Opérer les réseaux de l'Internet des Objets à l'aide de contrats de qualité de service (Service Level Agreements)

Guillaume Gaillard

N° d'ordre NNT : 2016LYSEI152

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée au sein de l'INSA Lyon

Ecole Doctorale N° ED512
Ecole Doctorale Informatique et Mathématiques

Spécialité/ discipline de doctorat : Informatique

Soutenue publiquement le 19/12/2016, par :
Guillaume Gaillard

Opérer les réseaux de l'Internet des Objets à l'aide de contrats de qualité de service (Service Level Agreements)

Devant le jury composé de :

Turletti, Thierry	Directeur de Recherche	Inria – Sophia Antipolis	Rapporteur
Minet, Pascale	Chargée de Recherche HDR	Inria – Paris	Rapporteure
Thubert, Pascal	Ingénieur de Recherche	Cisco Systems	Examinateur
Owezarski, Philippe	Directeur de Recherche	CNRS – Toulouse	Président
Guérin-Lassous, Isabelle	Professeur des universités	Université Lyon 1	Examinatrice
Barthel, Dominique	Ingénieur de Recherche	Orange Labs – Meylan	Coencadrant
Valois, Fabrice	Professeur des universités	INSA-LYON	Directeur de thèse
Theoleyre, Fabrice	Chargé de Recherche	CNRS – ICube	Invité

Département FEDORA – INSA Lyon - Ecoles Doctorales – Quinquennal 2016-2020

SIGLE	ECOLE DOCTORALE	NOM ET COORDONNEES DU RESPONSABLE
CHIMIE	**CHIMIE DE LYON** **http://www.edchimie-lyon.fr** Sec : Renée EL MELHEM Bat Blaise Pascal 3e etage secretariat@edchimie-lyon.fr Insa : R. GOURDON	M. Stéphane DANIELE Institut de Recherches sur la Catalyse et l'Environnement de Lyon IRCELYON-UMR 5256 Équipe CDFA 2 avenue Albert Einstein 69626 Villeurbanne cedex directeur@edchimie-lyon.fr
E.E.A.	**ELECTRONIQUE, ELECTROTECHNIQUE, AUTOMATIQUE** **http://edeea.ec-lyon.fr** Sec : M.C. HAVGOUDOUKIAN Ecole-Doctorale.eea@ec-lyon.fr	M. Gérard SCORLETTI Ecole Centrale de Lyon 36 avenue Guy de Collongue 69134 ECULLY Tél : 04.72.18 60.97 Fax : 04 78 43 37 17 Gerard.scorletti@ec-lyon.fr
E2M2	**EVOLUTION, ECOSYSTEME, MICROBIOLOGIE, MODELISATION** **http://e2m2.universite-lyon.fr** Sec : Sylvie ROBERJOT Bât Atrium - UCB Lyon 1 04.72.44.83.62 Insa : H. CHARLES secretariat.e2m2@univ-lyon1.fr	M. Fabrice CORDEY CNRS UMR 5276 Lab. de géologie de Lyon Université Claude Bernard Lyon 1 Bât Géode 2 rue Raphaël Dubois 69622 VILLEURBANNE Cédex Tél : 06.07.53.89.13 cordey@univ-lyon1.fr
EDISS	**INTERDISCIPLINAIRE SCIENCES-SANTE** **http://www.ediss-lyon.fr** Sec : Sylvie ROBERJOT Bât Atrium - UCB Lyon 1 04.72.44.83.62 Insa : M. LAGARDE secretariat.ediss@univ-lyon1.fr	Mme Emmanuelle CANET-SOULAS INSERM U1060, CarMeN lab, Univ. Lyon 1 Bâtiment IMBL 11 avenue Jean Capelle INSA de Lyon 696621 Villeurbanne Tél : 04.72.68.49.09 Fax :04 72 68 49 16 Emmanuelle.canet@univ-lyon1.fr
INFOMATHS	**INFORMATIQUE ET MATHEMATIQUES** **http://infomaths.univ-lyon1.fr** Sec :Renée EL MELHEM Bat Blaise Pascal 3e etage infomaths@univ-lyon1.fr	Mme Sylvie CALABRETTO LIRIS – INSA de Lyon Bat Blaise Pascal 7 avenue Jean Capelle 69622 VILLEURBANNE Cedex Tél : 04.72. 43. 80. 46 Fax 04 72 43 16 87 Sylvie.calabretto@insa-lyon.fr
Matériaux	**MATERIAUX DE LYON** **http://ed34.universite-lyon.fr** Sec : M. LABOUNE PM : 71.70 –Fax : 87.12 Bat. Direction Ed.materiaux@insa-lyon.fr	M. Jean-Yves BUFFIERE INSA de Lyon MATEIS Bâtiment Saint Exupéry 7 avenue Jean Capelle 69621 VILLEURBANNE Cedex Tél : 04.72.43 71.70 Fax 04 72 43 85 28 jean-yves.buffiere@insa-lyon.fr
MEGA	**MECANIQUE, ENERGETIQUE, GENIE CIVIL, ACOUSTIQUE** **http://mega.universite-lyon.fr** Sec : M. LABOUNE PM : 71.70 –Fax : 87.12 Bat. Direction mega@insa-lyon.fr	M. Philippe BOISSE INSA de Lyon Laboratoire LAMCOS Bâtiment Jacquard 25 bis avenue Jean Capelle 69621 VILLEURBANNE Cedex Tél : 04.72 .43.71.70 Fax : 04 72 43 72 37 Philippe.boisse@insa-lyon.fr
ScSo	**ScSo*** **http://recherche.univ-lyon2.fr/scso/** Sec : Viviane POLSINELLI Brigitte DUBOIS Insa : J.Y. TOUSSAINT Tél : 04 78 69 72 76 viviane.polsinelli@univ-lyon2.fr	M. Christian MONTES Université Lyon 2 86 rue Pasteur 69365 LYON Cedex 07 Christian.montes@univ-lyon2.fr

*ScSo : Histoire, Géographie, Aménagement, Urbanisme, Archéologie, Science politique, Sociologie, Anthropologie

Résumé des travaux de thèse

Avec l'utilisation grandissante des technologies distribuées sans fil pour la modernisation des services, les déploiements d'infrastructures radio dédiées ne permettent plus de garantir des communications fiables, à grande échelle et pour un bas coût. Cette thèse vise à permettre à un opérateur de déployer une infrastructure de réseau radio pour plusieurs applications clientes de l'Internet des Objets (IoT). Nous étudions la mutualisation d'une architecture pour différents flux de trafic afin de rentabiliser le déploiement du réseau en partageant la capacité des nœuds et une large couverture. Nous devons alors garantir une Qualité de Service (QoS) différenciée pour les flux de chaque application.

Nous proposons de spécifier des contrats de QoS nommés *Service Level Agreements (SLA)* dans le domaine de l'IoT. Ceux-ci définissent les indicateurs clés de performance (KPI) de délai de transit et de taux de livraison pour le trafic provenant d'objets connectés distribués géographiquement. Dans un second temps, nous détaillons les fonctionnalités nécessaires à la mise en œuvre des SLA sur le réseau opéré, sous la forme d'une architecture de gestion de SLA. Nous envisageons l'admission de nouveaux flux, l'analyse des performances courantes et la configuration des relais de l'opérateur.

Sur la base d'une technologie robuste, multi-saut, *IEEE Std 802.15.4-2015 mode TSCH*, nous proposons un mécanisme d'observation de réseau permettant de vérifier les différents KPI. Nous utilisons les trames de données existantes comme support de collecte afin de réduire le surcoût en termes de ressources de communication. Nous comparons différentes stratégies de *piggybacking* afin de trouver un compromis entre la performance et l'efficacité de l'observation. Puis nous détaillons $KAUSA$, un algorithme d'allocation de ressources sous contraintes de QoS multi-flux. Nous dédions des ressources temps-fréquences ajustées saut-par-saut pour chaque message. $KAUSA$ prend en compte les interférences, la fiabilité des liens radio et la charge attendue afin d'améliorer la répartition des ressources allouées et ainsi prolonger la durée de vie du réseau. Nous montrons les gains et la validité de nos contributions par simulation, sur la base de scénarios réalistes de trafic et d'exigences.

Mots-clés : Qualité de Service, Réseaux de Capteurs Sans Fil, Multi-saut, Internet des Objets, *Service Level Agreement*, Indicateurs Clés de Performance, Fiabilité, Gestion de Réseaux, Observation de Réseaux, Ordonnancement, *6TiSCH*

Thesis summary

With the growing use of distributed wireless technologies for modern services, the deployments of dedicated radio infrastructures do not enable to ensure large-scale, low-cost and reliable communications. This PhD research work aims at enabling an operator to deploy a radio network infrastructure for several client applications, hence forming the *Internet of Things (IoT)*. We evaluate the benefits earned by sharing an architecture among different traffic flows, in order to reduce the costs of deployment, obtaining a wide coverage through efficient use of the capacity on the network nodes. We thus need to ensure a differentiated *Quality of Service (QoS)* for the flows of each application.

