Ecole des Mines de Saint Etienne

Rapport de Stage

Amélioration des performances d'un réseau de capteurs sans fils grâce aux techniques de fouille de données

Gaëtan HEIDSIECK
Promo 2017

Tutrice d'entreprise : Mme. MARC-ZWECKER
Tutrice académique : Mme. BORODIN

28 Février 2017 — 28 Août 2017

Remerciements

Je remercie M. Michel de Mathelin, directeur du laboratoire ICube de m'avoir accepté pour effectuer mon stage de fin d'étude. Je tiens à remercier particulièrement Mme Stella Marc-Zwecker, pour son accueil et son soutien au cours de mon stage. Je remercie également M. Gabriel Frey et M. Fabrice Theoleyre pour leur implication et leur aide, ainsi que Mme Florence Le Ber et l'ensemble de l'équipe sciences des connaissances pour leur bon accueil et l'équipe réseau pour leur collaboration.

Je remercie M. Philippe Beaune, directeur du cycle ISMIN pour m'avoir aidé à trouver mon stage ; ainsi que Mme Valéria Borodin pour avoir accepter d'encadrer et de m'avoir conseillé au cours de mon stage.

Table des matières

Chapitre 1

Introduction générale

1.1 Contexte du Stage

Mon stage se déroule au sein d'iCube, le centre de recherche du CNRS et de l'Université de Strasbourg. Je travaille dans le Département Informatique Recherche (D-IR), conjointement avec deux équipes du département :

- L'équipe Science des Données et Connaissances (SDC) pour la partie fouille de données où je suis encadré par Madame Marc-Zecker, Monsieur Frey et Madame Le Ber
- L'équipe Réseaux (R) pour la partie réseau où je suis encadré par Monsieur Theoleyre

1.2 Sujet de l'étude

Exploitation de jeux de données expérimentales issues de réseaux de capteurs sans fil, au moyen de techniques de fouille de données et de machine learning. Ces données, qui devront être analysées selon leur composante temporelle, sont relatives à la qualité des liens, aux informations de routage et de trafic. L'objectif est d'effectuer un apprentissage sur la corrélation des données, afin de définir des algorithmes de diagnostic de pannes et/ou de prédiction de performances.

1.3 Thématiques abordées

J'ai travaillé sur l'analyse et la préparation de données provenant de logs d'un réseau de capteurs dans l'optique d'étudier deux problématiques distinctes :

1. Le taux de livraison entre capteurs
2. L'efficacité du taux de réservation de créneau de communication entre les capteurs.

Les données proviennent d'expérimentations spécifiques utilisant le réseau de capteurs du site de Grenoble.

J'ai développé des outils Python qui extraient les informations pertinentes des logs générés par ces expérimentations. J'ai également testé des algorithmes de fouilles de données.

1.4 Plan du rapport

Après un rappel de l'état de l'art dans les domaines de l'internet des objets et de la fouille de données, je présenterai la plateforme IoTLAB utilisée au Laboratoire ICube pour les expérimentations de l'internet des objets.

Je décrirai les données extraites de ces expérimentations et le travail de préparation que j'ai effectué sur ces données.

Je présenterai à la fois les problématiques, les outils développés et leurs limitations, ainsi que les challenges rencontrés lors de mon stage.

Chapitre 2

Etat de l'art

2.1 L'internet des objets

2.1.1 Présentation générale

L'Internet des objets (IdO ou IoT pour Internet of Things en anglais) représente l'extension d'Internet au monde physique. Plus précisément, l'internet des objets permet les échanges d'informations et de données provenant de dispositifs présents dans le monde réel vers le réseau Internet. L'internet des objets fait suite à l'ère du Web social comme la troisième évolution de l'Internet et s'applique à de vastes applications des objets connectés. Par exemple, dans le domaine de la e-santé, de la domotique ou du coaching personnalisé, l'internet des objets joue un rôle primordial dans l'accroissement exponentiel du volume de données générées sur le réseau (Big Data).

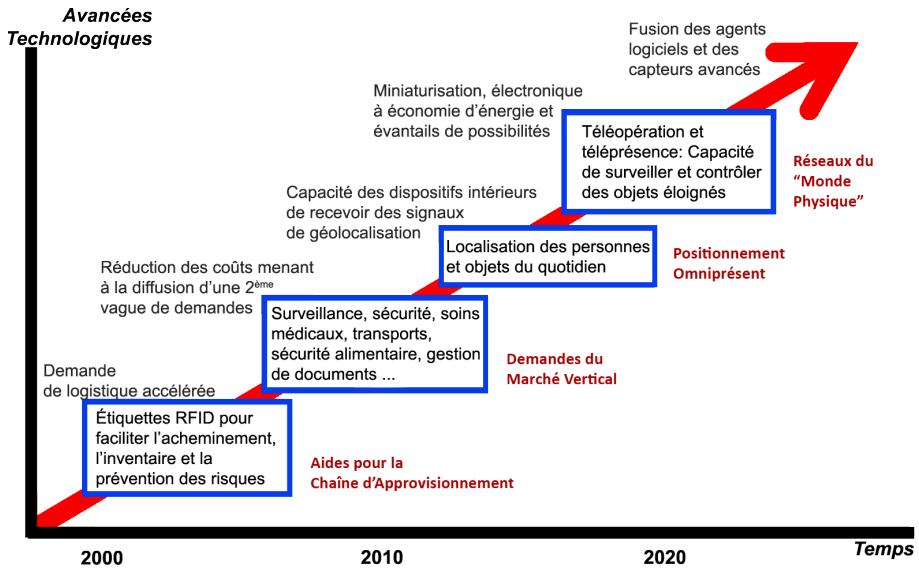

FIGURE 2.1 – Historique de la connectivité des choses

Source: SRI Consulting Business Intelligence

La première utilisation de l'IoT a été dans les années 2000 les tags RFID pour le milieu industriel, afin de suivre et d'inventorier des produits. Par la suite les tags RFID se sont médiatisés et sont présents dans de nombreux autres domaines tels que les cartes de transport et la sécurité

(fig. 2.1). L'internet des objets ne se limite plus aux tags RFID. En effet avec la mise en réseau de microprocesseurs potentiellement associés à des capteurs, les éléments de l'IoT deviennent intelligents.

2.1.2 Présentation des réseaux de capteurs sans fil (WSN)

Un WSN (Wireless Sensor Network/ réseau de capteurs sans fils) est un réseau de capteurs, dans notre étude, associés à des microcontroleurs autonomes appelés nœuds. Ces nœuds mesurent et transmettent des données vers un nœud définit comme puits du réseau. Les données peuvent ensuite être traitées et récupérées par un utilisateur. Les WSN appartiennent à une partie récente de l'IoT où chaque nœud du réseau peut être doté d'une "intelligence", ce qui permet l'application de protocoles réseaux dynamiques.

Les WSN sont utilisés dans de nombreux domaines, par exemple :

- Les réseaux de capteurs sans fils permettent de récupérer des données répartie sur un champs large du fait de leur caractère sans fils. Cela permet de surveiller et d'analyser des environnements peu accessibles tels que des volcans, des larges forets, ou des îles [1] [2].
- Les WSN sont aussi utilisé dans la surveillance de personnes dans leur maisons. Il est en effet possible de surveiller le comportement d'enfants en bas age [3], ou de personnes âgés [4].

Ces applications créent un volume important de données temporelles, dynamiques, hétérogènes et géographiquement distribuées.

2.1.3 Architectures et Protocoles

Le développement de l'Internet des Objets demande la définition de modèles de référence. Il convient de décrire la manière avec laquelle ces systèmes, ces réseaux et ces applications interagissent entre eux. Le bénéfice de tels modèles est de simplifier la compréhension de systèmes complexes découpés en parties plus compréhensibles, de clarifier à l'aide d'une terminologie commune l'apport d'informations supplémentaires identifiant les niveaux de l'IoT, d'identifier des types spécifiques de traitement optimisés dans les différentes parties du système et finalement de standardiser pour créer les conditions d'une interopérabilité entre des produits IoT de différents fabricants. La définition de tels standards fait l'objet d'une activité intense (compétition) entre les grands acteurs du domaine (e.g. Cisco : modèle à 7 couches ; eclipse MQTT & CoAP, ...).

Des exemples d'architectures et de protocoles sont fournis en annexe.

2.2 Fouille de données

2.2.1 Principes

La fouille de données est l'action d'extraire des informations non triviales à partir de données. C'est une discipline très large et qui peut s'appliquer à de nombreux domaines [5]. La fouille de données s'appuie beaucoup sur des techniques d'apprentissage automatique qui s'interprète de la façon suivante :

- Le développement de modèles informatiques des processus d'apprentissages dans le but de répondre à la problématique de prédire l'évolution d'une situation à partir des expériences passées [6].
- L'utilisation de modèles statistiques pour améliorer les performances d'une machine en révélant des liens entre les données ou des patterns dans des données d'apprentissage [7].

Les approches de fouille de données les plus connues sont la classification, le clustering et les règles d'associations.

- Les techniques de classification permettent de prédire l'appartenance d'un élément à un groupe. En analyse de données et en reconnaissance de modèle, un classificateur est une fonction qui assigne une étiquette à des instances décrites par un ensemble d'attributs. Ce sont des techniques d'apprentissage supervisé, et nécessite un jeu d'apprentissage. Dans le domaine de la santé, ce genre d'algorithme peut être utilisé pour prédire des cancers, des échantillons d'ADN sont analysés et classifiés par des combinaisons de classificateurs [8].
- La clusterisation est la division de données en groupes similaires. Elle a pour but de regrouper un ensemble de données en différents groupes homogènes. les algorithmes minimisent la distance intra-classe et maximise la distance interclasse afin d'obtenir des sous ensembles les plus distincts possible. Elle peut être vue comme une technique pour rendre les données plus concises, plus simples [9]. C'est de l'apprentissage non supervisé.
- Les règles d'associations sont des règles extraites de bases de données qui décrivent des liens entre certains éléments. Cette technique permet de faire ressortir les associations entre les données. Plus précisément les règles d'associations sont une expression X=>Y avec X et Y des jeux d'objets. Les objets caractérisés par X possèdent probablement aussi Y[10].

