relationnelle MySQL. Elle est composée d'un noyau, de modules, et de servlets faisant office de points d'entrées dans l'application.

1. Extension du compilateur GNU Compiler Collection pour supporter les micro-contrôleurs MSP430

4.2.1 Architecture globale

Tout d'abord, nous avons choisi de définir le noyau en 3 couches distinctes. Dans un premier temps, il comporte une couche dédiée à la vérification et conversion des flux d'entrée. Son rôle est de vérifier l'intégrité des données contenues dans le flux et de créer à partir de celles-ci les objets correspondants. Précédant la création des objets, la phase de conversion des données est paramétrable depuis l'interface d'administration du site. En effet, il est possible d'appliquer une formule différente à chaque entier contenu dans la chaîne de caractère en fonction de sa position par le biais de filtres et de classes appelées stratégies instanciées dynamiquement selon le filtre choisi.

Suivant la couche précédente, la couche métier est composée des entités et des DAO (Data Access Object). Ces derniers correspondent en fait à un des objets fournissant une interface pour la gestion de la persistance de l'entité à laquelle ils sont associés. Pour cela, les DAO utilisent les annotations EJB (Enterprise Java Beans) contenues dans les entités et dialoguent avec la base de données par l'intermédiaire du manageur d'entité avec des requêtes JPQL (Java Persistence Query Language) ou des requêtes SQL natives. Parmi les entités, voici celles qui sont essentielles au traitement des données :

- Le type d'un flux
- Le libellé représente la sémantique d'un entier au sein des chaînes de caractère d'un flux
- La stratégie représente la classe à instancier dynamiquement lors du processus de conversion
- Le filtre représente la logique de conversion des données. Il lie un libellé, un type de flux, une position et une stratégie de conversion
- Le noeud du réseau

Enfin, il met également à disposition une couche de gestion des membres et d'authentification décrite dans la partie suivante.

Gravitant autour du noyau, les modules actuellement déployés sont les suivants : la géolocalisation, un module gérant les données des capteurs, un module de supervision du réseau, et le module d'administration de la plateforme. Typiquement, la figure 4.2 illustre le processus de collecte des données provenant du réseau de capteurs jusqu'à la réponse du serveur.

L'accès à ces modules et aux ressources du noyau se fait exclusivement par l'intermédiaire d'une API REST, que l'on a très largement étendue par rapport à sa version initiale.

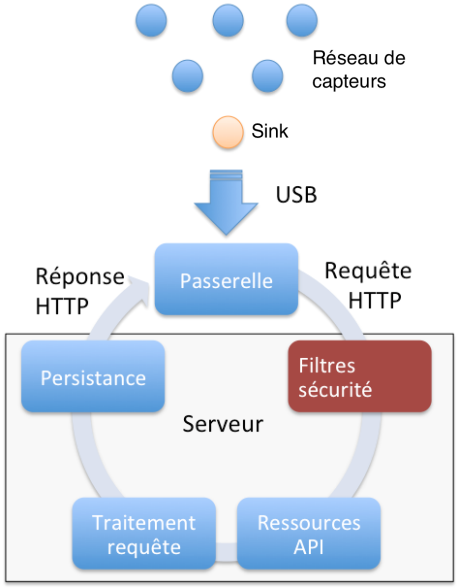


FIGURE 4.2 – Processus de collecte et de traitement des données

4.2.2 Le module de géolocalisation

Une fois les données de géolocalisation prises en charge par le serveur, les algorithmes décrits dans la partie 3 sont appliqués sur les nœuds dont on cherche la position. Pour cela, il est nécessaire de définir des nœuds ancrés dont on connaît la position, et les autres nœuds que l'on cherche à positionner. Au niveau implantation, nous utilisons le patron de conception décorateur pour la différenciation des nœuds en ancre ou sink.

La première étape consiste à calibrer le modèle en calculant l'exposant d'atténuation α, comme détaillé dans la partie 3.1. Le problème de moindres carré associé à cette calibration est résolu grâce à la librairie Apache Commons Math [2]. Ensuite, en prenant pour références les nœuds ancrés, on applique la méthode de la moyenne pondérée (cf 3.2), puis la trilatération avec ses corrections (cf 3.3), et enfin la multilatération (cf 3.4) afin de déterminer la position des nœuds non ancrés. Le problème d'optimisation non linéaire rencontré dans la multilatération est résolu en utilisant de nouveau la librairie Apache Commons Math, et l'algorithme de Levenberg Marquardt.

La calibration et les algorithmes de géolocalisation sont gérés par un gestionnaire qui permet, au travers de l'API, d'activer ou de désactiver leur utilisation. Bien qu'indépendants, ce gestionnaire permet également de combiner les positions obtenu à partir des algorithmes en faisant une moyenne non pondérée, et une autre pondérée avec des coefficients que l'utilisateur peut définir.

Pour permettre aux interfaces graphiques de proposer du temps réel, nous avons choisi d'utiliser des websockets, ce qui permet au serveur de pousser immédiatement les résultats de géolocalisation après calcul. WebSocket est un protocole créé en 2011 et permet un échange full duplex entre un serveur web et un client. Ainsi, les clients web reçoivent les résultats en temps réel et mettent à jour la représentation visuelle. Concernant celle que j'ai mise en place, elle utilise un plan des bureaux où j'ai travaillé et la librairie KineticJS qui permet d'animer et dynamiser une zone d'une page web HTML5. On peut par exemple voir en temps réel les noeuds se déplacer ainsi que la sélection des ancres et les cercles résultant de l'algorithme de trilatération. Une capture d'écran de cette représentation 2D est disponible en annexe E.1, ainsi que du panneau de contrôle (annexe E.2) qui affiche le paramètre de calibration α. Celui-ci permet aussi de décider de quel algorithme on souhaite afficher les résultats, ou encore de démarrer le testeur des algorithmes présenté dans une partie ultérieure.