We propose to specify QoS contracts, namely *Service Level Agreements (SLAs)*, in the context of the *IoT*. *SLAs* include specific *Key Performance Indicators (KPIs)*, such as the transit time and the delivery ratio, concerning connected devices that are geographically distributed in the environment. The operator agrees with each client on the sources and amount of traffic for which the performance is guaranteed. Secondly, we describe the features needed to implement *SLAs* on the operated network, and we organize them into an *SLA* management architecture. We consider the admission of new flows, the analysis of current performance and the configuration of the operator's relays.

Based on a robust, multi-hop technology, *IEEE Std 802.15.4-2015 TSCH* mode, we provide two essential elements to implement the *SLAs* : a mechanism for the monitoring of the *KPIs*, and *KAUSA*, a resource allocation algorithm with multi-flow *QoS* constraints. The former uses existing data frames as a transport medium to reduce the overhead in terms of communication resources. We compare different *piggybacking* strategies to find a tradeoff between the performance and the efficiency of the monitoring. With the latter, *KAUSA*, we dedicate adjusted time-frequency resources for each message, hop by hop. *KAUSA* takes into account the interference, the reliability of radio links and the expected load to improve the distribution of allocated resources and prolong the network lifetime. We show the gains and the validity of our contributions with a simulation based on realistic traffic scenarios and requirements.

Keywords: Quality of Service, Wireless Sensor Networks, Multi-hop, Internet of Things, Service Level Agreement, Key Performance Indicators, Reliability, Network Management, Network Monitoring, Scheduling, *6TiSCH*

Remerciements

Je voudrais d'abord remercier les membres du jury pour avoir accepté de rapporter et d'examiner ma thèse et mes travaux. Merci en particulier à mes encadrants, Dominique, Fabrice et Fabrice pour leur patience, leur confiance et leur soutien tout au long du parcours doctoral.

J'adresse des remerciements chaleureux aux amis qui m'ont aidé pendant la période de rédaction, par des conseils et des relectures attentionnées. En particulier, merci à Béa, Jule, Fred et Arnaud.

Je voudrais remercier Orange Labs, Inria et le laboraoire CITI pour avoir mis à ma disposition les moyens nécessaires à l'accomplissement de mes travaux. En particulier, la participation aux réunions de l'IETF et l'accès facilité à la plate-forme FIT/IoT-Lab m'ont grandement servi. Merci à l'Icube pour le soutien financier en fin de convention CIFRE. Rien n'aurait pu aboutir sans l'aide inconditionnelle pour mes démarches de la part des assistants et responsables des équipes, des directions, et des différents secrétariats.

Pendant quatre ans, les personnes rencontrées et cotoyées au laboratoire CITI, à Orange Labs et en mission m'ont enrichi par des échanges d'expériences, des collaborations scientifiques et des partages culturels. Les étudiants thésards, post-doc, stagiaires, enseignants, chercheurs, techniciens et ingénieurs que j'ai connus ont fortement contribué à mes réflexions et travaux. En particulier, les échanges au sein de l'équipe UrbaNet m'ont beaucoup inspirés et guidés. J'ai également eu l'opportunité de collaborer aux travaux de deux stages, je remercie en particulier Jonathan pour nos discussions et avancées communes.

Enfin, je salue mes proches, colocs, famille, amis et amour, qui m'ont accompagnés à chaque instant de la thèse. Ce que leur présence m'apporte au quotidien est inestimable.

Table des matières

Table des figures

Chapitre 1

Introduction

La collecte par transmissions radio d'informations issues de capteurs, distribués dans l'environnement pour des usages variés, est un des défis de ce début du vingt et unième siècle dans le secteur des télécommunications. Dans ce contexte, nos travaux visent à permettre à un acteur tel qu'*Orange*[1] d'exercer son rôle d'opérateur dans le domaine de l'Internet des Objets (IoT), en particulier dans les villes, où se généralisent les usages des technologies distribuées sans fil.

L'IoT regroupe les applications reliant de nombreux dispositifs à l'internet afin de rendre l'environnement plus « intelligent ». Les usages varient de l'échelle humaine, de l'habitat (le *Smart Home*, avec par exemple le contrôle automatique ou à distance de la température) à celle de la ville (les *Smart Cities* avec par exemple la gestion de l'éclairage public), des transports (e.g. supervision du trafic routier) et de l'industrie (les réseaux radio industriels) [**?**].

La connexion à l'internet se fait en particulier grâce à des communications radio (elles-mêmes utilisant ou non le protocole IP). Ce médium est sujet à des variations et à des pertes d'information. L'utilisation de bandes de fréquences sans licences pour l'IoT accentue le problème de la fiabilité et de la Qualité de Service (QoS) des communications radio. Les opérateurs doivent proposer des solutions durables, à l'échelle des trafics, et raisonnables en coût de déploiement.

Dans ce document, nous étudions la gestion de ressources dédiées sur une architecture multi-saut d'opérateur. Nous définissons ce terme et justifions le bien-fondé de s'y intéresser dans la suite de ce chapitre.

Dans cette introduction, nous décrivons en premier lieu le contexte et la nécessité de la gestion opérée des télécommunications sans fil, puis nous détaillons les défis à relever pour mettre en œuvre l'approche opérée sur les réseaux multi-sauts. Nous expliquons enfin sur quels aspects nous répondons à ces défis.

1.1 Depuis les réseaux de capteurs dédiés vers les réseaux multi-sauts opérés

Dans cette section, nous décrivons les applications les plus importantes de l'IoT, en particulier dans le contexte des *Smart Cities*. Puis nous détaillons les deux tendances opposées de la réalisation de services par l'IoT sans fil :

1. Ce travail de thèse est issu d'une collaboration entre *Orange Labs*, l'*Insa de Lyon*, l'équipe *Réseaux* de l'*ICube*, et l'équipe *Inria UrbaNet* au sein du laboratoire *Citi*.

1. une approche « historique » prône un déploiement indépendant par application (e.g. différents systèmes domotiques) ;
2. une approche « mutualisée » dans laquelle un opérateur offre des services différenciés de communication à plusieurs clients.

Enfin, nous discutons du choix d'une topologie de réseau mono-saut ou multi-saut.

1.1.1 La diversification des applications

Depuis les années 2000, les objets connectés par des capteurs sans fil apparaissent de plus en plus fréquemment dans la vie des utilisateurs de services, notamment urbains. En effet, les applications des technologies radio distribuées sur des dispositifs contraints se popularisent [**?**] et améliorent les services :

1. individuels : par exemple pour le confort (domotique, gestion du chauffage, prévisions météorologiques), la sécurité (intrusion, alarme incendie, fuite de gaz), ou l'amélioration des services (la télérelève des compteurs d'eau ou d'électricité facilite la mesure de consommation et les économies) ;
2. collectifs : par exemple les services urbains (mesure du trafic routier, surveillance de l'état d'un bâtiment [**?**], gestion optimisée de la collecte de déchets), la collecte de mesures dans des environnements variés (sur route pour la pollution, en mer pour l'étude des courants marins, dans les champs pour l'agriculture, ou en forêt [**?**] pour les risques d'incendies).

Dans un contexte d'urbanisation, c'est-à-dire d'accroissement de la population urbaine, l'*IoT* vise à collecter massivement les informations générées depuis les divers capteurs et objets connectés afin d'améliorer les services pour les citoyens : l'*IoT* contribue à former les *Smart Cities* [**?**]. On y réduit les dépenses d'énergie, tout en améliorant les conditions de vie des hebitants [**?**]. Par exemple, la collecte d'informations sur le trafic automobile, sur les transports en commun, le stationnement, ou sur la mobilité des personnes permet de réduire les temps de déplacements, les dangers et les pollutions. En effet, ces données permettent d'adapter la signalisation urbaine et d'améliorer l'offre de transports. Ainsi, la collectivité peut mesurer les niveaux de bruit, de polluants, de lumière et ajuster les services d'éclairage et de nettoyage. Connaître le niveau de remplissage de bennes rend la collecte d'ordures ménagères plus efficace. Enfin, la télérelève des compteurs d'eau, de gaz et d'électricité permet aux distributeurs d'ajuster leurs infrastructures et de mesurer précisément les évolutions de consommation.

Pour chaque usage qui se modernise, un client peut demander des informations collectées depuis ses dispositifs (e.g. un client souhaite collecter les informations de disponibilité des places de stationnement d'une ville donnée pour un système de *parking intelligent*). Afin de mesurer tous les événements (e.g. une place se libère), les trafics applicatifs clients se densifient. Par exemple, l'information de télérelève de la consommation journalière s'enrichit pour offrir la possibilité de mesurer celle de la dernière heure, sur demande, au moment de la clôture d'une facture. De plus, les mesures sont plus précises : les informations fournies par les capteurs sont plus riches, du fait de la qualité de la mesure, et plus personnalisées (le nombre d'usagers augmentant, l'identification des messages exige de plus longs en-têtes).