2.2.2 Utilisation pour les WSN

Au cours de ces dernières années, la fouille de donnée par techniques d'apprentissage automatique est devenu un axe important pour l'étude et la mise en place des WSNs. Les algorithmes d'apprentissage sont applicables pour les WSN dans des domaines tels que :

- La fiabilité des données [11][12] : pour travailler sur des données il est nécessaire de les préparer, par exemple gérer des valeurs manquantes, le bruit et les données aberrantes. La fouille de données permet par exemple de prédire une valeur pour compenser une valeur manquante. Ce type d'algorithme est très pratique dans les WSNs où la fiabilité peut être mauvaise.
- L'agrégation de données et le clustering de nœuds [13] [14][15] : dans les WSN, une problématique clef est la gestion des restrictions de ressources : taille de mémoire, batterie limitée, bande passante disponible. Le clustering de nœud ou de données permet de réduire ces contraintes de ressources grâce à une structure hiérarchique des nœuds. Les flux d'information sont rationalisés en utilisant des notions de voisinage et de chef de cluster.

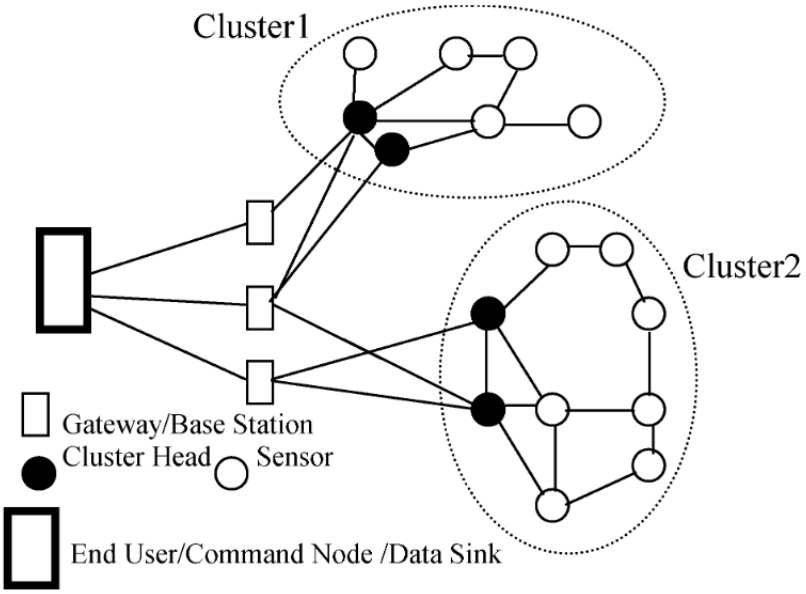

FIGURE 2.2 – Exemple de clusterisation de noeuds

Source: Computing reliability and message delay for Cooperative wireless distributed sensor networks subject to random failures - H.M.F. AboElFotoh ; S.S.Iyengar ; K. Chakrabarty

- La localisation, pour déterminer les coordonnées géographiques des nœuds du réseau. C'est une information importante dans de nombreux réseaux car les données récupérées par un capteur dépendent de leur position. En fonction du type de localisation utilisée (RFID, WLAN, GPS, ...), il existe des tables de comparaison de différents algorithmes de fouille [16].
- Le routage [17][18]. Concevoir un protocole de routage est une problématique de base des WSN. Il faut prendre en compte les limites du réseau telles que la consommation d'énergie, la grande surface du réseau, la tolérance de faute. Les algorithmes de fouille de données se prêtent bien à ces problématiques. Dans [19] les auteurs mettent en avant une méthode basée sur l'algorithme de l'arbre recouvrant qui permet un routage efficace pour des réseaux de nœuds importants.
- La sécurité [20]. L'apprentissage automatique permet de détecter les données aberrantes et les comportements anormaux, ce qui rend la discipline efficace pour gérer des problèmes de sécurité.

Les méthodes de fouille utilisées pour les WSN sont limitées par les conditions du réseau : sources d'alimentations limitées, important nombre de nœuds, taux de perte élevé. Dans le cas de l'IoTLAB, le réseau est distribué et les capacités de calcul sont limitées.

2.2.3 Quelques outils intéressants

Arbre de décision :

L'arbre de décision est une méthode de classification qui repose sur une approche dite "diviser pour régner". L'arbre de décision en apprentissage automatique prend un ensemble de données caractérisés par des variables, il teste les valeurs des variables qui permettent de séparer les données au mieux. Le principe est de représenter un ensemble de choix sous la forme d'un arbre. Chaque embranchement est un choix et les différentes branches correspondent à une décision. Il permet de découvrir des caractéristiques, extraire des modèles [21].

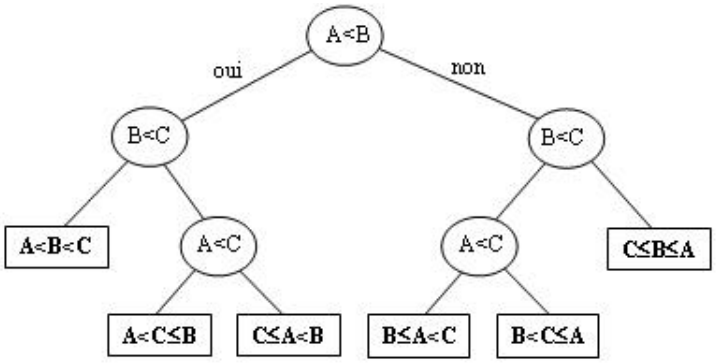

FIGURE 2.3 – Exemple d'un arbre de décision pour un algorithme de tri

Source: Cours de l'upmf de Grenoble

Chaines de Markov :

D'autres méthodes utilisées sont les techniques dérivées des réseaux bayésiens : les chaînes de Markov et les modèles de Markov cachés (Hidden Markov Model/HMM).

Un réseau bayésien représente des variables aléatoires sous forme de graphe orienté acyclique. Il est défini par la description qualitative des dépendances entre des variables (un graphe orienté acyclique), et la description quantitative de ces dépendances (les probabilités de chaque nœud conditionnellement à l'état de ses parents).

Les réseaux bayésiens permettent de représenter l'incertain et de raisonner à partir de données incomplètes. Ce type d'algorithme est utile notamment pour la génération de liens entre les sites web, méthode utilisée par les moteurs de recherches [22].

Un modèle de Markov caché est défini par :

- Le nombre d'état possible du modèle.
- Les probabilités de l'état initial.
- Les probabilités d'émission (émettre une valeur donnée pour chaque état).
- Les probabilités de transition d'un état à l'autre.

On distingue plusieurs usages de l'apprentissage automatique sur modèles de Markov cachés en fonction de l'objectif :

- Affiner ou former un modèle pour qu'il corresponde à un jeu de données.
- Déterminer l'enchaînement d'état le plus probable dont provient une séquence d'émission donnée

Les modèles de Markov sont applicables à une situation si le principe de Markov est respecté : la probabilité d'occurence d'un état dépend uniquement de l'état précédent. Dans le cadre des réseaux de capteurs sans fils on peut faire l'hypothèse que ce principe est respecté car il y a une indépendance des états précédent.

Chapitre 3

Plateforme IotLAB

3.1 Présentation de la plateforme

IoT-LAB est une plateforme d'expérimentation pour effectuer des tests sur les relations et la mise en réseau d'appareils sans fils et d'objets en communication hétérogène. L'ensemble d'un microcontrolleur et d'un capteur est appelé nœud. La plateforme est constituée de quelques milliers de nœuds. Elle est distribuée sur plusieurs sites : Grenoble, Lille, Strasbourg, Saclay, Rennes, Paris, Lyon et Berlin [site web IotLAB].

La plateforme propose différents types de nœuds :

- Les WSN430, ils sont basés sur le MCU(microcontrôleur) MSP430F1611. Ils fonctionnent en consommation faible d'énergie.
- Les M3, basés sur le MCU ARM Cortex M3. Ils possèdent plus de capteurs que les autres types de nœuds.
- Les A8, qui sont les nœuds les plus puissants. Ils possèdent un processeur ARM Cortex-A8 combiné à un MCU ARM Cortex M3, qui leur permet de faire tourner des OS de haut niveau.

Les trois types de nœuds possèdent une interface radio et un ensemble de capteurs (température, luminosité). Ils sont tous programmables et accessibles en temps réel.

- Il existe aussi des nœuds mobiles, attachés à des robots avec des trajectoires prédéfinies.

La plateforme propose aussi différents environnements et topologies selon les sites :

- À Grenoble les nœuds sont aménagés dans le faux plafond et dans le sol du bâtiment de INRIA Grenoble. Ce qui améliore le "réalisme" des expérimentations.
- La plateforme de Strasbourg et de Lille possède le plus de nœuds mobiles.

Chaque plateforme a un jeu de nœud et une topologie unique, et permet de simuler des cas "réels".

3.2 WSN dans IoTLAB

Description des protocoles (routages, mac, ...) et définition de cellule.

3.2.1 Description des protocoles utilisés

Les divers protocoles utilisés permettent de définir le comportement de chacune des couches d'abstraction du réseau, des couches physiques jusqu'aux couches d'applications du réseau. C'est un domaine complexe.

Je détaille ici uniquement les protocoles d'intérêt dans le cadre du projet :

- 6LoWPAN : 6LoWPAN est un acronyme de IPv6 over Low Power Wireless Personal Area Networks. C'est une couche d'adaptation pour IPv6 sur les liens IEEE 802.15.4. Ce protocole opère pour une gamme de fréquence autour 2.4 GHz avec un taux de transfert de 250kbps. Il gère l'encapsulation et compression d'en-tête de paquets IPv6, il transmet les paquets IPv6 de la couche Réseau vers son équivalent sur l'équipement distant dans des trames 808.15.4
- RPL (IPv6 routing for low power/lossy networks) : Protocole de routage qui définit les arborescences de nœuds de manière dynamique. Définition du « Puits », nœud principal qui contient la structure d'arborescence / parentèle (fils, petits fils, ...). Chaque nœud remonte l'information au niveau de parentèle supérieur (père). Ces communications s'effectuent à partir des messages DAO et DIO que transmettent les nœuds.
- 6TiSCH : C'est un planificateur, il gère les informations de réservation de créneau de transmission (cellule). Les nœuds sont synchronisés sur une fenêtre (slot frame) cyclique déterminée par le protocole. Cette fenêtre est elle-même scindée en intervalles plus petits (times slots). Il y a plusieurs canaux (channel) différents, un couple time slot/channel représente une cellule. Il y a des intervalles de temps partagés au cours des quels tous les nœuds écoutent et peuvent transmettre. Les autres cellules se réservent et deviennent des créneaux privés pour les transmissions entre deux nœuds. Les canaux permettent à plusieurs nœuds d'utiliser le même intervalle de temps sans risque de collision.