Pour administrer la plateforme, nous avons mis en place une interface d'administration qui permet la gestion totale de l'application serveur. Concernant la géolocalisation, il est par exemple possible d'ajouter ou de supprimer dynamiquement des ancres, et d'activer ou désactiver les fonctionnalités du module. Une capture d'écran de cette interface d'administration est disponible en annexe E.3.

2. Librairie de mathématiques en Java : http://commons.apache.org/

4.3 Supervision et sécurisation du système d'information

4.3.1 Le système d'authentification

Afin de sécuriser le système d'information, c'est à dire permettre des communications authentifiées et chiffrées, j'ai mis en place différentes mesures de sécurité au niveau du serveur et de la passerelle. La sécurité au sein des réseau de capteurs faisant l'objet de recherche, je n'ai rien ajouté dans cette partie.

Le serveur : Tout d'abord, un système d'authentification permet de sécuriser l'API et l'interface d'administration. Cependant, les différentes pages du site web ne nécessitent pas de comptes. Tous les modifications sur la plateforme permises par l'API requièrent au moins un compte valide, et certaines le titre d'administrateur. Pour cela le site web propose une page où l'on peut se connecter, ainsi qu'une permettant la gestion des membres dans l'interface d'administration. Techniquement, ce filtrage des accès est géré par des filtres nativement intégrés dans la plateforme JEE. Ceux-ci accèdent aux requêtes des utilisateurs avant même les servlets et permettent donc de rejeter les demandes, de modifier les entêtes, de rediriger l'utilisateur, et ceci avant tout traitement par l'application. Au sein de la plateforme, j'ai implanté deux filtres, un pour l'API qui rejète systématiquement toutes les requêtes HTTP POST si l'utilisateur n'est pas connecté et un pour l'administration.

La passerelle : Tout comme un navigateur web, j'ai intégré à la passerelle le support de la connexion et la gestion des sessions. Celle-ci stocke le cookie de session généré par le serveur, et lui renvoie à chaque requête, permettant ainsi à ce dernier d'identifier chaque passerelle connectée. Évidemment, cette fonctionnalité nécessite que la passerelle utilise un compte dont les identifiants sont demandés en ligne de commande au moment de son exécution. Plus encore, nous avions intérêt à ce que la passerelle utilise une connexion full duplex avec le serveur, pour que celui-ci puisse lui envoyer des informations. En effet, il est très probable que les passerelles ne soient pas accessibles depuis l'Internet, étant donné qu'elles sont potentiellement exécutées dans une entreprise, ou un organisme publique où le réseau interne est isolé et protégé. Pour cela, nous avons utilisé une nouvelle fois le protocole WebSocket. Le handshake de ce protocole se fait grâce à l'entête "Upgrade" de HTTP pour ensuite terminer sur un socket au dessus de TCP. Par conséquent, il m'a été possible d'authentifier également les websockets en récupérant l'identifiant de session au moment du handshake, et de savoir ainsi si l'utilisateur est connecté au moment de l'ouverture de connexion websocket.

Enfin, dans un soucis d'authentification du serveur et de chiffrement, la passerelle ainsi que le serveur sont totalement compatibles avec HTTPS et donc également avec WSS (WebSocket Secure). En pratique, grâce à son module proxy, un serveur web frontal tel qu'Apache peut assurer ces communications sécurisées et relayer les requêtes au serveur JEE dans le réseau interne.

4.3.2 Le module de commandes

Comme vu précédemment, les modules et particulièrement le module de géolocalisation sont consommateurs de batterie et n'ont pas vocation à être activés en permanence dans certains contextes. C'est pourquoi, nous avons choisi de d'ajouter un module de commandes au sein du serveur et du réseau de capteurs, nous permettant d'activer, de désactiver, ou de modifier certaines fonctionnalités.

Ainsi, depuis l'interface d'administration, un administrateur peut envoyer des commandes au réseau de capteur. Il peut par exemple activer et désactiver tous les modules ou un en particulier, changer la période d'envoi des données des capteurs, modifier la puissance d'émission de la puce radio, ou encore demander une reconstruction du graphe de routage.

Ces commandes sont envoyées par l'intermédiaire du websocket établi entre la passerelle et le serveur, transférées sur le port USB et interprétées par le sink. Au niveau du réseau, le sink envoie les commandes en multicast, couvrant ainsi la globalité du réseau, puisque tous les nœuds se sont préalablement inscrits dans le groupe multicast.

Ce module est critique car il permet d'engendrer des actions directes dans le réseau de capteurs sans fil. Il a directement motivé le développement des fonctionnalités de sécurité sur la plateforme. Par conséquent, il ne fonctionne que si la passerelle est authentifiée, et que le websocket établie entre celle-ci est le serveur l'est également.

Liens

Le code source des différentes composantes de l'architecture est disponible sur un dépôt public : https://github.com/AnthonyDeroche/iotlab

De même, un tutoriel de configuration et d'installation de l'application serveur est mis à disposition sur une page web : https://anthonyderoche.github.io/iotlab/

Enfin, nous avons déployé l'application sur un serveur accessible à des fins de démonstration : http://www.anthonyderoche.com/telecomnancy/iotlab/home

5. Evaluation des algorithmes

Une fois implantés, nous avons voulu tester les algorithmes de géolocalisation afin de les comparer et de connaître leur précision.

5.1 Réalisation des tests

La réalisation des tests consiste pour une même topologie des nœuds du réseau (ancres et nœuds à localiser), à relever les positions des nœuds résultant des trois algorithmes utilisés et à les comparer avec la position réelle du nœud. Le testeur considère les 50 dernières valeurs calculées pour chaque algorithme et calcule la moyenne de l'erreur qui est en fait la distance entre le point réel et le point trouvé.