L'accroissement de la densité des sources de trafic et leur diversification posent le problème de la collecte des informations de l'*IoT*. En particulier, les opérateurs doivent choisir l'architecture de réseau et les technologies adéquates pour répondre à ces nouveaux besoins.

1.1.2 Les réseaux dédiés à une application

Jusqu'en 2015, les déploiements connus de réseaux de capteurs sans fil sont dédiés (e.g. *SensorScope* utilise un réseau dédié pour la surveillance météorologique [**?**]) :

1. ils sont spécifiques à une application (e.g. la télérelève de compteurs d'eau), c'est-à-dire que les choix technologiques et le dimensionnement du réseau sont étudiés pour répondre au mieux aux besoins d'une application donnée ou d'un ensemble homogène d'applications ;
2. ils sont ajustés au type de données à collecter (taille et fréquence de génération des messages) et aux besoins applicatifs (e.g. remontée d'informations agrégées, issues de plusieurs capteurs) ;
3. ils ne concernent qu'un seul acteur (e.g. un distributeur de gaz) pour une application donnée ;
4. les nœuds déployés forment une topologie dédiée. À chaque nouveau service correspond un nouveau déploiement, avec ses propres antennes, relais, contrats, sa couche protocolaire, et un usage particulier des ressources radio, qui est défini indépendamment et qui peut entrer en compétition avec les autres.

Souvent, les mesures issues de capteurs de différentes natures doivent être corrélées pour réaliser des tâches plus complexes [**?**] (e.g. orienter un usager se déplaçant en milieu urbain vers un stationnement libre, détecté par le système, en évitant les embouteillages). L'approche en réseaux dédiés rend les interconnexions difficiles entre les différentes infrastructures. En effet, l'hétérogénéité des technologies et des architectures de réseau rend les passerelles d'interconnexion onéreuses. L'analyse des données collectées est complexe du fait de l'indépendance des formats de données et des modes de collecte de l'information. En particulier, il est difficile de définir des critères communs de *QoS* de bout-en-bout sur différents réseaux dédiés puisque la nature des messages et des supports utilisés varie.

Les dépenses liées au déploiement et le faible retour sur investissement ont freiné les solutions dédiées. En effet, la mise en place d'une infrastructure dont la couverture est suffisante pour relier l'ensemble des capteurs n'est pas rentabilisée pour une seule application, car elle implique de faibles trafics et peu de réutilisation des ressources déployées dans le temps et dans l'espace.

Enfin, de multiples déploiements concurrents favorisent un usage non concerté des ressources de communication. Pour le médium radio, et en particulier sur des bandes de fréquences sans licences, l'approche dédiée accroît le risque d'interférences et de compétition à l'accès. La réglementation édictée par l'agence nationale des fréquences (ANFR) en termes d'utilisation du médium partagé par chacun des nœuds et réseaux ne suffit pas à empêcher une occupation non contrôlée des ressources. En effet, l'utilisation d'un canal particulier par plusieurs technologies peut entraîner des conflits et des pertes de données.

1.1.3 Un opérateur de réseaux de l'*IoT*

Afin de pallier les limites structurelles des réseaux dédiés, nous proposons dans ce travail d'analyse d'étudier le modèle d'infrastructures radio mutualisées pour différentes applications. Dans ce cas, un acteur nommé ***opérateur*** prend en charge le déploiement et l'exploitation d'une technologie radio facilitant la collecte des données de différents ***clients***. Une entité centrale (un serveur de l'opérateur) est responsable du partage des ressources. L'opérateur peut alors organiser les communications sous la forme de flux applicatifs. Il construit un réseau multi-client, multi-application, multi-flux de manière à rentabiliser le déploiement de son infrastructure en termes de trafic et de couverture.

La mutualisation de services par un opérateur s'inscrit dans le cadre de l'*IoT*. En effet, la généralisation de l'usage des protocoles *IP* pour le transport des données depuis ou jusqu'à l'utilisateur final permet de donner facilement accès aux informations issues des capteurs. L'opérateur assure donc la collecte de données de manière transparente et facilite leur analyse en utilisant *IP*.

L'*IoT* permet de surcroît d'exploiter les mécanismes de gestion des réseaux *IP* adaptés aux contraintes des objets communicants. En particulier, l'opérateur peut tirer profit des efforts de standardisation reliant les standards *IP* et les protocoles pour réseaux de capteurs robustes et efficaces comme *IEEE Std 802.15.4-2015*. Dans le cadre défini par *IPv6 over the Timeslotted Channel Hopping mode of IEEE Std 802.15.4-2015* (*6TiSCH*) [**?**], l'opérateur organise les communications partageant les ressources entre les flux applicatifs.

Respecter les besoins spécifiques à chaque application Chaque application a ses propres caractéristiques d'usages et de trafic, et requiert donc un certain niveau de Qualité de Service (*QoS*) [**?**]. En particulier, chaque application présente des besoins variés en termes de taux de livraison des messages et de délai de collecte des informations. Ainsi, une alarme requiert un faible délai de transit, tandis que les messages de télérelève doivent être collectés pour chaque compteur. Ces besoins s'expriment sous la forme d'indicateurs clé de performance ou *Key Performance Indicators* (*KPI*), variables selon les applications.

Jusqu'en 2015, les réseaux pour l'*IoT* proposés par les opérateurs sur bandes sans licences partagent indifféremment les ressources entre les différents flux afin de rentabiliser le déploiement et de réduire les coûts d'utilisation. Lorsque les clients exigent de l'opérateur des garanties de performances, l'approche n'est plus adaptée. Pour limiter l'impact des autres trafics et garantir des niveaux de *QoS* élevés, l'opérateur doit envisager de dédier des ressources à chaque flux.

Afin de formaliser les engagements de l'opérateur vis-à-vis de l'ensemble de ses clients, nous envisageons dans cette étude la négociation de *Service Level Agreement* (*SLA*). Ceux-ci permettent de spécifier sans ambiguïté les trafics attendus, les *KPI* exigés et les actions et pénalités consécutives au non-respect des conditions contractuelles. L'usage des *SLA* pour les réseaux *IP* a déjà prouvé son utilité pour quantifier les ressources nécessaires à chaque application et dimensionner l'infrastructure des opérateurs [**?**].

À la différence des réseaux *IP* filaires, les liens radio dans le domaine de l'*IoT* ne sont pas fiables. Du fait de la multiplication des déploiements, l'environnement radio subit des perturbations entraînant de plus en plus de contraintes. Ainsi, la tâche d'un opérateur de réseaux radio devient de plus en plus complexe.

1.1.4 Une architecture de réseaux radio multi-sauts

Deux architectures sont actuellement en compétition pour les réseaux de l'*IoT* :

1. l'architecture multi-saut, issue des travaux académiques sur les réseaux de capteurs sans fil, dominante au début de nos travaux : plusieurs nœuds relais retransmettent les messages issus des sources de trafic vers les points de sortie du réseau ;
2. l'architecture mono-saut, mise en œuvre pour l'*IoT* par les solutions *LoRa* [**?**] et celles de l'opérateur *SIGFOX* [**?**] : les topologies sont formées de liens directs de longue portée entre les sources et les points de sortie du réseau.

C'est le coût de l'infrastructure (relais) et de déploiement qui favorise la seconde approche dès le début des années 2009/2010, y compris chez *Orange* à partir de 2013/2014. En effet, les liens de longue portée requièrent un nombre moins important de nœuds. Les bas débits utilisés réduisent les performances nécessaires et la consommation énergétique des nœuds.

À l'inverse d'une topologie mono-saut à bas coût, nous privilégions une solution multi-saut. La mise en place de relais accompagnée de stratégies de routage permet une organisation des communications plus fiable et plus efficace en ressources. En effet, le multi-saut permet de choisir les liens à utiliser pour former les routes selon les interférences locales, afin de garantir un taux de livraison de bout-en-bout. Les retransmissions saut-par-saut ont moins d'impact sur le délai et l'occupation des ressources car elles n'interfèrent qu'au voisinage d'un lien.

En supposant une connaissance précise de la topologie et des interférences, un opérateur peut déterminer la quantité de ressources nécessaires pour écouler un trafic donné et donc la capacité de son réseau.

1.2 Les défis de l'opérateur multi-service pour l'Internet des Objets

Assurer le rôle d'opérateur nécessite de répondre au problème de fiabilité, qui est étudié pour les réseaux de capteurs sans fil. La mutualisation des services sur une seule infrastructure crée de nouveaux problèmes liés à la différenciation des applications et au maintien de garanties de QoS. Nous revenons d'abord sur la fiabilité, puis nous décrivons les ressources de communication et leur gestion.