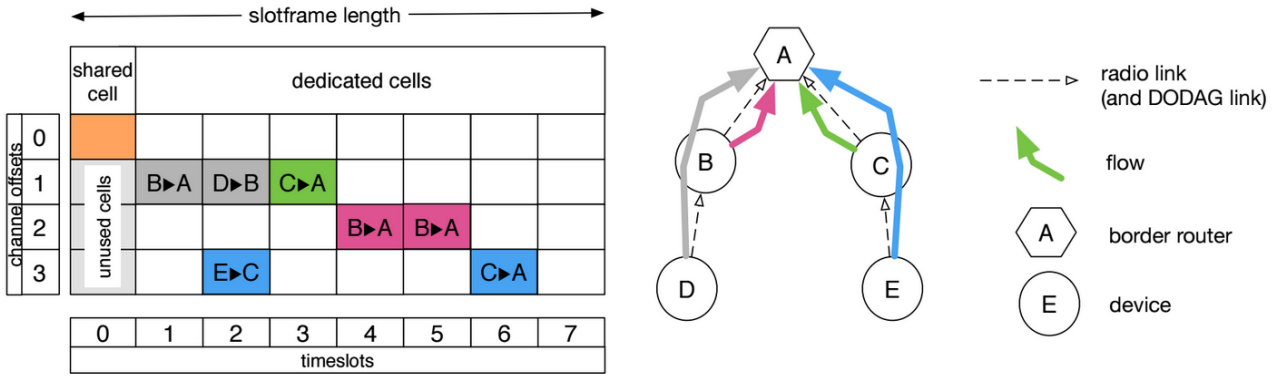

FIGURE 3.1 – Planning dans un réseau IEEE802.15.4TSCH - Illustration d'un slotframe avec 8 timeslots

Source : Self-Healing Distributed Scheduling for End-to-End Delay Optimization in Multi-hop Wireless Networks with 6TiSCH - Inès Hosni, Fabrice Theoleyre, Noureddine Hamdi

3.3 Description des données

Les nœuds sont connectés à une machine centrale qui récupère des informations sur les paquets échangés entre les nœuds ainsi que des métriques liées au réseau (état des transmissions,

distances des nœuds au puits, ...). Ces informations sont stockées dans des fichiers logs. Les logs sont transmis par un lien série, chaque ligne correspond à une information provenant d'un nœud dans une période de temps donnée (ASN). Les logs sont ordonnés par ASN croissant.

La forme générique de chaque ligne est la suivante :

ID Temporel (ASN) / ID Nœud / Événement

L'arborescence des nœuds (ID Nœud) est défini de manière dynamique par le protocole RPL. Le créneau temporel (ASN) est défini par le planificateur selon le protocole de réservation 6TiSCH. L'événement est en général l'état du nœud. Le nœud peut avoir 14 états possibles qui sont définis par 4 à 8 arguments selon l'état, dont l'adresse du nœud et l'ASN.

On trouvera en annexe la liste complète de ces états et de leurs arguments.

Chapitre 4

Problématiques et Expérimentation

La problématique principale de mon stage a été de déterminer la faisabilité de l'usage de la fouille de données pour l'amélioration des performances d'un réseau de nœuds de capteurs intelligents.

A partir de la plateforme IoTLAB, l'équipe réseau d'ICube génère une quantité importante de données : plusieurs dizaines de mégaoctets(Mo) de logs par heures de fonctionnement. L'idée est d'utiliser ces logs pour prévoir le comportement des capteurs, déterminer les facteurs responsables d'exceptions (comportement exceptionnel) ou d'erreurs, et définir des stratégies de remédiation en cas d'exceptions.

J'ai étudié différentes parties des logs en fonctions de ma progression dans leur compréhension, du partitionnement et des pistes qui m'ont été fournies. J'ai donc dans un premier temps étudié les taux de livraisons des paquets de toutes les transmissions. Puis je me suis tourné vers un champ plus restreint : les messages du protocole 6TiSCH.

4.1 Environnement expérimental

Pour générer les données de mon étude, j'utilise la plateforme IoTLAB pour démarrer et gérer des expérimentations. Pour utiliser cette plateforme, je passe par une interface web, à partir de laquelle je réserve des nœuds pour une expérimentation(cf. fig. 4.1). Je choisis ensuite la durée de l'expérimentation, et le firmware à lancer sur les nœuds. Pour l'instant pour mes expérimentations au plus 5 ou 6 nœuds se connectent au réseau (même si j'en réserve plus).

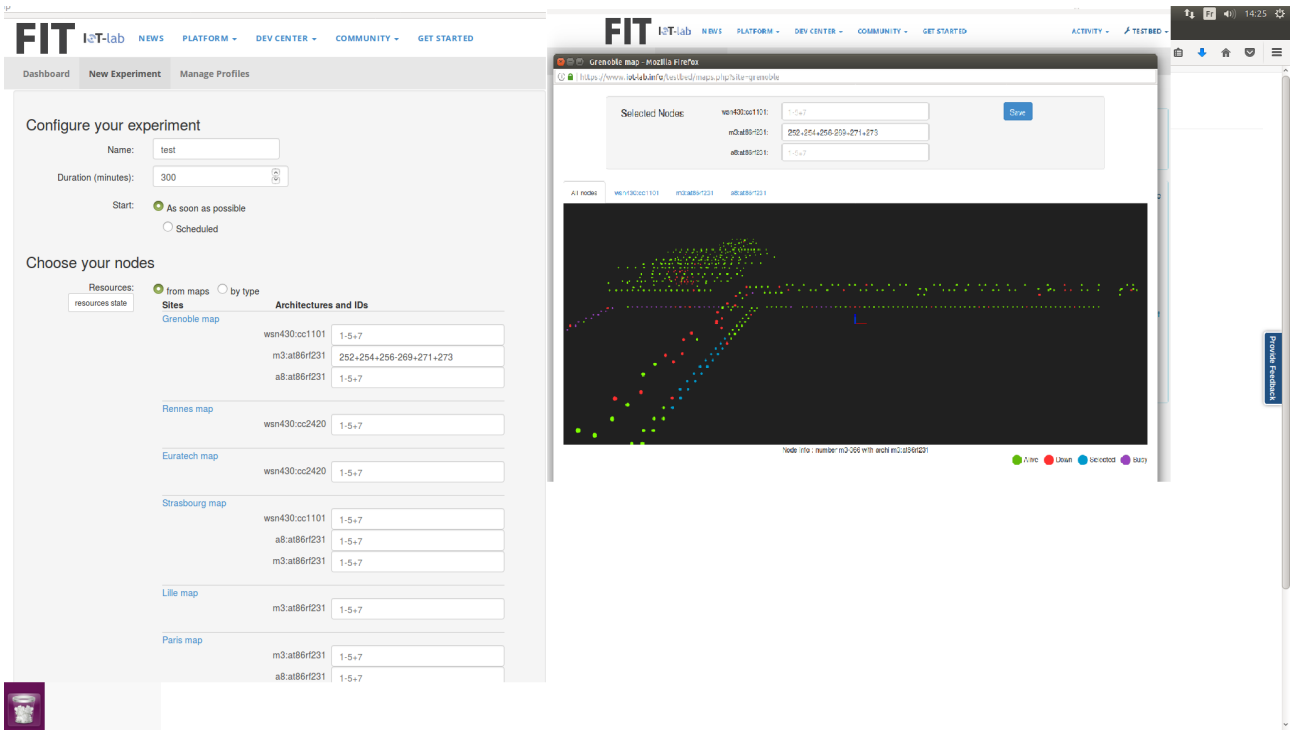

FIGURE 4.1 – Interface web de sélection de nœud

Une fois l'expérimentation lancée, j'ai accès à une interface web (cf. fig. 4.2) qui affiche en temps réel un certain nombre de données sur le déroulement de l'expérimentation, telles que les buffers, les connections, la distance au puits. Cette interface affiche aussi une représentation de l'arbre généré par le protocole RPL (i.e. une représentation du réseau, cf. fig. 4.3). Une fois l'expérimentation terminée, il reste un fichier log qui a enregistré l'activité des nœuds.