Afin de faire ressortir un algorithme ou d'en éliminer un, j'ai ajouté à ce testeur un optimiseur qui combine les algorithmes avec une moyenne pondérée sur les coordonnées. L'optimiseur résout un problème linéaire et trouve les meilleurs poids possibles dans cette moyenne pour se rapprocher au maximum de la position réelle. Une fois les coefficients trouvés, nous changeons la position des nœuds à localiser pour observer si l'amélioration perdure.

Nous avons voulu soumettre les algorithmes à deux topologies différentes. La première est composée de nœuds ancrés entourant entièrement la zone dans laquelle les noeuds à localiser sont disposés. Contrairement à la première, la seconde comporte les nœuds à localiser à l'extérieur de la zone délimitée par les nœuds ancrés. Cette seconde topologie permet de mettre en évidence la possibilité ou non d'extension géographique du réseau au niveau géolocalisation.

L'environnement de test : Il s'agit d'une zone de près de 400 mètres carrés comportant des bureaux et des couloirs. Il est essentiel de préciser que cet environnement intérieur est très hostile à la propagation des ondes ce qui engendre des valeurs de RSSI peu fiables. En effet, au delà de la présence de murs, les ondes électromagnétiques souffrent de diffraction, réflexion, de perturbations entraînées par les trajets multiples, et cela se manifeste par un signal bruité, même pour des nœuds géographiquement proches. Par conséquent, il est indispensable de prendre les résultats présentés par la suite avec précaution.

5.2 Comparaison des résultats

Concernant la topologie où les nœuds à localiser sont à l'intérieur de la zone délimitée par les nœuds ancrés, on constate que la méthode de la moyenne pondérée est meilleure que les autres algorithmes. Elle nous offre une précision entre 2.5m et 3m alors que les deux autres algorithmes sont à plus de 4m.

En revanche, concernant la seconde topologie, la moyenne pondérée est la plus mauvaise. En effet, sa définition mathématique montre qu'elle ne peut trouver que des positions à l'intérieur du polygone délimité par les ancres pour des poids de même signe. La multilatération est la meilleure pour cette topologie, avec une précision d'un peu plus de 3m.

Néanmoins, la trilatération n'est pas à exclure, se plaçant deuxième dans les deux cas, elle peut être utile si la topologie des ancres par rapport à l'environnement à géolocaliser est complètement inconnue.

La combinaison des algorithmes ne s'est pas révélée meilleure, puisqu'elle a été faite de manière naïve. Il convient certainement de revoir cette partie dans un travail futur afin de combiner les algorithmes de manière intelligente en fonction du cas à traiter. De même l'optimiseur n'a pas exclu d'algorithme précis et a conforté nos constatations en ignorant davantage les algorithmes mauvais dans les deux topologies.

Les tests effectués nous donnent des indications intéressantes mais sont insuffisants pour tirer des conclusions fermes. C'est pourquoi, nous projetons de continuer les tests dans un environnement plus vaste à partir de septembre 2014 dans le cadre d'un déploiement du réseau dans TELECOM Nancy.

Les résultats obtenus sont disponibles ci-dessous sous forme d'histogrammes :