1.2.1 La fiabilité pour les réseaux de l'Internet des Objets

Le principal défi à relever dans les réseaux de capteurs sans fil est la fiabilité des communications dans un environnement radio sujet à des perturbations. L'enfouissement des capteurs, la diversité des conditions de propagation des ondes électromagnétiques et la variabilité des objets présents (e.g. passages de véhicules sur route, reflexions sur les immeubles) rendent les succès de transmission difficiles à prévoir.

La qualité des liens radio entre deux nœuds n'est généralement ni homogène ni symétrique. En effet, les capteurs sont souvent concentrés du fait de la proximité des phénomènes physiques à mesurer, (e.g. l'occupation de place de parking), et cloisonnés (e.g. entourés de murs ou sous une plaque métallique). En particulier, cet environnement impacte le trafic descendant lié à des commandes d'actionneurs ou à des messages de configuration.

L'opérateur doit prévoir suffisamment de ressources pour assurer une fiabilité minimum le long de son infrastructure. Les communications doivent être organisées de manière à réduire les conflits et l'impact des interférences. Nous répondons à ce problème dans la suite du document. L'opérateur devra aussi savoir dimensionner le nombre et le placement de ses nœuds relais de manière à assurer une connectivité suffisante. Enfin, il devra adopter une gestion dynamique des ressources de communication pour s'adapter aux variations des conditions radio, de l'état des nœuds et du réseau.

1.2.2 Les ressources de communication dans les réseaux de l'IoT

Dédier une ressource correspond à réserver à certains nœuds l'utilisation d'un canal de fréquences dans un voisinage donné à un instant donné. Avec des ressources dédiées, l'opérateur peut réduire les conflits à l'accès au médium s'il sait mesurer les interférences et maîtriser la topologie dans le voisinage des nœuds.

Si le trafic et les conditions de propagation sont stables et connus, l'opérateur est en mesure de connaître la quantité de ressources utilisées et la *capacité* disponible. Il peut alors planifier l'usage des ressources de son réseau en choisissant les nœuds et liens

adéquats. En supposant que le fonctionnement du module radio des nœuds constitue la principale dépense d'énergie, et en reliant celle-ci au trafic, l'opérateur peut aussi en déduire leur consommation d'énergie. Ainsi, il peut adopter une stratégie durable d'utilisation des ressources.

Afin de répondre aux exigences des différentes applications, l'opérateur doit choisir quelles ressources allouer à chacune. En effet, ce choix influe sur le délai et le taux de livraison des différents flux. Il doit trouver un compromis entre performances, garanties et coûts pour l'ensemble des nœuds et applications.

Avec l'augmentation de l'usage de l'*IoT* et la multiplication des opérateurs, la capacité maximale du médium radio sera vite atteinte (accroissement du trafic et du nombre de nœuds) [**?**]. Une optimisation de l'usage des ressources est donc nécessaire, en particulier sur des bandes non licenciées. Nous étudions comment réduire l'usage des ressources de l'ensemble des clients, en supposant les trafics connus à l'avance. Nous n'étudions pas en profondeur l'usage de ressources partagées pour des trafics dynamiques. Le multiplexage des ressources entre flux d'une même application, ou entre applications, pourrait réduire le coût en termes de capacité (en réduisant la quantité de ressources allouées), mais ne garantirait plus leur fiabilité.

Dans cette étude, nous adoptons la notion de ressources temps-fréquences étudiée par le groupe *6TiSCH*. Elle facilite la gestion des ressources sous les hypothèses de connaissance de la topologie, de stabilité des conditions, de trafics périodiques et de capacité de contrôle de la configuration du réseau.

1.2.3 La gestion des réseaux de l'Internet des Objets

Le problème de la gestion des réseaux de capteurs sans fil est bien connu [**?**]. Le gestionnaire doit maintenir une robustesse aux pannes et aux attaques sur son réseau : il vérifie régulièrement l'état du réseau, de manière à diagnostiquer les problèmes et leurs causes (***observation***). Il doit être en mesure de modifier certains paramètres de sa configuration (e.g. l'allocation des ressources) afin de s'adapter aux changements sur le réseau ou de rétablir une exécution normale en cas de détection d'un dysfonctionnement.

De plus, la gestion du réseau doit être efficace en termes de ressources de communication, d'énergie et de coût. Le gestionnaire est donc confronté au problème de la mesure de l'information : comment connaître à faible coût l'état du réseau ? De même que pour les données applicatives, l'information d'observation souffre d'une qualité de service variable (pertes lors de la collecte).

Pour l'*IoT* opéré, le rôle d'opérateur génère deux nouvelles contraintes :

1. la gestion du réseau doit être au moins en partie centralisée : les exigences applicatives s'expriment pour l'ensemble du réseau puisqu'elles concernent des capteurs repartis géographiquement. La négociation des *SLA* porte sur des critères globaux vérifiables uniquement de manière centrale ;
2. le multi-service impose de différencier la gestion de réseau pour chaque application (e.g. la période d'observation varie selon l'exigence des clients, mais pour une application sensible aux pertes de trames, l'opérateur doit adapter plus souvent la configuration aux changements de l'environnement radio).

Dans ce document, nous étudions des mécanismes centralisés d'observation et d'allocation de ressources dans le cadre de réseaux multi-services et multi-sauts. Les nœuds considérés sont statiques. Nous abordons le problème de la gestion de flux de trafic pour un fonctionnement périodique (avec priorités dans le choix des ressources et vérifications

de contraintes par flux). Notre étude ne porte pas sur les stratégies commerciales à adopter par l'opérateur au moment de la négociation ou de la mise en œuvre de *SLA* sur le réseau. Le domaine de la sécurité du réseau (résistance aux attaques, intrusions, corruptions des données) n'est pas non plus traité mais fait l'objet d'études plus générales pour l'*IoT*. Enfin, les actions humaines sur les nœuds, liées à la maintenance ou à de nouveaux déploiements, ne sont pas étudiées ici.

1.3 Contributions

Nous avons focalisé nos recherches sur deux défis majeurs : l'expression des besoins applicatifs, et la supervision du réseau de l'opérateur. En effet, ce sont deux aspects primordiaux pour assumer le rôle d'opérateur : ils sont nécessaires pour différencier les applications et leur traitement.

1.3.1 Spécifier les *Service Level Agreements (SLA)* pour l'*IoT*

Nous proposons d'abord d'étendre le formalisme des *SLA* pour définir les besoins des applications de l'*IoT* et spécifier les exigences à respecter par l'opérateur. Dans le Chapitre 3, nous étudions d'abord les *KPI* essentiels pour l'*IoT*. Nous décrivons comment les construire à partir de métriques brutes, afin de prendre en compte les spécificités de la collecte de données pour chaque application. Nous répondons au problème de la spécification de *KPI* pour une multiplicité de nœuds et pour des réseaux multi-sauts.

Puis nous précisons la définition des trafics de l'*IoT* : nous proposons de définir des restrictions sur les trafics contractualisés, sous la forme de flux périodiques générés depuis les nœuds sources. Ces limites permettent à l'opérateur d'estimer les ressources utilisées dans le cas du respect des *SLA*, en lien avec la capacité du réseau.

Enfin, nous spécifions dans les *SLA* les engagements de l'opérateur concernant la collecte d'informations d'état du réseau, afin de prouver la validation des *KPI*. Nous définissons les garanties et les actions à réaliser en cas de dégradation du service.

1.3.2 Une architecture de gestion de réseaux radio multi-sauts opérés

Une fois l'expression des besoins des clients et des engagements de l'opérateur spécifiés dans les *SLA*, ceux-ci doivent être mis en œuvre sur l'infrastructure radio. Au Chapitre 4, nous proposons une architecture de gestion de services dédiée à l'*IoT* opéré. Elle comprend un ensemble de fonctionnalités permettant à l'opérateur de collecter les trafics des clients en respectant les *SLA*.

Nous décrivons comment articuler la négociation et la spécification des *SLA* à la capacité du réseau de l'opérateur, et à la distribution des sources de trafic. Nous détaillons la nécessaire transposition des exigences applicatives en ressources réseau. Nous distinguons les aspects dépendants de la technologie utilisée dans la mise en place des ressources et justifions les mécanismes d'allocation et de réallocation selon les *SLA* mis en œuvre.

Nous montrons la nécessité de proposer des mécanismes de contrôle du réseau pour le maintien de la *QoS* des applications clientes. Nous organisons la construction d'informations d'observation de l'état du réseau, leur collecte. Nous décrivons leur utilisation pour maintenir les performances en changeant la configuration du réseau.

Enfin, nous expliquons comment l'analyse des performances du réseau doit permettre l'anticipation des défaillances et les choix d'admission de nouveaux flux.

1.3.3 Vérifier les *SLA* et l'état des ressources par une observation efficace

L'opérateur doit démontrer à chaque client le respect des exigences de *QoS*. La remontée d'informations de contrôle ne doit pas perturber le fonctionnement du réseau ni réduire sa durée de vie au delà du trafic de données.