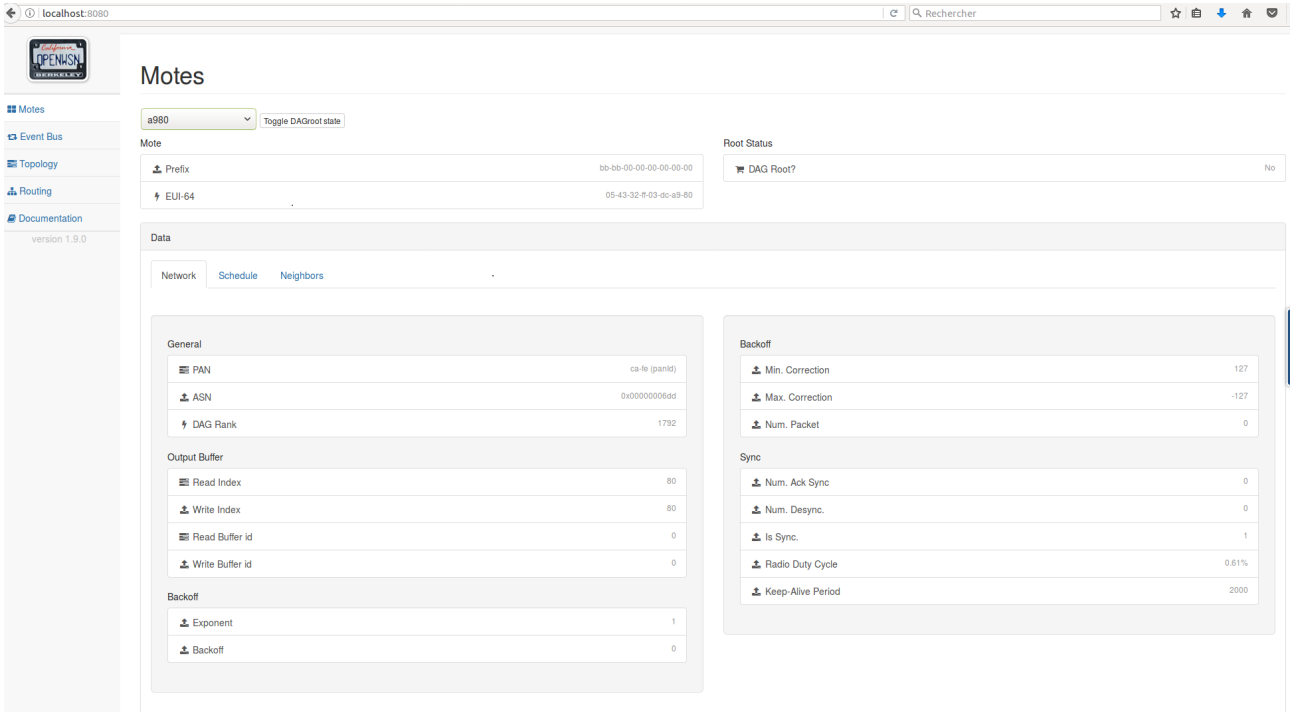

FIGURE 4.2 – Interface web de gestion des nœuds

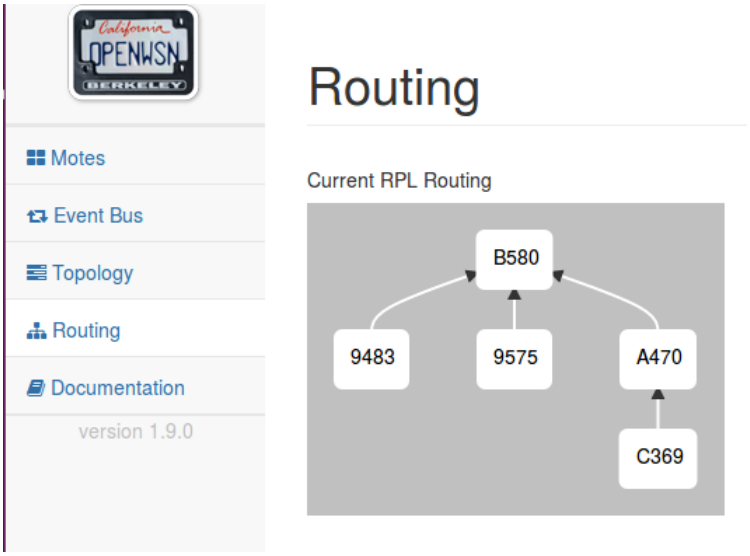

FIGURE 4.3 – Interface web de visionnage de l'arbre

4.2 Taux de livraison

4.2.1 Contexte et objectifs

L'objectif est d'analyser les variations du taux de livraison entre nœuds : prévoir les variations (notamment les baisses), puis si possible déterminer les causes de ces variations. Puis avec ces informations remonter aux causes d'échec de réservations de cellules pour l'ordonnancement du protocole 6TiSCH.

4.2.2 Expérimentation

Je sélectionne un jeu de nœuds que je connecte entre eux pour former un réseau. Ces nœuds vont générer et transmettre des données. Pour mes expérimentations j'ai utilisé le site de Grenoble avec des nœuds M3. J'ai effectué une dizaine d'expérimentations avec deux à cinq nœuds pendant 2 à 4 heures, qui génèrent de 100 à 200 Mo de données.

Des expérimentations plus spécifiques, telles qu'éteindre volontairement un nœud au cours de l'expérimentation, peuvent être envisagées afin de générer des données propres à une erreur précise.

4.2.3 Préparation des données

La première partie de mon travail a été d'analyser et mettre en forme les logs et de rendre les données plus accessibles. Pour ce faire, j'ai développé un ensemble de fonctions pour séparer et modifier les logs. J'ai commencé par développer des fonctions pour obtenir les données nécessaires pour l'étude d'un nœud :

- Des fonctions pour séparer les événements en fonction de leur types ou du nœud concerné.
- Une fonction pour déterminer les voisins d'un nœud à tout instant.

Puis j'ai développé des fonctions permettant de calculer les taux de livraison dans différents cas :

- De bout en bout : on vérifie que le paquet généré par un nœud est bien reçu par le puits du réseau.
- Analyse de tous les paquets transmis par un nœud.

Les fonctions de calcul sur les données sont des fonctions à deux variables : une fenêtre et un pas de parcours afin de conserver la composante temporelle des données. Je mets en avant des taux de transmission de certain nœuds qui se sont connectés au réseau (cf. fig. 4.4). Ces taux ne sont pas très élevé : entre 0.3 et 1, il y a donc beaucoup de perte bien que le réseau ne comporte que peu de nœud.

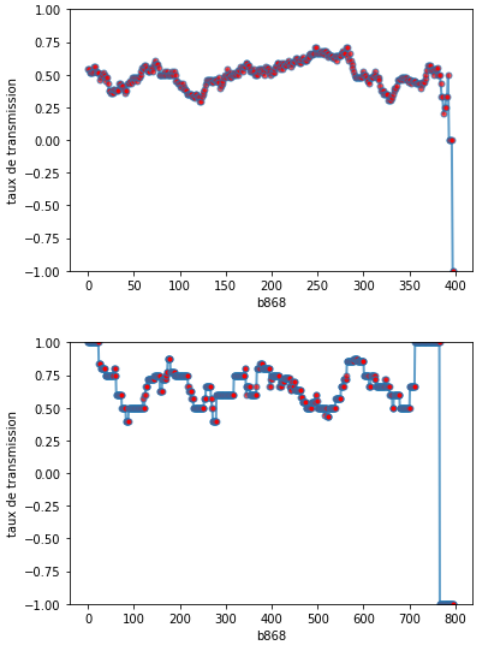

FIGURE 4.4 – Taux de transmission corrects

D'autre part seuls 5 ou 6 nœuds se connectent, les autres ont un taux de transmission nul et passent à l'état -1 que j'ai défini comme un état à partir duquel le nœud arrête d'envoyer des messages (cf. nœud c078 fig. 4.5). En étudiant un peu plus les logs on s'aperçoit que le nœud a buggé.

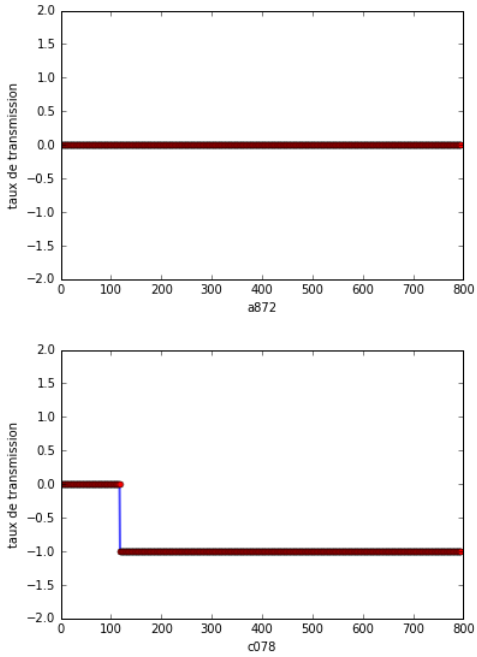

FIGURE 4.5 – Taux de transmission des nœuds buggés

J'ai obtenu de ces données :

- La liste des actions qu'effectue chacun des nœuds au cours de l'expérimentation.
- Une liste des taux de livraisons bout en bout de chacun des nœuds à chaque instant de l'expérimentation.

- Une liste des taux de livraisons des paquets d'un nœud à tous ses voisins à chaque instant.

Les données supplémentaires dont j'aurais besoin sont :

- Une liste des flux existants pour un nœud.
- Une liste des transmission par flux : un nœud peut transmettre un paquet via différents flux

4.2.4 Fouille de données

J'ai appliqué un algorithme de calcul de chaîne de Markov sur la liste des actions d'un nœud. J'ai obtenu une liste de probabilité de changement d'état pour chaque action. Ces probabilités permettent de prévoir quelle sera la prochaine action qu'effectua le nœud en fonction de celle qu'il est en train d'effectuer. Par exemple les probabilités montrent qu'après un envoi de paquet les prochaines actions les plus probables sont l'envoi de paquet ou la réception d'acquittement de transmission.

4.2.5 Conclusion

Pour approfondir la fouille j'ai besoin de données supplémentaires sur les taux de livraisons. Données que je pourrais utiliser pour mettre en avant les problèmes possibles lors des transmissions de paquets. Ces erreurs sont principalement liées au flux utilisé par les nœuds pour transmettre les paquets.

Au cours de cette partie j'ai pu mettre en avant quelques bugs sur la plateforme IoTLAB, notamment sur les microcontroleurs M3. Ces bugs m'empêchent de faire les expérimentations voulues : je ne peux lancer de réseaux de plus de 6 nœuds. Cela fait que l'arborescence de l'arbre est trop simple et qu'il n'y a pas assez d'interférences entre les nœuds. De nombreux autres bugs plus mineurs ont pu être remontés et corrigés

Au cours de cette première partie j'ai principalement travaillé sur la compréhension du problème et le traitement des données afin de les rendre utilisables pour la fouille.

Mon responsable de stage partie réseau a décidé de changer la direction de mon stage. Il lui semble plus intéressant que je travaille sur la réservation de cellule, notamment car les données sont selon lui plus accessibles. Je mets donc les taux de livraison en attente.

4.3 Réservation de cellules

Ma nouvelle tâche est d'analyser la réservation de cellule du protocole 6TiSCH (couple slot et chanel, qui détermine un créneau de temps disponible). Ces réservations sont essentielles pour le fonctionnement du réseau et leur taux de réussite n'est pas suffisant, notamment pour des réseaux comportant un grand nombre de nœuds. Je dois donc déterminer les facteurs des échecs de réservations.