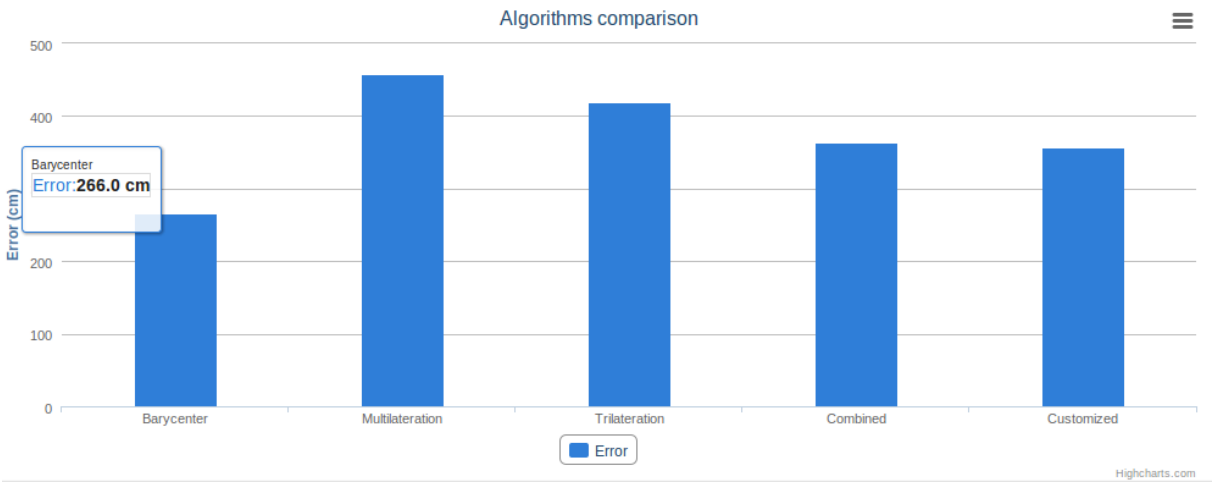


FIGURE 5.1 – Noeuds mobiles à l'intérieur des ancres

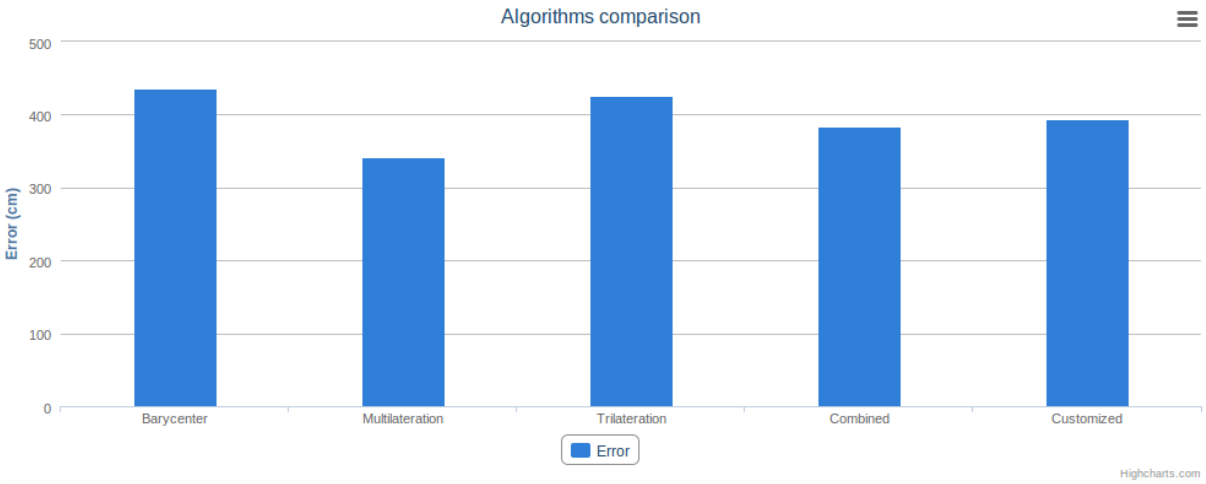


FIGURE 5.2 – Noeuds mobiles à l'extérieur des ancres

Conclusion

Du logiciel embarqué jusqu'au développement logiciel en passant par les réseaux, ce stage techniquement très riche a été pour moi l'occasion de combiner mes compétences pour la mise en œuvre d'une plateforme de gestion d'un réseau de capteurs sans-fil. Supervision, sécurité mais aussi géolocalisation sont des fonctionnalités attendues lorsque l'on parle de réseau de capteurs sans-fil. Fruit d'une collaboration quotidienne, nous avons enrichi les différentes composantes de notre architecture de toutes ces fonctionnalités. Par ailleurs, nous avons attaché une grande importance à la modularité et à l'évolutivité de la plateforme web, laquelle, fournissant les données par l'intermédiaire d'une API, s'inscrit bien dans la tendance des Open Data. Concernant la géolocalisation, nous avons obtenu des résultats satisfaisants alors même qu'un environnement intérieur est inexorablement hostile à la propagation uniforme des ondes et donc à l'obtention d'indicateurs fiables basés sur la force du signal.

Outre l'aspect technique, ce projet m'a aussi confronté aux avantages et difficultés du travail collaboratif, où la communication est essentielle, et la gestion des versions du travail effectué est indispensable. Les développements effectués au sein de cette architecture ont permis de renforcer mes compétences en génie logiciel, particulièrement par l'utilisation de la technologie Java EE ainsi que du langage C dans les environnements contraints en mémoire. De plus, le développement de fonctionnalités avancées au sein du réseau de capteurs ont nécessité l'étude et la compréhension du système d'exploitation embarqué ContikiOS et des principaux protocoles réseau utilisés.

Dans une perspective de travaux futurs, il serait souhaitable d'étudier la possibilité de combiner intelligemment les algorithmes de géolocalisation voire de les enrichir. De même, nous pourrions trouver intérêt à la possibilité de gérer plusieurs réseaux indépendants ou instances du même réseau au sein de la plateforme, fonctionnalité qui nécessiterait de retoucher très légèrement le modèle. Par ailleurs, la plateforme peut s'inscrire dans la tendance de favoriser la science reproductible en informatique. En effet, à travers l'API, d'autres chercheurs peuvent implanter leurs propres algorithmes et les confronter aux positions réelles de notre testbed. Pour finir, dans un contexte de traitement de gros volume de données, il faut envisager une compatibilité de l'application web avec un modèle de persistance adapté pour le Big Data.

Bibliographie / Webographie

[1] Eduardo F. Nakamura Antonio A. F. Loureiro Azzedine Boukerche, Horacio A. B. F. Oliveira. Localization systems for wireless sensor networks. *IEEE Wireless Communications*, dec 2006. 7

[2] Stefano Chessa Member IEEE Gaetano Giunta Member IEEE Paolo Barsocchi, Stefano Lenzi. A novel approach to indoor rssi localization by automatic calibration of the wireless propagation model. *Vehicular Technology Conference, 2009. VTC Spring 2009. IEEE 69th*, apr 2009. 7, 11, 24

[3] Stefano Chessa Gaetano Giunta Paolo Barsocchi, Stefano Lenzi. Virtual calibration for rssi-based indoor localization with ieee 802.15.4. *Communications, 2009. ICC '09. IEEE International Conference on*, jun 2009. 7

[4] Ishak Abdul Azid Marjan Moradi Zaniani, Nibong Tebal. Trilateration target estimation improvement using new error correction algorithm. *Electrical Engineering (ICEE), 2010 18th Iranian Conference on*, may 2010. 10

[5] Christian Appold Martin Schröder Florian Füller Marcel Baunach, Clemens Mühlberger. Analysis of radio signal parameters for calibrating rssi localization systems.

[6] Andrea Zanella Giovanni Zanca, Francesco Zorzi and Michele Zorzi. Experimental comparison of rssi-based localization algorithms for indoor wireless sensor networks. *Proceeding REALWSN '08 Proceedings of the workshop on Real-world wireless sensor networks*, apr 2008.

[7] M. Kanaan and K. Pahlavan. Algorithm for toa-based indoor geolocation. *ELECTRONICS LETTERS*, 40(22), oct 2004.

[8] Hongyu Shi. A new weighted centroid localization algorithm based on rssi. *Information and Automation (ICIA), 2012 International Conference on*, jun 2012.

[9] Contiki OS. http://www.contiki-os.org/ http://github.com/contiki-os/.