Nous proposons au Chapitre 5 un mécanisme de collecte d'informations d'observation efficace en termes d'utilisation de ressources. Nous évaluons par simulation différentes stratégies de *piggybacking*, afin de trouver un compromis entre le coût en ressources du mécanisme et l'emploi de ressources pour une collecte dédiée et séparée.

Afin d'observer la réalisation de chaque *SLA* et de vérifier l'état du réseau, nous décrivons un ensemble de métriques ayant un sens dans le domaine de l'*IoT* opéré. Nous détaillons la manière de les mesurer sur les nœuds pour ensuite les collecter. Par exemple, nous montrons comment des mesures de *Packet Delivery Ratio* (*PDR*) peuvent être réalisées de bout-en-bout sur le réseau de l'opérateur, ou estimées à partir des valeurs locales de *Packet Error Rate* (*PER*).

1.3.4 Ordonnancer les ressources par contraintes de flux pour l'*IoT* opéré

La configuration du réseau sous contraintes multiples est une tâche essentielle pour exercer le rôle d'opérateur en distinguant différentes applications. Le Chapitre 6 détaille notre contribution pour quantifier les ressources nécessaires à chaque flux applicatif et les allouer en prenant en compte les contraintes du réseau, les exigences applicatives et la durabilité de l'infrastructure déployée.

Nous montrons d'abord comment choisir les relais formant les chemins radio pour chaque flux applicatif, de manière à équilibrer globalement la charge des nœuds. Nous prévoyons des retransmissions de trames saut-par-saut sur chaque chemin radio. Celles-ci permettent d'améliorer le *PDR* des flux, au détriment du délai de transit et du coût en ressources.

Nous montrons ensuite comment respecter les *KPI* de chaque application en proposant *KPI-Aware Scheduling Algorithm* (*KAUSA*) : l'allocation est réalisée de manière centralisée suivant un échéancier temps-fréquence. *KAUSA* alloue les ressources nécessaires à chaque flux en fonction des contraintes et des besoins. Nous évaluons les performances de l'algorithme en comparant à l'existant. Nous étudions la mise en œuvre de *KAUSA* dans le cadre proposé par *6TiSCH*. Les résultats des simulations montrent la pertinence de l'approche.

1.4 Organisation du document

1.4.1 Chapitre 2 : « Les réseaux de l'*IoT* : état de l'art et prérequis techniques »

Dans le Chapitre 2, nous présentons les applications caractéristiques de l'*IoT*. Nous justifions le choix d'une topologie de réseaux multi-sauts sans fil. Nous listons les mécanismes d'accès au médium permettant de différencier les transmissions et de les rendre fiables. En particulier, nous détaillons le protocole *IEEE Std 802.15.4-2015 mode TSCH*. Nous étudions le choix des mécanismes de routage permettant de garantir une certaine *QoS* tout en préservant le bon fonctionnement du réseau. Enfin, nous abordons la gestion de réseau par un opérateur et en particulier l'observation et l'ordonnancement des ressources par flux.

1.4.2 Chapitre 3 : « Spécification des *SLA* pour les réseaux de l'*IoT* »

Le Chapitre 3 explique la construction des contrats de *QoS* que proposons d'établir entre un opérateur de l'*IoT* et chacun de ses clients. Nous donnons les modèles existants de *SLA*, en particulier dans le domaine des *web services*. Nous détaillons la structure et les outils de construction des *SLA* que nous utilisons dans le domaine de l'*IoT*. En particulier, nous expliquons la mise en place de métriques spécifiques et de garanties couvrant un ensemble de nœuds.

1.4.3 Chapitre 4 : « Architecture de gestion opérée de réseaux de l'*IoT* »

Dans le Chapitre 4, nous introduisons une architecture fonctionnelle donnant la possibilité à un opérateur multi-service de mettre en œuvre un *SLA* pour chacun de ses clients. Nous précisons les éléments de l'infrastructure de l'opérateur, puis nous listons les fonctions essentielles à la gestion des applications. Nous détaillons la manière de traduire les *SLA* en termes de ressources réseau, afin d'évaluer si leur mise en œuvre est viable. Nous montrons comment superviser le réseau afin de vérifier son bon fonctionnement, et comment le contrôler pour maintenir les performances attendues.

1.4.4 Chapitre 5 : « Observation efficace des réseaux de l'*IoT* opérés »

Le Chapitre 5 montre plus précisément comment observer la réalisation des *SLA* sur le réseau et comment collecter les informations d'observation. Nous revenons sur le contexte, les objectifs, les défis et les solutions existantes pour l'observation des réseaux de l'*IoT*. Puis nous proposons un mécanisme efficace en termes d'utilisation de ressources pour la remontée d'informations d'observation et la vérification des *SLA*. Nous évaluons notre proposition par simulation.

1.4.5 Chapitre 6 : « Allocation déterministe pour l'*IoT* opéré »

Dans le Chapitre 6, nous traitons de l'allocation de ressources sur les réseaux multi-sauts utilisant un multiplexage en temps et en fréquence tel que proposé par *6TiSCH*. Nous modélisons d'abord les réseaux multi-sauts opérés compte tenu de nos hypothèses de topologie, de trafic et de propagation. Nous proposons deux approches de sur-dimensionnement des ressources temps-fréquences, basées sur un mécanisme de retransmissions. Celles-ci permettent d'assurer une probabilité donnée de respect des exigences de fiabilité des *SLA*. Nous proposons l'algorithme *KAUSA* qui ordonnance les ressources temps-fréquences dans le but de respecter les *SLA* en termes de fiabilité et de délai.

Chapitre 2

Les réseaux de l'*IoT* : état de l'art et prérequis techniques

Dans le chapitre précédent, nous avons introduit l'approche opérateur pour réseaux de l'Internet des Objets (*IoT*). Opérer un réseau de capteurs consiste à maîtriser le transit des données des usagers entre les capteurs et actionneurs et les points de collecte et de contrôle du réseau.

Ici, nous expliquons d'abord quels sont les besoins des applications de l'*IoT*. Nous décrivons les solutions existantes permettant de réaliser le transport de l'information tout en garantissant le respect de la qualité de service (*QoS*) attendue par chacune des applications. Nous justifions le choix d'une architecture multi-saut, puis nous détaillons comment la *QoS* y est traitée dans l'accès au médium physique, dans le routage, et dans la réservation et l'allocation de ressources.

Nous montrons comment les éléments standardisés de la pile protocolaire de l'*IETF 6TiSCH* [**?**] favorisent un usage déterministe des ressources et comment l'opérateur peut tirer profit des mécanismes existants pour réaliser les tâches nécessaires à la gestion de réseaux sans fil multi-applicatifs.

2.1 Hypothèses et scénario applicatif pour les réseaux de l'Internet des Objets

Les contributions émergentes dans le domaine de l'*IoT* sont le résultat des recherches sur les réseaux de capteurs sans fil menées depuis 1990 [**?**]. Dans cette étude, les capteurs sur lesquels nous nous focalisons sont des dispositifs électroniques qui mesurent un phénomène physique présent dans leur environnement (e.g. le taux d'humidité dans une zone forestière).

2.1.1 La mesure physique de l'information

Les capteurs physiques sont placés sur des cartes électroniques que nous appelons nœuds. Chaque nœud est composé d'un ou plusieurs microprocesseurs, de mémoire (vive, ou mémoire de masse (*flash*)), permettant de garder sous forme numérique l'information générée par le ou les capteurs, et d'un composant radio (transmetteur et antenne) permettant de transmettre l'information et d'en recevoir d'autres nœuds. Une source d'énergie est associée au nœud (pile, batterie, source solaire, etc.). Plusieurs capteurs peuvent être placés sur un même nœud. C'est le cas par exemple des produits proposés par *HiKob* [**?**] ou par *Arduino* [**?**].

La Fig. 2.1 présente un nœud *M3* de la plate-forme *FIT/IoT-Lab* [**?**]. Il s'agit de la plus importante plate-forme française de tests pour l'évaluation des protocoles pour réseaux de capteurs sans fil. En plus des éléments listés ci-dessus, le nœud présenté inclut un connecteur qui permet à la plate-forme de collecter les informations d'état du nœud, ainsi que de lui dicter le comportement à adopter (e.g. écrire un exécutable sur le microprocesseur).