Ces réservations se font en deux temps. Si le nœud A veut réserver une cellule avec B (B est le parent de A). A crée une requête qu'il envoie à B. Cette requête contient plusieurs créneaux possibles. B reçoit la requête de A, choisit un des trois créneaux et prépare une réponse qui contient le créneau choisi. B envoie cette réponse et la réservation est considérée réussie si A reçoit la réponse.

Les raisons d'échecs considérés sont :

- Un paquet est perdu (que ce soit la requête de A ou la réponse de B).
- Un nœud s'est éteint (il a planté).
- Il y a eu collision et la réservation est perdue. Les réservations s'effectuent sur des créneaux partagés et il peut y avoir des interférences entre plusieurs demandes de réservations simultanées. Ce cas est à distinguer du paquet perdu.

4.3.1 Expérimentation

Même expérimentation que pour l'analyse du taux de livraison.

4.3.2 Préparation des données

Le protocole 6TiSCH se décompose dans les logs en six messages pour une réservation entre A et son père B :

- La réservation est demandée par le capteur de A et mise en attente par le microcontroleur pour envoi vers le parent.
- La requête est envoyé de A vers B.
- La requête est reçue par B et une réponse est préparée.
- La réponse est transmise par B vers A.
- La réponse est reçue par A et la réservation est réussie si la réponse est positive.
- Il y a eu une erreur de transmission.

Bien que j'utilise une autre partie des logs, je peux réutiliser une partie des fonctions développées lors du travail sur les taux de livraisons. J'ai en plus développé des fonctions qui permettent de :

- Déterminer le parent de chaque nœud à chaque instant en utilisant les messages du protocole RPL.
- Vérifier si une réservation est réussie.
- Déterminer le nombre de réservation simultanées.
- Déterminer le nombre de tentatives si la réservation échoue plusieurs fois.

J'ai donc une liste de réservations ainsi que leur taux de réussite, les nœuds impliqués, l'ASN de la demande, le nombre de réservations simultanées et le nombre de tentatives.

4.3.3 Fouille de données

A partir des données préparées, j'ai suivi deux pistes : quelles variables jouent sur la réussite des réservations, et est-il possible d'anticiper si une réservation va réussir ? Pour ces problématiques j'ai utilisé deux algorithmes de fouille de données : les arbres de décisions, car ils permettent de faire ressortir les variables importantes d'un problème, et les modèles de Markov cachés (MMC), car ils permettent de prédire l'évolution d'un système temporel.

Arbre de décision

J'ai appliqué un arbre de décision aux données afin de déterminer les variables les plus importantes lors de l'échec ou de la réussite d'une réservation. Les variables étudiées sont le nombre de fils, de frère d'un nœud, le nombre de réservations successive échouées, le nombre de réservations simultanées.

Pour déterminer les paramètres optimaux de l'arbre, j'ai utilisé les fonctions de la bibliothèque sklearn de python. Comme paramètres d'optimisation je choisi d'utiliser la fonction de score pré installé et une validation croisée en 7 partions (7-fold cross validation). J'ai fais des test pour des validation différentes et cette configuration m'a donné les meilleurs résultats. J'obtiens une matrice de confusion, qui me permet d'analyser mon résultat (cf. fig. 4.6). J'ai un taux de vrai négatif de 0.67 et de vrai positif de 0.94.

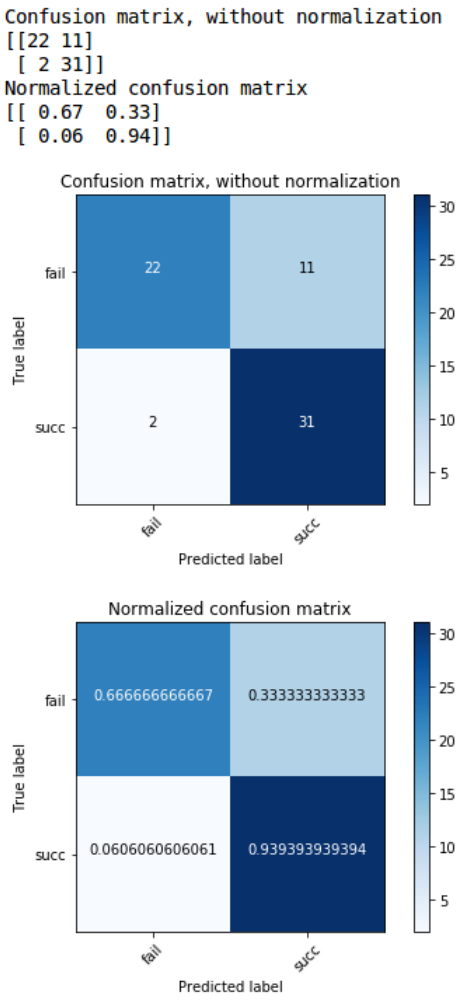

FIGURE 4.6 – Matrice de confusion

Je fais varier la profondeur de l'arbre entre 2 et 20 et le nombre d'élément minimal par feuille entre 2 et 50 sur 261 éléments. J'obtiens comme paramètres optimaux pour mon arbre une profondeur de 5 et un nombre minimal d'éléments de 3. Pour chacune des ces valeurs j'ai tracé les courbes rocs, pour un nombre d'éléments minimal de 7 (cf. fig. 4.7).

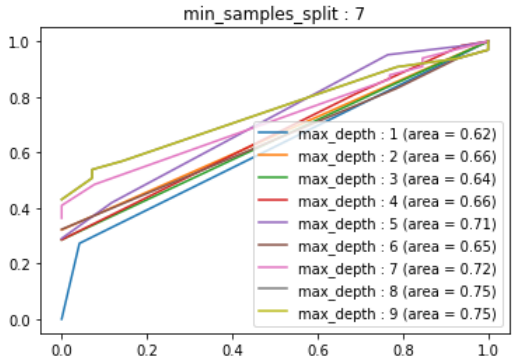

FIGURE 4.7 – Arbre de décision sur les variables des réservations

J'obtiens l'arbre suivant :

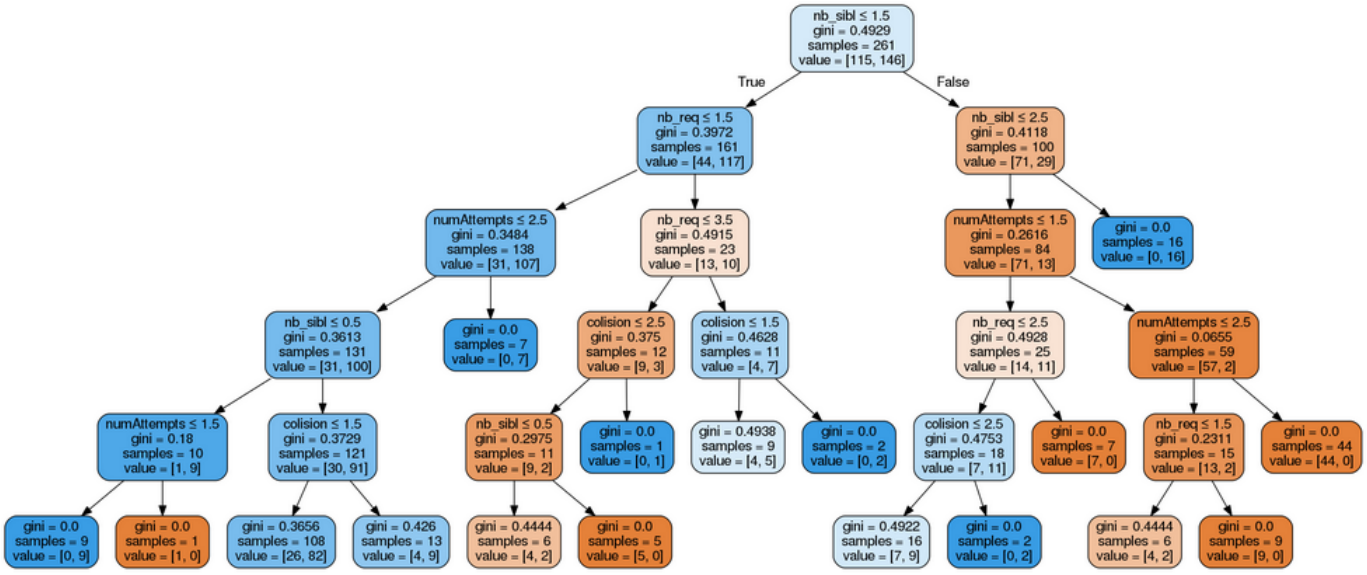

FIGURE 4.8 – Arbre de décision sur les variables des réservations

Les variables qui ressortent sont le nombre de frères, et le nombre de réservations échouées.

Les données obtenues par l'expérimentation étant trop limitées en raison du bugs sur les microcontroleurs M3, l'arbre a peu de sens car il n'a pas assez de variables à étudier. Je garde la méthode de construction et d'optimisation de l'arbre pour pouvoir la répliquer si j'obtiens plus de données.

Modèles de Markov Cachés

Pour la partie prédiction, j'ai utilisé les MMC. J'ai six type de messages du protocole 6TiSCH. Ces messages sont mes séquences d'éléments pour les algorithmes de MMC. Je sépare ces séquences en 3 jeux de données : jeu d'apprentissage pour les séquences qui mènent à une

réussite et celles qui mènent à un échec, ainsi qu'un jeu de données de test. J'ai développé un algorithme pour tester plusieurs modèles de Markov afin de trouver ceux qui prédisent le mieux les réservations réussies.

Pour ce problème j'ai travaillé avec deux classes : un modèle pour les réussites et un pour les échecs. Je génère un nombre important de modèles avec un nombre d'états différents et des matrices de transitions et d'émissions aléatoires. Chaque modèle apprend avec l'algorithme de Baum Welch et un jeu de données d'apprentissage, de réussite ou d'échec. Je classe ces modèles avec l'algorithme de Viterbi avant de les tester avec le jeu de test. J'utilise l'indicateur fmeasure pour classer mes résultats. J'introduis donc les indicateurs suivant : la précision, le rappel et la fmeasure. La précision compte la proportion d'items pertinents parmi les items sélectionnés alors que le rappel compte la proportion d'items pertinents sélectionnés parmi tous les items pertinents sélectionnables. La fmeasure est alors calculée :

$$Fmeasure = 2 * \frac{rappel * precision}{rappel + precision}$$

Les séquences sur lesquelles je travaille sont trop courtes et similaires. Les modèles ainsi formés ne peuvent pas différentier une séquence qui mène à une réussite d'une séquence qui mène à un échec.