[10] Commons Math : The Apache Commons Mathematics Library. http://commons.apache.org/proper/commons-math/.

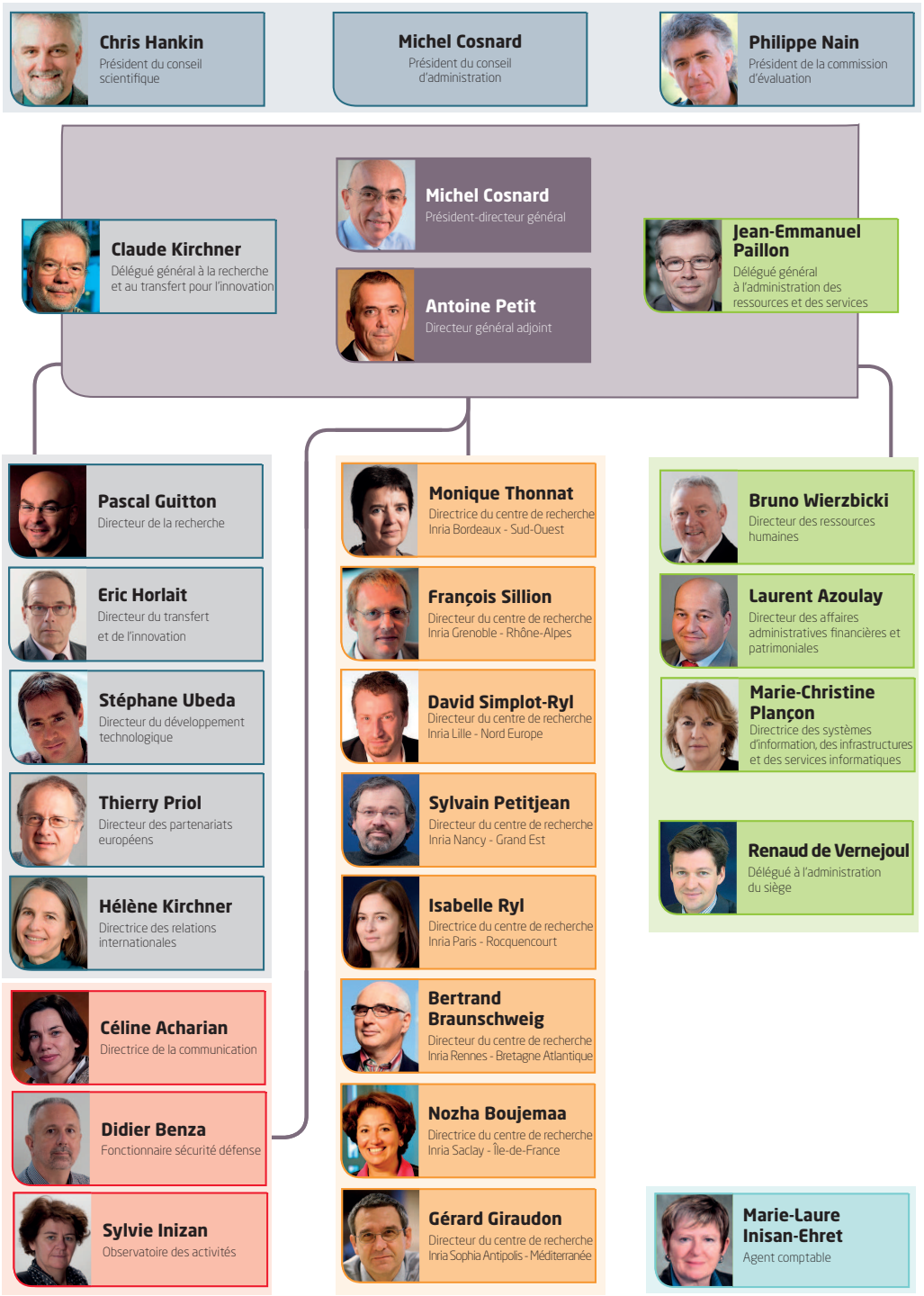
[11] Oracle Java EE 7 documentation. http://docs.oracle.com/javaee/7/api/.

[12] Texas Instruments 2.4 GHz IEEE 802.15.4 / ZigBee-Ready RF Transceiver CC2420 (Rev. C). http://www.ti.com/lit/gpn/cc2420.