De nombreuses applications utilisent des informations issues de capteurs et les nœuds se partagent le canal radio [**?**]. *Libelium* répertorie cinquante applications typiques, rassemblées en Fig. 2.2 [**?**]. On peut trouver différents types d'applications :

1. pour améliorer la surveillance et la maîtrise de l'environnement naturel [**?**]. Par exemple, la collecte d'informations de température et d'humidité dans les champs (pour la téléagriculture), ou les forêts (détection d'incendies) ;
2. pour des besoins d'automatisation dans l'industrie. Il s'agit de boucles de contrôle en temps court [**?**], où le réseau de capteurs doit permettre de détecter un problème dans le procédé en cours et d'agir sur les machines (actionneurs) ;
3. la maintenance et la prévention concernant les bâtiments et les infrastructures [**?**]. Par exemple, la mise en service et l'amélioration des conditions des bâtiments (le *building commissioning*) repose sur des informations issues de capteurs [**?**] ; ou bien la surveillance de la structure des ponts ou autres constructions critiques [**?**] (des nœuds radio sont intégrés dans la structure au moment de sa construction) ;
4. pour l'amélioration des conditions de vie et l'économie d'énergie dans les villes : le concept des villes intelligentes (*Smart Cities* [**?**]) repose sur la collecte d'informations de capteurs. On peut citer les applications d'aide au stationnement (*Smart Parking* [**?**]), la gestion des feux de circulation, la surveillance de la pollution de l'air, la gestion des déchets. Le contexte urbain et celui de la voirie rendent difficile la mise en place de solutions radio parce que l'environnement y est brouillé et dense et favorisent des architectures hétérogènes [**?**] ;
5. pour améliorer les services à destination des usagers individuels : la télérelève de compteurs d'eau, de gaz, est un cas typique d'usage de capteurs pour la collecte d'informations spécifiques à chaque foyer [**?**]. Le concept de maison connectée (*Smart Home*) repose également sur l'*IoT*. Les systèmes d'alarmes (fuites, intrusions, accidents domestiques) sont aussi des applications potentielles.

Les contraintes de fonctionnement de ces applications sont hétérogènes (temps de réaction, pertes d'information, etc.). La difficulté de l'opérateur est d'organiser efficacement les nœuds en réseau afin de gérer un maximum d'applications conjointement.

Étant donné que les sources de trafic (les capteurs) sont réparties sur de grandes zones, permettre la connectivité de chacune d'entre elles est difficile. La portée radio des nœuds est variable et limitée par les éléments de l'environnement [?]. L'ensemble des nœuds radio constitue un *réseau de capteurs sans fil* :

1. si les communications sont directes entre sources et points de collecte, on parle de réseau *mono-saut* ;
2. si un ou plusieurs nœuds relais permettent la connectivité des sources en relayant les informations d'un point à l'autre, le réseau est *multi-saut*.

La question est de savoir comment organiser les nœuds entre eux de manière autonome (sans intervention humaine) et efficace en énergie pour réaliser la collecte des informations. Au delà des points de collecte, l'opérateur s'interface avec ses clients par d'autres technologies, indépendamment du réseau de capteurs considéré (e.g. connexions au travers de réseaux privés virtuels (*Virtual Private Network* (*VPN*))).

2.1.2 Caractéristiques du trafic applicatif

Le rapport de l'*ETSI* sur les exigences des applications pour les réseaux maillés [?] pointe les différentes quantités de trafic généré par les capteurs (e.g. pour la télérelève de gaz, au maximum quatre messages applicatifs sont générés par heure et capteur).

En dehors du trafic de machine-à-machine (*M2M*) lié aux usages industriels, le trafic est principalement de remontée (depuis les capteurs vers le ou les points de collecte). Toutefois, le trafic descendant (point-à-point (*unicast*) pour contrôler un actionneur, point-à-multipoint (*multicast*) pour configurer le réseau globalement) est également à prendre en compte. Il existe plusieurs types de trafic [?] :

1. périodique : par exemple pour la télérelève de compteurs d'eau, la période de génération des messages est connue (1 message d'environ 100 octets par jour, pouvant être fragmenté en plusieurs trames) [?] ;
2. sur événement : par exemple pour une application d'alarme de gaz, le trafic est sporadique [?] ;
3. sur demande : par exemple une requête ponctuelle de l'application sur un niveau de pollution [?].

Nous supposons que ces différentes applications coexistent et nous utiliserons ces scénarios pour l'évaluation de la capacité d'un opérateur à respecter les exigences de chaque application.

2.1.3 Critères de performance et garanties attendues

Les clients formalisent généralement les critères de *QoS* de leurs applications sous la forme d'indicateurs clés de performance (*KPI*) [?]. Le temps de transit (i.e. le délai de bout-en-bout [?]) est souvent un *KPI* important (e.g. dans l'industrie ou le *M2M*, pour des boucles de contrôle avec actionneurs). Par exemple, les messages indiquant le remplissage des bennes pour une application de gestion des déchets doivent arriver en moins d'une heure au point de collecte, et nécessitent un acquittement [?]. D'autres applications nécessitent en priorité des garanties de fiabilité : le *KPI* est alors le taux de livraison [?].

Les messages applicatifs peuvent être de taille variable [?]. Il est possible de dissocier la génération de chaque message de son transit sur le réseau : en effet, un message pourra être fragmenté après sa création, s'il est de taille importante, puis placé en mémoire jusqu'à sa transmission complète.

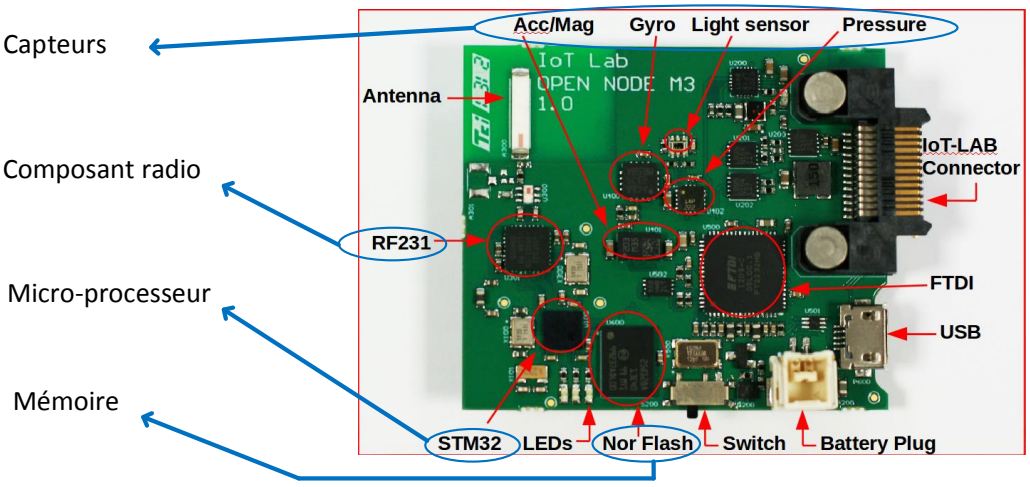


FIGURE 2.1 – Composition d'un nœud de capteur sans fil : exemple d'un nœud *M3* de *FIT/IoT-Lab* [?].