Les résultats de ces deux approches mettent en avant deux problèmes :

- L'expérimentation étant buggée, beaucoup d'informations n'apparaissent pas.
- Les informations contenues dans les messages du protocole 6TiSCH ne sont pas suffisantes pour déterminer les causes des échecs et réussites des réservations. Il me faut en effet plus de précisions sur l'état du réseau.

4.3.4 Conclusion

L'étude des réservations de cellules me confronte au même problème que l'étude des taux de transmissions. Dû aux problèmes des microcontrolleurs la plate-forme d'expérimentation ne me permet pas d'obtenir des données suffisantes. L'hypothèse qu'un réseau limité à cinq ou six nœuds a le même comportement qu'un réseau d'un vingtaine de nœuds ne parait pas juste. En effet les erreurs dûes aux interférences des nœuds et de collision de paquets, qui sont les erreurs que je cherche à prédire sont très réduites dans un petit réseau.

Mon tuteur me fournit une simulation basée sur la plate forme IoTLAB : la partie physique de l'expérimentation est remplacée par une simulation qui transmet les données à la même interface que j'utilise sur la plate forme. Je vais donc pouvoir obtenir des données de réseaux plus grands.

D'autre part même avec cette simulation, il me faut plus de précisions sur les données pour approfondir l'étude sur les réservations. Je décide donc d'étudier l'état du réseau complet à partir des messages de réservations.

De même que dans le cas des taux de livraison, la plus grande partie de mon travail est de comprendre l'environnement et de pré-traiter les données.

4.4 État du réseau

Afin d'avoir plus de précision sur le réseau pour pouvoir approfondir mes études précédentes j'essaie d'analyse l'état du réseau. Mon objectif est de : déterminer si le réseau fonctionne "correctement" en analysant les messages du protocole 6TiSCH. Le bon fonctionnement du réseau est défini par des simulations : j'ai fait des simulations avec des taux de livraisons entres tous les nœuds forcés à des valeurs arbitraires. Plus cette valeur est proche de 1, plus l'état du réseau est "correct" (1 signifiant que tous les paquets transmis sont reçus, et 0 qu'aucun paquet n'est reçu).

Cela me permettrait de déterminer si un réseau fonctionne correctement à un instant t en connaissant les logs.

4.4.1 Simulation

J'ai fais une trentaine de simulations avec des valeurs de taux de livraisons entre 0.3 et 0.9. Je considère qu'un taux de livraison au dessus de 0.9 n'améliore pas l'état du réseau et que en dessous de 0.3 l'état du réseau est suffisamment mauvais pour que tous les nœuds ne parviennent pas à se connecter au réseau.

Il y avait différents bugs de compatibilité entre la simulation et le code qui m'a été fourni, problèmes qui ont été résolus pour la suite de l'étude.

4.4.2 Préparation des données

Le format des données obtenu est très similaire au format des données de l'expérimentation. J'ai cependant dû modifier les données afin de pouvoir utiliser les algorithmes de mise en forme développés précédemment.

Pour chaque simulation je prends comme donnée la séquence de tous les messages 6TiSCH. J'ai donc des mes trente simulations, trente séquences réparties en séquence état correct si le taux de transmission est définit supérieur à 0.5, séquence état mauvais dans le cas contraire. Dix de ces séquences choisies aléatoirement constituent le jeu de test. Les autres sont le jeu de données "correct", et "mauvais".

4.4.3 Fouille de données

Pour simplifier le problème je décide de ne considérer que deux classes : un état du réseau bon et un état mauvais. Je réutilise mon algorithme pour déterminer le meilleur couple de modèle de Markov.

Une première version du programme génère aléatoirement des couples de modèles, les forme avec l'algorithme de Baum Welch et les teste avec l'algorithme de Viterbi sur les données de tests. Je calcule après la fmeasure de chaque couple. Je peux alors sélectionner les couples avec le meilleurs fmeasure. J'obtiens la courbe fg. 4.9. Les dix meilleurs couples me fournissent une moyenne de fmeasure de 0.589. J'ai fait tourner l'algorithme pour un nombre d'états de 2 à 15, le couple avec le meilleur fmeasure a 8 états.

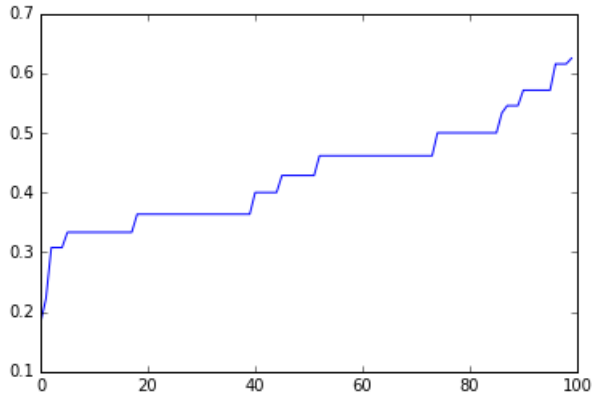

FIGURE 4.9 – Fmeasure de 100 couples de modèles de 2 à 15 état

Cette approche me donne un résultat assez faible car je teste les résultats par couple sans présélectionner de modèle pour chaque classes. Afin d'affiner mon algorithme, je teste et trie les modèles avant la phase de test sur le jeu de donnée de test. J'utilise pour cela l'indicateur de probabilité inclut dans l'algorithme de Viterbi et teste mes modèles avec les données d'apprentissage juste après celui ci. Je peux donc trier mes modèles en fonction de cet indicateur, pour l'instant je laisse les modèles séparé par le nombre d'états. Cela me permet de récupérer les meilleurs modèles pour l'état "correct" et "mauvais" séparément (cf. fig 4.10 et 4.11). J'associe en couple par ordre de "meilleur" selon le tri précédent un modèles de chaque classe. Je calcule la moyenne de mes indicateur pour chaque nombre d'état (cf. fig. 4.12).

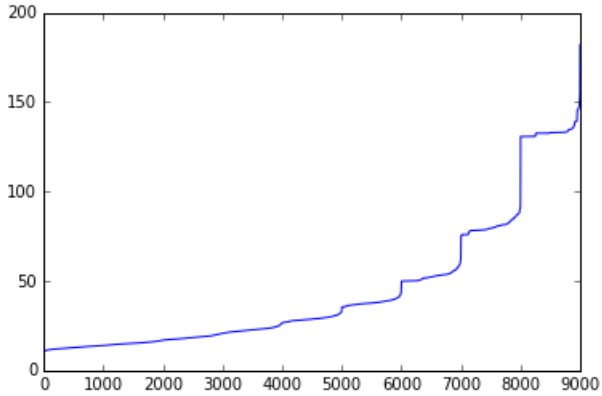

FIGURE 4.10 – Likelihood de la classe "correcte" pour 9000 modèles

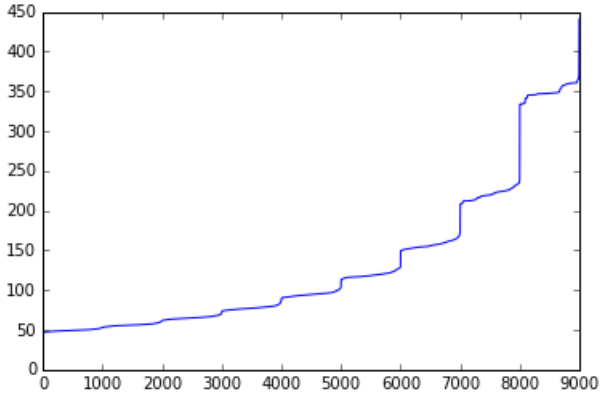

FIGURE 4.11 – Likelihood de la classe "mauvaise" pour 9000 modèles

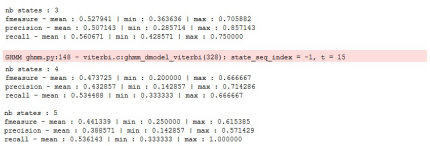

FIGURE 4.12 – Moyenne d'indicateurs en fonction du nombre d'états

Afin d'affiner encore l'algorithme, je ne sépare plus les modèles en fonction de leur nombre d'état. Je peux ainsi avoir des couples de modèle ayant un nombre d'état différent. J'associe les 10 meilleurs de chaque classe. La moyennes de leurs fmeasures est de 0.913. Le meilleur couple est constitué d'un modèle pour l'état du réseau correct de 6 états et d'un modèle pour l'état "mauvais" de 8 états.

4.4.4 Conclusion

Cette étude me permet de déterminer à partir des messages 6TiSCH l'état général du réseau.

Deux pistes possible à explorer sont :

- Prendre en compte tous les paquets transmis pour juger de l'état du réseau (pas seulement les messages du protocole 6TiSCH). Pour cela il faudrait que les logs soit modifiés car les paquets autres que ceux du protocole 6TiSCH ne sont pas différentiables tels qu'ils sont enregistrés.
- L'étude se fait sur l'ensemble du réseau. Cependant un réseau peut fonctionner correctement alors que le lien entre deux nœuds n'est pas bon et inversement. Il serait donc intéressant de considérer l'état des connexions entre chaque nœud plutôt que l'état du réseau global.

Je n'ai pas eu le temps d'approfondir ces pistes. D'autre part j'ai essayé de faire un script pour automatiser le lancement des simulations. Cela m'aurait permis d'avoir plusieurs milliers de simulations plutôt que plusieurs dizaines. Je n'y suis pas parvenu dans le temps imparti.