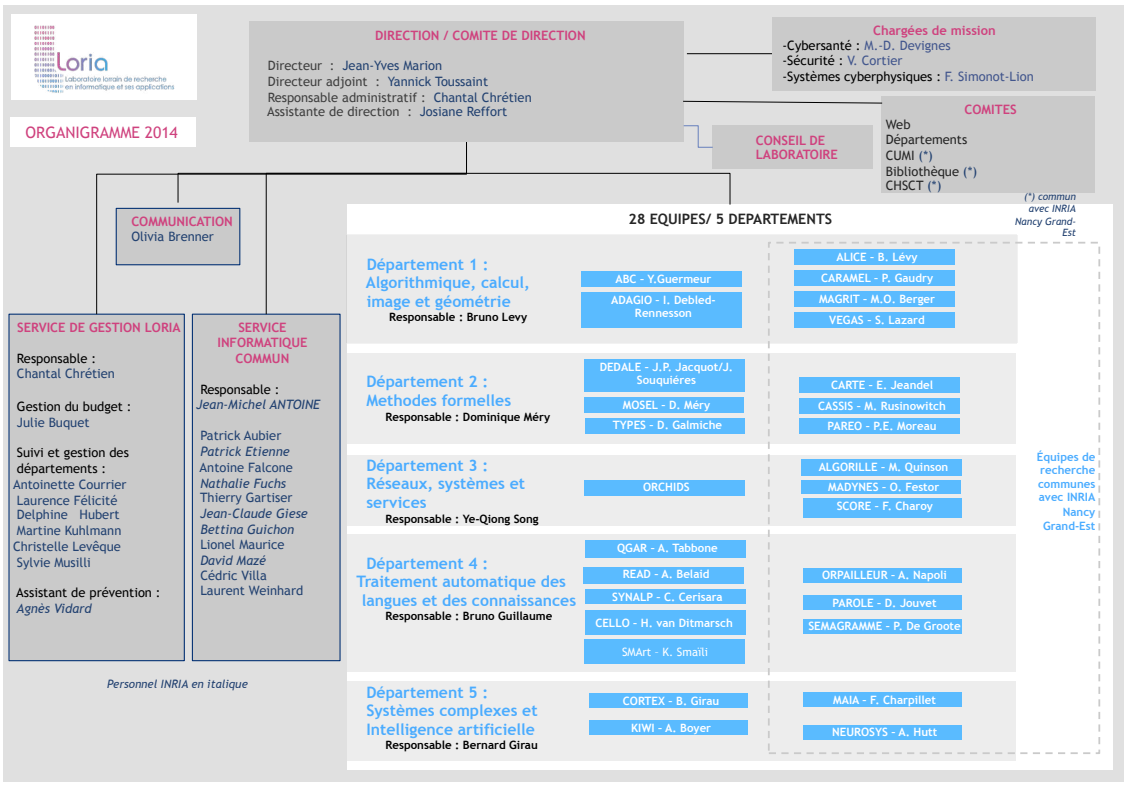
Liste des illustrations

Annexes

A. Organigramme Inria



B. Organigramme LORIA



C. Le simulateur Cooja

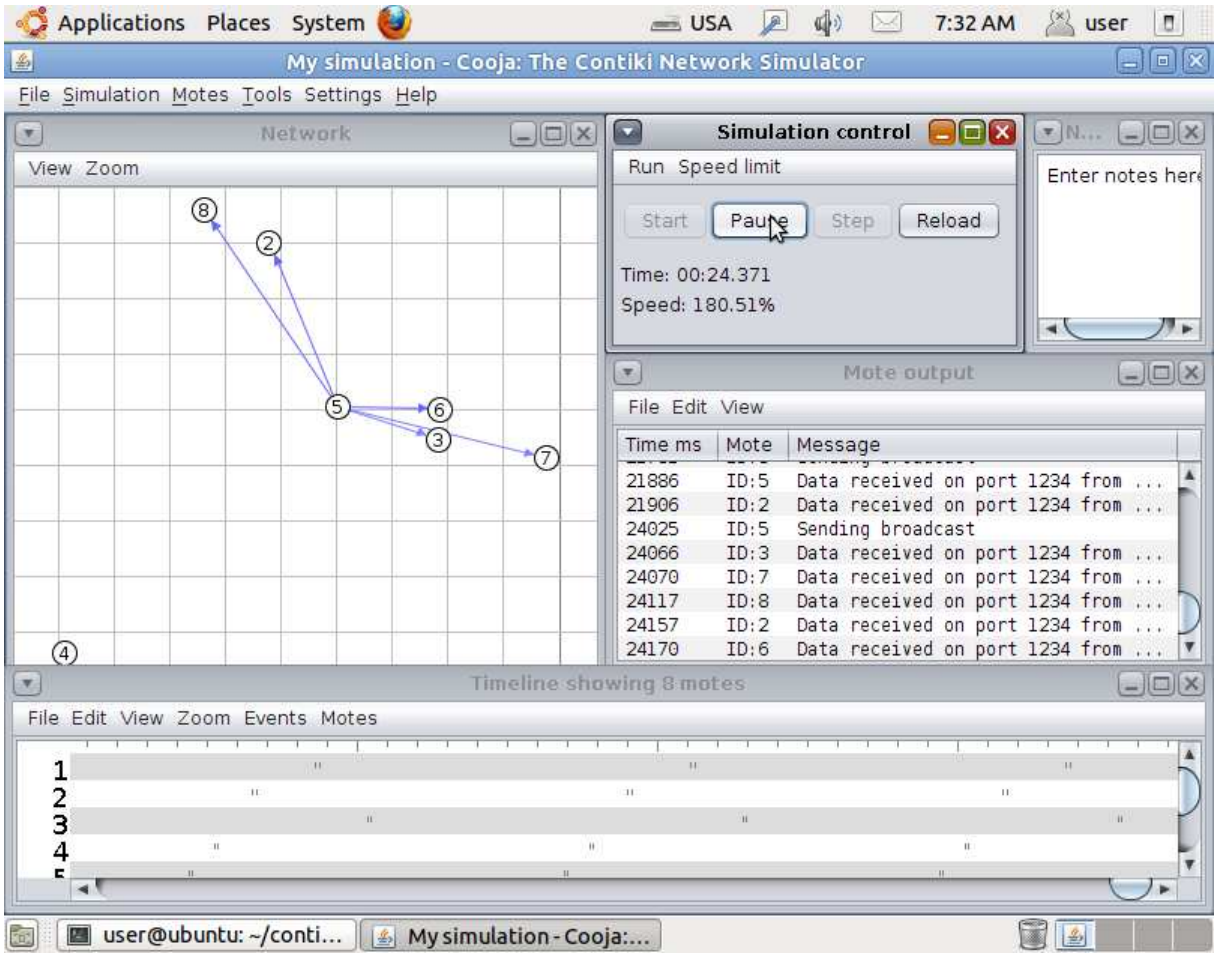


FIGURE C.1 – Capture d'écran de Cooja