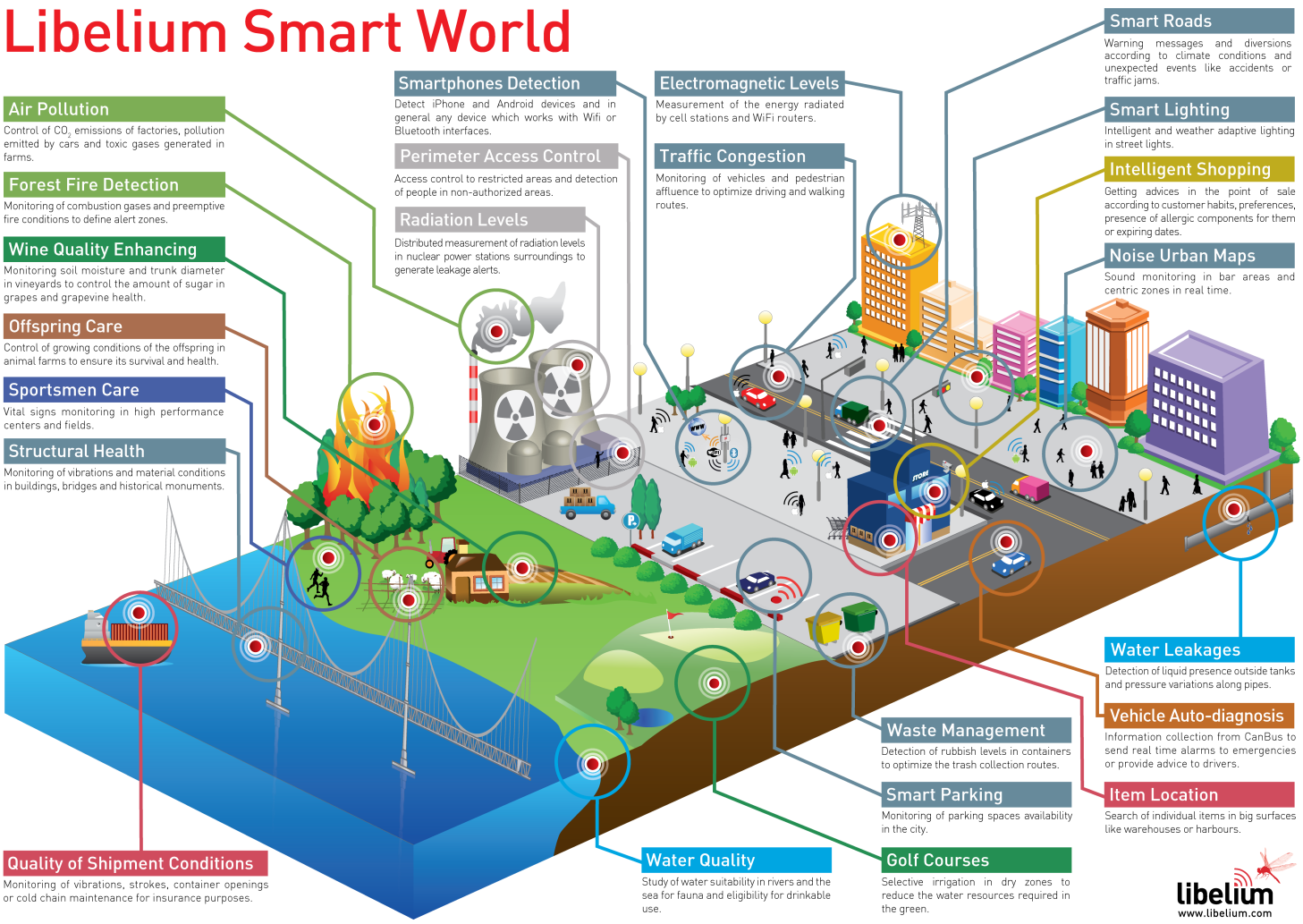


FIGURE 2.2 – Le multi-applicatif dans l'*IoT* (source : *Libelium* [**?**])

Les exigences, exprimées sous la forme de *KPI*, doivent être respectées tout en prenant en compte l'ensemble des contraintes liées à la nature des capteurs sans fil.

2.1.4 Contraintes des capteurs liées à leur placement et usage

Comme les capteurs sont déployés dans des environnements, tant privés (chez les particuliers, e.g. sur les compteurs d'eau) que publics (e.g. un nœud relais sur candélabre), l'opérateur doit prévoir des déploiements de longue durée. Malgré la présence de sources d'énergie dans certains cas, il faut considérer des nœuds de communication contraints par les ressources (énergie, mémoire, puissance de transmission, capacité de processeur) [**?**]. En effet, l'alimentation sur pile est encore aujourd'hui le cas le plus répandu [**?**]. Les nœuds sur pile ne dépendent pas de l'environnement : la pile permet un fonctionnement garanti quelles que soient les conditions. La pile limite aussi les risques d'endommagement car, étant de petite taille, le nœud peut être placé dans un boîtier protecteur.

Contrairement aux équipements classiques de l'internet filaire, les nœuds considérés sont contraints par le coût, la taille, le poids, la puissance et l'énergie disponible [**?**]. Ces contraintes restreignent la taille et l'exécution du code embarqué.

La consommation d'énergie la plus importante pour un nœud est liée au fonctionnement du composant radio [**?**]. Les étapes de communication (les transmissions et les réceptions, l'écoute du canal, la durée d'activité des nœuds) doivent donc être contrôlées afin de préserver l'énergie [**?**] [**?**]. Il est possible de réduire la consommation d'énergie sur chacun des nœuds par le contrôle de la puissance d'émission. Cela impacte la probalité de succès des communications [**?**]. En effet, augmenter la puissance d'émission améliore la portée radio des nœuds, mais augmente aussi le risque d'interférences entre eux. L'opérateur doit ainsi trouver un compromis entre le nombre de nœuds déployés, leur cycle d'activité, la connectivité et la puissance d'émission des nœuds.

Afin de répondre aux besoins applicatifs pour une durée de vie donnée, l'opérateur peut déployer des réseaux contraints (ou *Low-power and Lossy Network* (*LLN*) [**?**]), qui sont des réseaux à basse consommation et sujets à des pertes d'informations.

Pour collecter les informations de manière efficace en énergie et en bande passante, l'opérateur doit proposer des mécanismes additionnels de contrôle des ressources de communication sur les *LLN* (e.g. gestion de l'accès au médium, équilibrage de charge), tout en respectant la contrainte des coûts de déploiement [**?**].

2.1.5 Caractéristiques de propagation radio dans les réseaux contraints

Le taux d'erreur de chaque lien (*Packet Error Rate*, *PER*) est corrélé à l'environnement [**?**]. Pour estimer le *PER*, les modèles théoriques de propagation radio prennent en considération l'affaiblissement de propagation (*pathloss*), le masquage (*shadowing*) et l'évanouissement (*fading*) [**?**], [**?**]. Ces modèles sont affinés par les outils de simulations [**?**] ainsi que les résultats d'expérimentations [**?**]. Ils permettent d'estimer en première approche les relations entre *PER*, distance, atténuation et type de nœud [**?**]. La réalité est souvent différente du modèle, car le *PER* dépend aussi des interférences, des bruits parasites et de l'état des nœuds [**?**].

La Fig. 2.3 montre les variations du *PER* en fonction de la distance. Nous y appliquons le modèle *pathloss* simple, bruit et masquage gaussiens [**?**] sur un grand nombre de nœuds. Les résultats sont obtenus en simulation pour un réseau multi-saut de type *IEEE Std 802.15.4-2015*. Nous y supposons que l'atténuation dépend du type de nœud, selon leur situation (e.g. à l'intérieur d'une maison) et nous choisissons aussi leur puissance de transmission (e.g. les relais de l'opérateur émettent à plus forte puissance). Nous utilisons l'Équation 2.53 de [**?**],

	source	relais		source	relais
pathloss exp. vers relais	3.5	2.5	$\sigma_{Shadow.}$	3.4	3.4
pathloss exp. vers passerelle	-	1.9	σ_{Noise}	0.4	0.4
Puissance de trans. (dBm)	0	3	Bruit (dB)	-65	-65
SINR seuil (dB)	-90	-90	Dist. de ref. (m)	10	22
Constante K [?](dB)	-52	-62			

TABLE 2.1 – Valeurs utilisées pour le modèle simulé de propagation

avec les valeurs présentées en Tab. 2.1. La simplicité du modèle permet, en supposant un faible impact du bruit, de bien distinguer les écarts de qualité et de portée des liens radio liés à la nature de l'environnement. La portée dépasse rarement 50 m dans les environnements enfouis ou intérieurs.

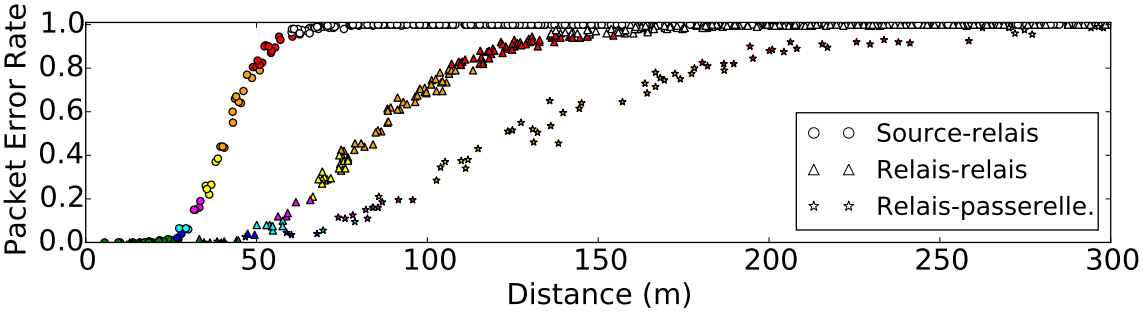


FIGURE 2.3 – Distribution des taux d'erreur (*PER*) par type de liens (nœuds sources, relais, passerelles).

En plus de l'environnement et du positionnement des nœuds, le *PER* évolue dans le temps et l'espace, en fonction de la fréquence utilisée et de la taille des messages [?]. L'opérateur peut caractériser la qualité de chacun des liens à partir du *PER*, estimé par modèle ou mesuré sur le réseau lui même. En supposant les *PER* fixes pendant une période donnée, et les liens indépendants, on peut estimer le taux de livraison, en termes de probabilités, sur un chemin, à partir des *PER* des liens le constituant. Ainsi, l'opérateur peut optimiser l'utilisation de son réseau en fonction des mesures de *PER*.

2.2 Pourquoi s'appuyer sur une topologie multi-saut ?

Le déploiement des capteurs est contraint par les applications : le positionnement des nœuds crée des topologies hétérogènes, comprenant des zones à forte densité de nœuds et de trafic. Les nœuds sont parfois dans des conditions de propagation radio contraintes (nœuds enfouis, isolés) rendant les liens radio asymétriques et les communications plus difficiles [?]. Ces deux aspects (forte densité, conditions restreintes) rendent la mise en place de réseaux de capteurs coûteuse [?].