Chapitre 5

Conclusion

Au cours de ce stage le bilan de ce qui j'ai réalisé est :

- Une étude bibliographique et état de l'art sur la fouille de données appliquée à l'IoT. Cela m'a permis de comprendre et de situer les grandes problématiques associées aux réseaux de capteurs sans fils. J'ai pu notamment m'apercevoir qu'il n'existe aucune publication sur la fouille de données associé au protocole d'ordonnancement (6TiSCH) utilisé sur la plateforme IoTLAB.
- Une analyse des besoins et des pistes basée sur des revues régulières multi-équipes.
- L'apprentissage de l'utilisation de la plateforme IoTLAB pour réaliser des expériences et des simulations.
- Le développement en Python d'outils de mise en forme et d'analyse des logs issus des expériences et des simulations.
- L'identification de limitations des environnements d'expérimentation et de simulation du réseau de capteurs. Pour l'expérimentation : mise en évidence d'une faiblesse de design des capteurs microcontrolleurs M3 résultant d'un dysfonctionnement du réseau en cas de sélection de plusieurs nœuds ($>$ 5-6). Pour la simulation : (1) mise en évidence d'une incompatibilité entre le mode log et la simulation et (2) mise en évidence d'une erreur implémentation sur la machine d'état des nœuds simulés.
- Des tests de fouille de données sur des logs des capteurs IoT : Application d'un algorithme d'arbre de décision sur les facteurs influençant le taux de succès d'une réservation d'un créneau de communication. Apprentissage des algorithmes Modèles de Markov Caché appliqué aux réservations et à la distinction de l'état du réseau, ainsi que la mise en place d'un algorithme d'optimisation des paramètres d'un modèle Modèle de Markov Caché pour l'analyse de log basé sur un calcul de score.

Ce stage m'a permis d'approfondir mes connaissances dans les domaines de l'internet des objets, plus particulièrement dans les réseaux de capteurs et de différents protocoles utilisés (RFC, 6P, 6TiSCH). De même dans le domaine de la fouille de données, j'ai appris à utiliser sur des cas concret des algorithmes que j'avais déjà vu dans des cours de statistique. Notamment les modèles de Markov caché dont l'utilisation a été assez complexe.

D'autre part, ce stage m'a apporté une expérience en laboratoire de recherche, expérience très différente de ce que j'avais pu voir au cours de mon expérience en entreprise en première année. Cela m'a permis d'apprendre un méthodologie de recherche qui commence par savoir rédiger et comprendre l'état de l'art sur le sujet. Il faut bien cerner et comprendre les enjeu d'un problème afin de déterminer les enjeux technologique qui bloquent l'avancé du projet. Et bien que la recherche m'a formé à être autonome, j'ai aussi compris l'importance de demander l'expertise des spécialistes.

Une grande partie de mon travail a été de comprendre et d'éclaircir le sujet. J'ai d'autre part été confronté à des difficultés très technique, l'obtention des données par exemple. Cela m'a appris une méthodologie à appliquer pour face face à des problématiques ouvertes.

Il est possible de continuer cette étude en approfondissant la fouille avec plus de données, en quantité et en précision. J'ai en effet été limité par la quantité de données que j'ai utilisé et je n'ai pas eu le temps de récupérer d'autre scénarii de simulation ou d'expérimentation.

Pour la recherche dans le domaine de l'internet des objets, une étude plus approfondie sur ce sujet pourrait s'appliquer par exemple sur les protocoles. Les protocoles réseaux implémentés ont des paramètres variables déterminés par des algorithmes réseaux. Une surveillance du réseau via machine learning pourrait être une autre approche pour le calcul de ces paramètres.

Bibliographie

[1] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, and John Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 88–97. Acm, 2002.

[2] Lilia Paradis and Qi Han. A survey of fault management in wireless sensor networks. *Journal of Network and systems management*, 15(2) :171–190, 2007.

[3] Azat Rozyyev, Halabi Hasbullah, and Fazli Subhan. Indoor child tracking in wireless sensor network using fuzzy logic. *Research Journal of Information Technology*, 3(2) :81–92, 2011.

[4] Nagender Kumar Suryadevara, Subhas C Mukhopadhyay, Ruili Wang, and RK Rayudu. Forecasting the behavior of an elderly using wireless sensors data in a smart home. *Engineering Applications of Artificial Intelligence*, 26(10) :2641–2652, 2013.

[5] Cristobal Romero and Sebastian Ventura. Educational data mining : A survey from 1995 to 2005. *Expert systems with applications*, 33(1) :135–146, 2007.

[6] Yalin Başţanlar and Mustafa Özuysal. *Introduction to Machine Learning*, pages 105–128. Humana Press, Totowa, NJ, 2014.

[7] Pat Langley and Herbert A Simon. Applications of machine learning and rule induction. *Communications of the ACM*, 38(11) :54–64, 1995.

[8] Sung-Bae Cho and Hong-Hee Won. Machine learning in dna microarray analysis for cancer classification. In *Proceedings of the First Asia-Pacific bioinformatics conference on Bioinformatics 2003-Volume 19*, pages 189–198. Australian Computer Society, Inc., 2003.

[9] Pavel Berkhin. A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. Springer, 2006.

[10] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994.

[11] Hosam MF AboElFotoh, SS Iyengar, and Krishnendu Chakrabarty. Computing reliability and message delay for cooperative wireless distributed sensor networks subject to random failures. *IEEE Transactions on Reliability*, 54(1) :145–155, 2005.

[12] Dharanipragada Janakiram, VA Reddy, and AVU Phani Kumar. Outlier detection in wireless sensor networks using bayesian belief networks. In *Communication System Software and Middleware, 2006. Comsware 2006. First International Conference on*, pages 1–6. IEEE, 2006.

[13] Song Ci, Mohsen Guizani, and Hamid Sharif. Adaptive clustering in wireless sensor networks by mining sensor energy data. *Computer Communications*, 30(14) :2968–2975, 2007.

[14] Ameer Ahmed Abbasi and Mohamed Younis. A survey on clustering algorithms for wireless sensor networks. *Computer communications*, 30(14) :2826–2841, 2007.

[15] João Gama, Pedro Pereira Rodrigues, and Luís Lopes. Clustering distributed sensor data streams using local processing and reduced communication. *Intelligent Data Analysis*, 15(1) :3–28, 2011.

[16] Hui Liu, Houshang Darabi, Pat Banerjee, and Jing Liu. Survey of wireless indoor positioning techniques and systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(6) :1067–1080, 2007.

[17] L Krishnamachari, Deborah Estrin, and Stephen Wicker. The impact of data aggregation in wireless sensor networks. In *Distributed Computing Systems Workshops, 2002. Proceedings. 22nd International Conference on*, pages 575–578. IEEE, 2002.

[18] Jamal N Al-Karaki and Ahmed E Kamal. Routing techniques in wireless sensor networks : a survey. *IEEE wireless communications*, 11(6) :6–28, 2004.

[19] Guangyan Huang, Xiaowei Li, and Jing He. Dynamic minimal spanning tree routing protocol for large wireless sensor networks. In *Industrial Electronics and Applications, 2006 1ST IEEE Conference on*, pages 1–5. IEEE, 2006.

[20] Luigi Coppolino, Salvatore D'Antonio, Alessia Garofalo, and Luigi Romano. Applying data mining techniques to intrusion detection in wireless sensor networks. In *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2013 Eighth International Conference on*, pages 247–254. IEEE, 2013.

[21] Stuart Russell, Peter Norvig, and Artificial Intelligence. A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, 25 :27, 1995.

[22] Ramesh R Sarukkai. Link prediction and path analysis using markov chains. *Computer Networks*, 33(1) :377–386, 2000.

Chapitre 6

Annexes

6.1 Exemples d'architectures et protocole IoT

Nous donnerons ici quelques exemples d'architecture :

1. Vision comparative du monde d'internet et de l'internet des objets

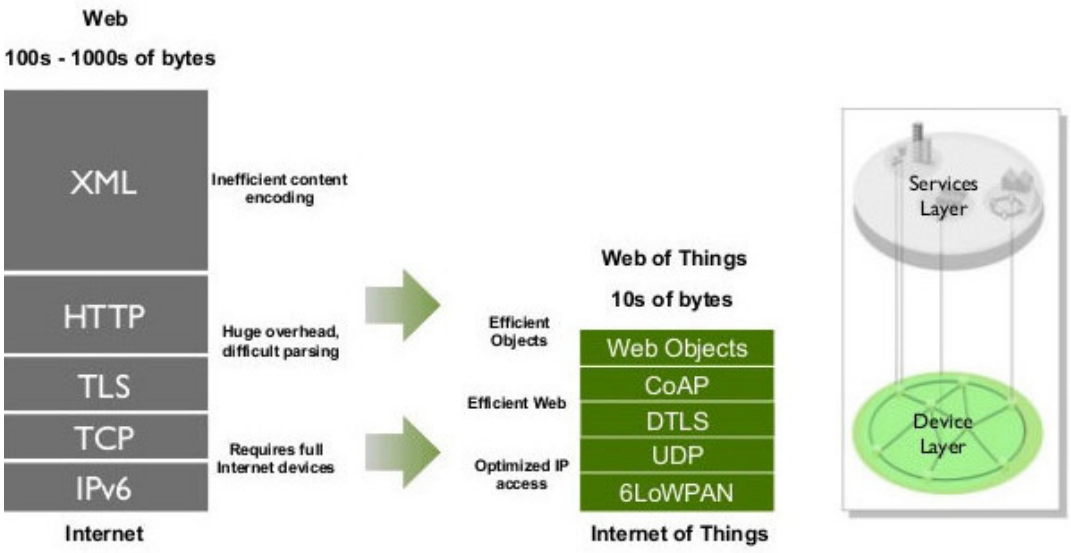

FIGURE 6.1 – Différences entre les besoin d'internet et d'IoT

Source : Sensinode : - Zach Shelby : Is the Internet Protocol enough ? (Full Presentation)