D. Déploiement du réseau

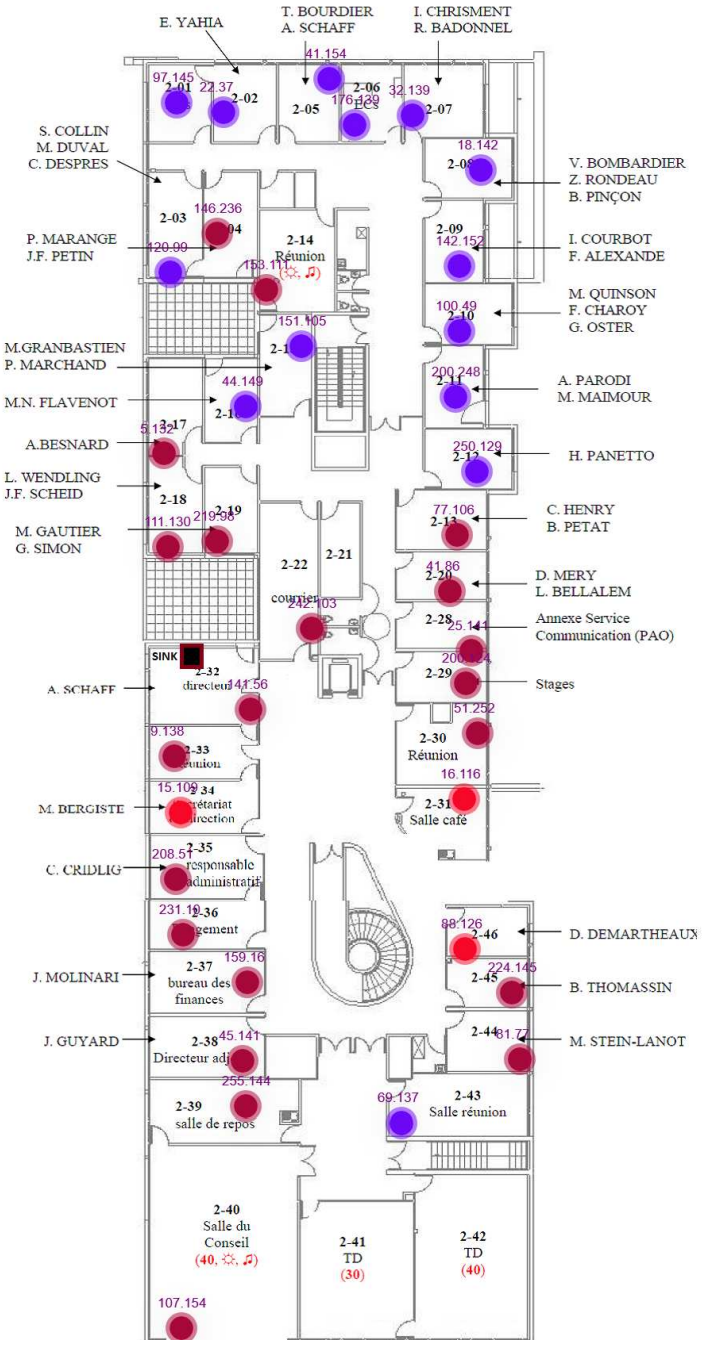


FIGURE D.1 – Carte de la disposition des capteurs colorés selon la température relevée, lors d'un déploiement à TELECOM Nancy

E. Les interfaces graphique de géolocalisation

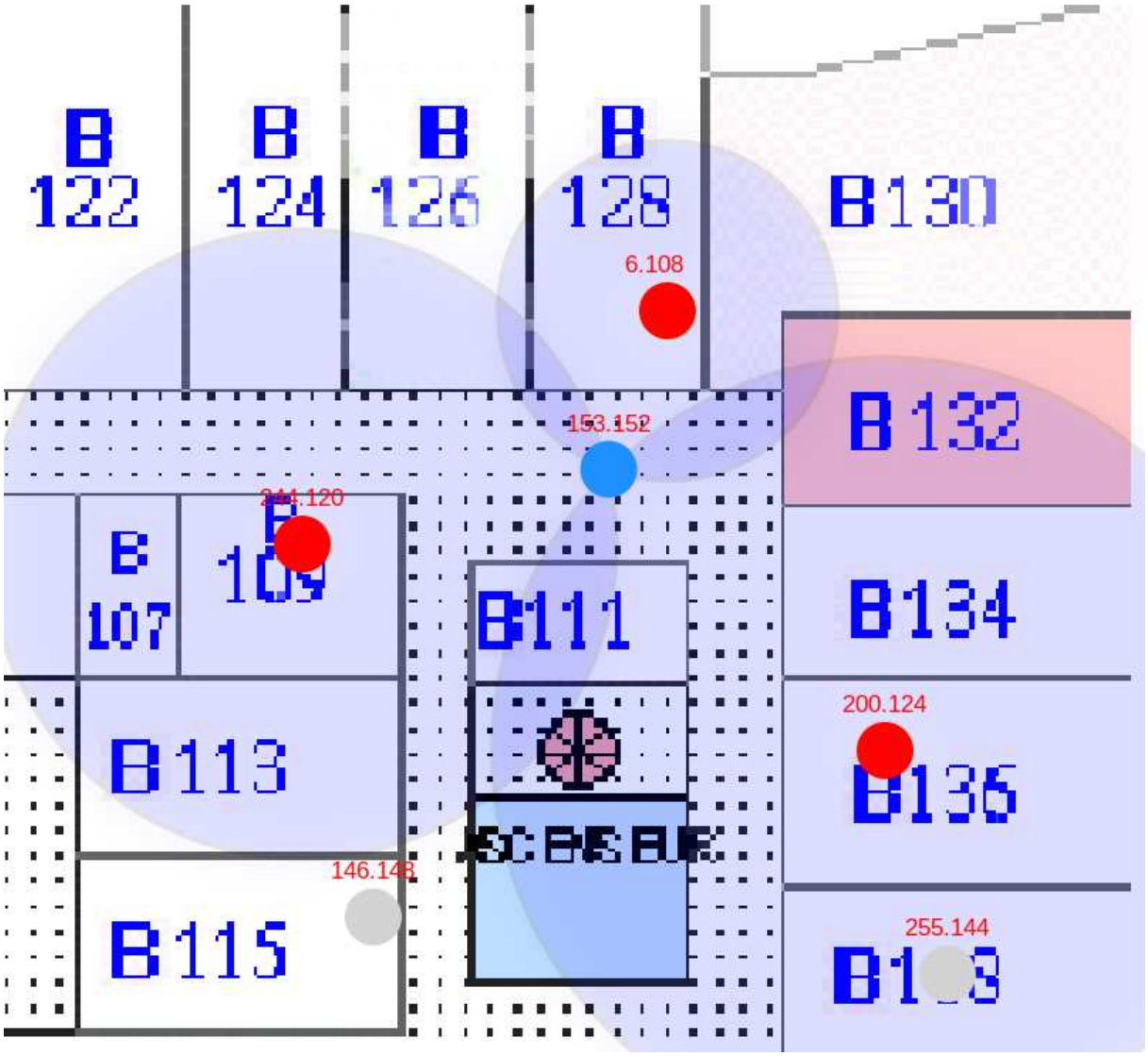


FIGURE E.1 – Représentation visuelle de la trilatération sur le plan des bureaux de MADYNES. Les ancres sont grises, les selectionnées sont rouges, et le noeud à localiser est bleu.

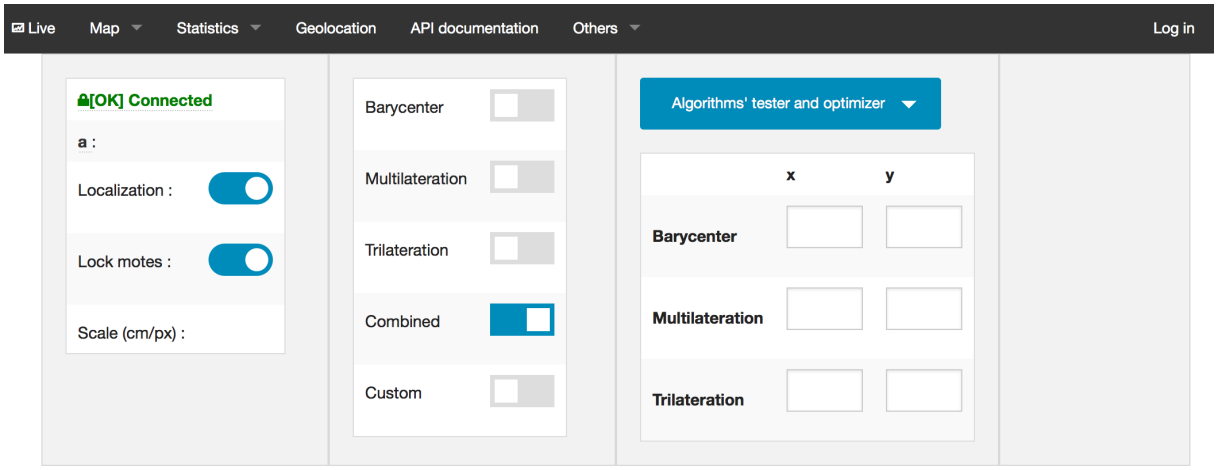


FIGURE E.2 – Panneau de contrôle du client web

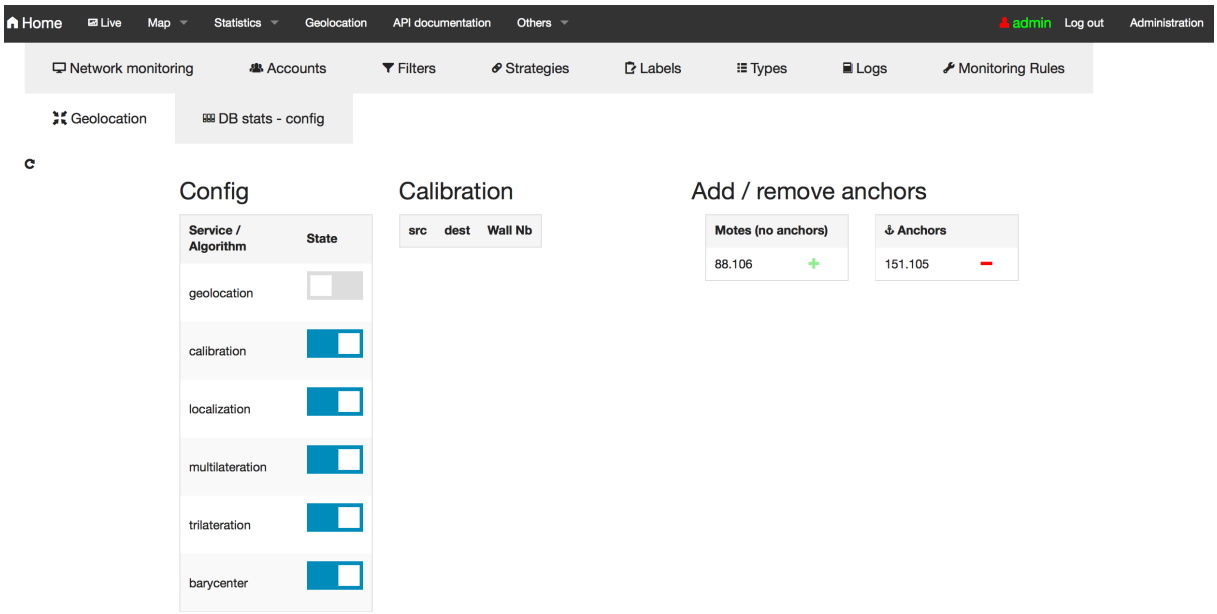


FIGURE E.3 – Panneau d'administration de la géolocalisation