2. Deux Architectures Type IoT

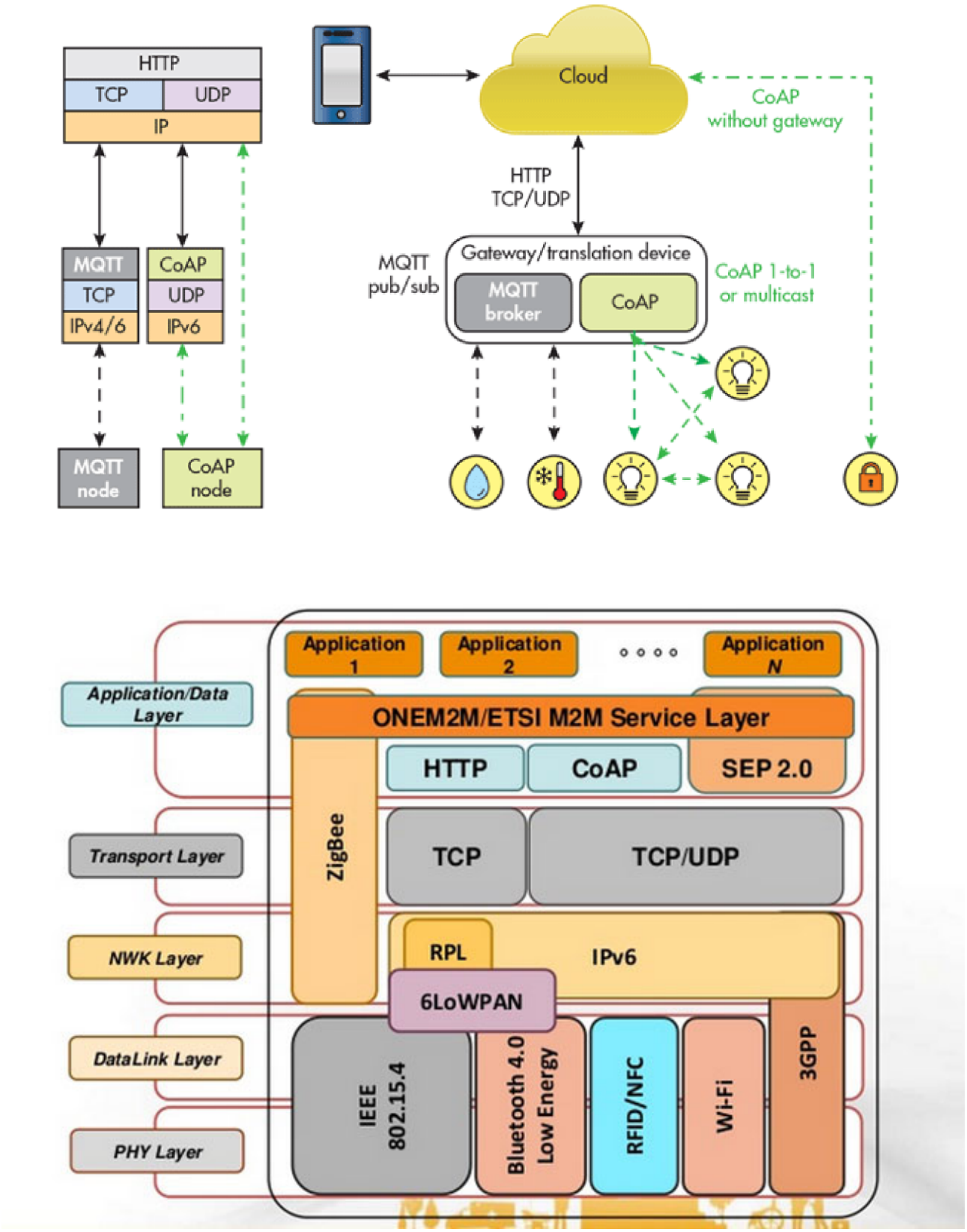

FIGURE 6.2 – architectures possible d'IoT

Source : EU Butler Project - Communication Issues

3. Représentation de standards

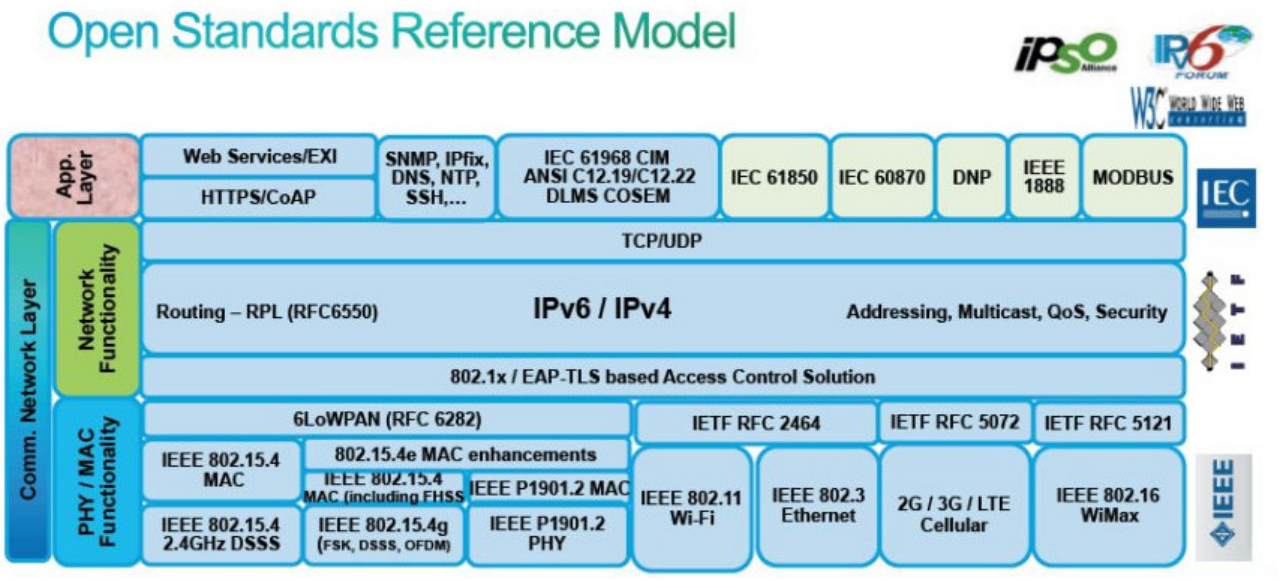

FIGURE 6.3 – Protocoles standards

Source : David E Culler Open Standards Reference Model - The Internet of Every Thing - steps toward sustainability CWSN Keynote, Sept. 26, 2011 (Presentation)

6.2 Description détaillée des données

Les événements sont répertoriés selon 14 types : (tous les événements ont comme argument l'ASN et l'adresse du nœud)

6.2.1 Description des évènements ou états du nœud

- Data generation, c'est la création d'une donnée par un nœud qui va être envoyée au puits. Ces données sont générées automatiquement à une fréquence donnée paramétrable. Les arguments sont : trackowner, seqnum, l2Src, l2Dest, queuePos.

- Data reception, c'est la réception d'un message contenant une donnée par le puits. Les arguments sont : trackowner, seqnum, l2Src, l2Dest, queuePos.

- Packet transmission, c'est l'envoi d'un paquet. Ces paquets peuvent contenir différentes informations :
  - les messages beacon servent à rejoindre le réseau et se synchroniser.
  - Les messages de réservation de cellule sont envoyés au parent du nœud pour réserver une ou plusieurs cellules de transmission.
  - les messages de données contiennent les données générées par l'étape data génération. Ils sont transmis des nœuds vers le puits et peuvent passer par des étapes intermédiaires.

Les arguments sont : trackinstance, trackowner, length, frameType, slotOffset, frequency, l2Dest, txPower, numTxAttempts, queuePos.

- Packet reception, c'est la réception d'un paquet. Les arguments sont : trackinstance, trackowner, length, frameType, slotOffset, frequency, l2Src, rssi, lqi, crc, queuePos

- Cell add, c'est l'ajout d'une cellule dans la table des créneaux de transmission. Cet ajout suit un message de demande de réservation réussie de cellule envoyé depuis un enfant vers son

parent. Les arguments sont : trackinstance, trackowner, slotOffset, type, shared, channelOffset, neighbor.

- Cell remove, c'est la suppression d'une cellule de transmission. Les arguments sont : trackistance, trackowner, slotOffset, type, sharedn channelOffset, neighbor.

- Ack reception, c'est la réception d'un paquet qui vient d'être transmis.

- Ack transmission, c'est l'acquittement d'une transmission d'un paquet.

- DIO transmission, c'est l'envoi d'un DIO, élément du protocole RPL. Le DIO contient les informations de l'instance RPL existante pour la découverte de nouveaux nœuds et pour construire les routes.

- DAO transmission, c'est l'envoi d'un DAO, élément du protocole RPL. Le DAO propage les informations du réseau vers le haut. Il a en argument son parent.

- Node state, contient le rapport cyclique et le nombre de synchronisation du nœud.

Les événements suivant sont des erreurs, ils ont les mêmes arguments que la transmission de paquet.

- Packet timeout, c'est une erreur. Le paquet a atteint son temps de timeout et est supprimé.

- Packet error.

- Packet bufferoverflow, c'est une erreur. Le paquet arrive dans un nœud dont le buffer est plein. Le paquet ne peut être conservé, il est supprimé.

6.2.2 Liste des arguments possibles et leur description

asn : variable qui s'incrémente pour représenter le temps.

adresse : nom du nœud

trackowner : nom du track pour connaître le chemin qu'a emprunté le message

seqnum : une séquence numérique qui définit le message

l2Src : source du message

l2Dest : destination du message

queuePos : position dans la queue pour l'envoi de message

frameType : type de paquet (beacon, data)

trackinstance : numéro de track

length : la taille du paquet

slotOffset : Slot alloué pour la communication du paquet

frequency : c'est le canal de fréquence sur lequel passe les transmission (16 dans cette experimentation)

txPower : puissance de l'envoi

numTxAttempts : nombre d'essai de l'envoi (1 si c'est la première fois)

rssi : indicateur de la qualité de la réception

lqi : autre indicateur de la qualité du lien

crc : indicateur d'erreur dans le paquet reçu

type : type de cellule (0=off, 1=transmission, 2=réception, 3=transmission/réception, 4=réception serie, 5=more serial rx (?), 6=occupé, 7=réservée)

shared : indique qui a le droit d'utiliser la cellule (0=seul un émetteur à droit d'émettre, 1=la cellule est partagée)

channelOffset : ?

neighbor : nœud avec qui la cellule à été créée

parent : parent du nœud qui émet le DAO