Plusieurs acteurs et alliances technologiques se partagent les protocoles, standards et les déploiements existants de solutions. Les acteurs les plus connus sont :

1. l'alliance *LoRa* [?], supportée par *Orange*, *Bouygues*, *IBM*, *Cisco*, etc. propose une solution de longue portée et bas débit ;
2. l'opérateur français *SIGFOX* [?], [?], propose également un réseau bas débit sur des communications en bande étroite ;

3. l'alliance *ZigBee* [**?**] basée sur le standard *IEEE Std 802.15.4-2015* ;
4. *Homerider* [**?**] est un exemple pour la télérelève basé sur le standard industriel *Wireless M-Bus* [**?**], et une des technologies utilisées par l'opérateur *M2M m2ocity* ;
5. les réseaux sans fil industriels [**?**] avec les technologies *WirelessHART* [**?**], *ISA100.11a* [**?**], les solutions *SmartMesh* de *Linear Technologies* [**?**] ;
6. la pile protocolaire de l'*IETF 6TiSCH* basée sur le standard *IEEE Std 802.15.4-2015 mode TSCH* [**?**].

Dans la suite de cette section, nous discutons les choix et leur pertinence pour opérer les réseaux de l'*IoT*.

Deux topologies alternatives sont étudiées et utilisées pour transmettre une information sur une longue distance :

1. les réseaux mono-sauts connectant les nœuds clients aux passerelles en un seul lien radio de longue portée. Les technologies *LoRa* et *SIGFOX* font partie de cette catégorie appelée *Low-Power Wide Area Networks* (*LPWAN*) [**?**] ainsi que *Homerider* (avec la mise en place de répéteurs, transparents et à configuration statique, permettant d'augmenter la portée des liens) ;
2. les réseaux multi-sauts avec des relais. C'est le cas pour des déploiements sur zones larges avec le standard *IEEE Std 802.15.4-2015* [**?**], en maille ou en arbre (choix par exemple de l'alliance *ZigBee* [**?**]), des réseaux industriels, et de la pile *6TiSCH*.

Voyons leurs avantages et leurs inconvénients.

2.2.1 Portée, débit, largeur de bande des technologies existantes

Les *LPWAN* privilégient les topologies en étoile car ils utilisent des fréquences basses (en dessous du GHz, e.g. 868 MHz) et ont donc une portée plus longue que les réseaux multi-sauts [**?**] :

1. longue portée de quelques kilomètres à plusieurs dizaines de kilomètres (20, 50) pour les *LPWAN* comme *LoRa* [**?**] ;
2. courte à moyenne portée de plusieurs dizaines de mètres pour les réseaux multi-sauts, e.g. sur la bande de fréquences des 2.4 GHz, cf. Fig. 2.3.

Les *LPWAN* mettent en place des communications bas débit, basse consommation, qui permettent de rendre plus fiables les liens longs [**?**]. Les modulations utilisées offrent un compromis entre la fiabilité du canal et la consommation d'énergie [**?**]. Deux solutions différentes sont adoptées [**?**] :

1. l'utilisation de bandes très étroites (les *Ultra-Narrow Bands* (*UNB*)) pour *SIGFOX* (quelques dizaines de Hertz) ;
2. l'étalement de spectre pour *LoRa* (sur des bandes moins étroites, e.g. 125 kHz).

Le bas débit implique des contraintes sur la taille des messages (12 octets pour *SIGFOX*, un maximum de 256 octets pour *LoRa*) afin de limiter la durée d'émission de chaque message. En effet, l'émission d'un message peut requérir plusieurs secondes pour *LoRa*, tandis que l'émission d'une trame de 127 octets en haut débit (e.g. 2.4 GHz pour *6TiSCH*) occupe une durée de l'ordre de 10 ms.

Le multi-saut exige une organisation des communications sur le réseau lié à l'usage de relais, la gestion des interférences, au choix des routes. Si les techniques *LoRa* peuvent aussi être appliquées en multi-saut, elles perdraient en capacité à cause de ce surcoût de contrôle. En effet, la mise en place de relais *LoRa* imposerait une importante mise en mémoire du trafic et des délais supplémentaires du fait du bas débit.

La Fig. 2.4 illustre ces deux alternatives : les liens $J \rightarrow D$ et $I \rightarrow H$ sont des exemples de liens radio de courte portée sur un chemin multi-saut, $J \rightarrow D$ et $I \rightarrow H$ présentent peu de risques d'interférences, tandis que $A \rightarrow Gw$, $B \rightarrow Gw$, $K \rightarrow Gw$, qui sont des liens longue portée, doivent partager le spectre radio pour leur transmission vers le point de collecte, et présentent ainsi plus de risques de pertes de trames. Pour atteindre D, les nœuds C et J transmettent à moindre coût énergétique et risque d'interférences que les transmissions $B \rightarrow Gw$ [**?**].

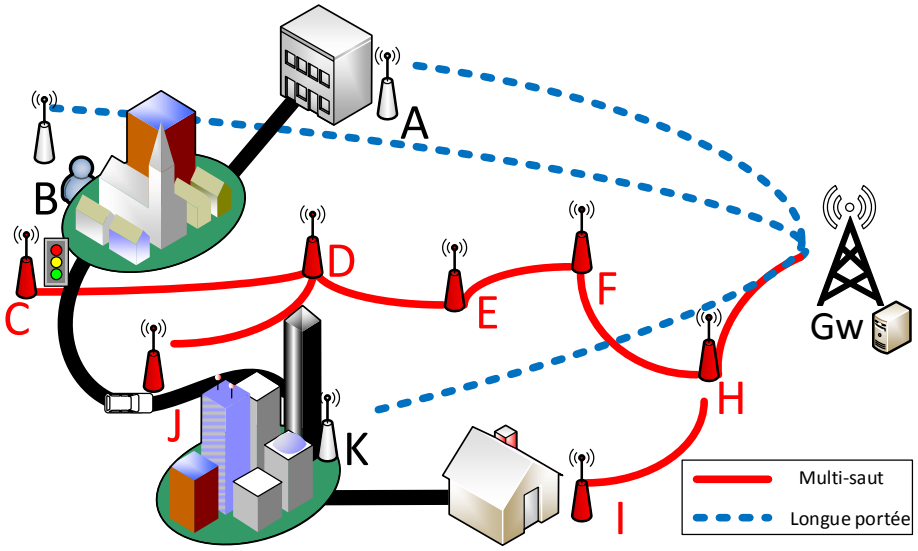


FIGURE 2.4 – Les routes multi-sauts et les liens de longue portée

2.2.2 Comment obtenir la plus grande capacité de trafic ?

LoRa et *SIGFOX* offrent une grande capacité de collecte de trafic. Du fait de la différence entre les nœuds et les passerelles (les nœuds doivent être bon marché et consommer peu d'énergie), les communications descendantes sont plus complexes à mettre en œuvre : les liens sont bidirectionnels mais non symétriques. La taille des messages restreint l'usage des *LPWAN* à des applications où le débit de données attendu est faible.

En revanche, la capacité dans les réseaux multi-sauts est fortement liée au nombre de nœuds installés et à leur agencement (en particulier la capacité du voisinage à deux sauts de chaque nœud) [**?**]. En effet, les nœuds relais sont contraints par leur module radio (qui ne peut effectuer qu'une opération à la fois), et leur espace en mémoire. Si le déploiement de relais ne correspond pas à la distribution géographique des sources de trafic, des goulets d'étranglement se forment, et des pertes de trames ont lieu.

Les solutions de type *Homerider* nécessitent une configuration manuelle et statique des répéteurs (architecture sans routage, à deux sauts, avec un nombre limité de terminaux associés à chaque répéteur). Ces limitations ne permettent pas de s'adapter facilement à une évolution du nombre de terminaux ou à un changement des conditions radio.

Le nombre de passerelles permet d'augmenter la capacité des réseaux multi-sauts. Le contrôle des communications sur plusieurs canaux de fréquences, orthogonaux, permet d'optimiser l'usage des ressources simultanées de communication en augmentant le débit de sortie. L'asymétrie des liens est accentuée aux extrémités des chemins multi-sauts, puisque les liens entre les nœuds relais intermédiaires sont souvent plus homogènes (nœuds de même nature et stratégiquement positionnés). La consommation d'énergie est moins importante

pour des densités de trafic élevées [**?**]. Les messages peuvent être plus longs et fragmentés si c'est nécessaire en plusieurs trames. Les portées courtes offrent un meilleur débit.

2.2.3 Discussion sur le risque d'interférences

Les transmissions sur bande très étroite (*UNB*) et l'accès multiple par répartition aléatoire de fréquence (*Random-FDMA*) permettent à *SIGFOX* une utilisation efficace de la bande de fréquences, en termes de sensibilité par rapport au bruit et aux interférences [**?**]. Pour *LoRa*, le choix des paramètres de facteur d'étalement et le contrôle de la puissance d'émission jouent le même rôle.

En multi-saut, l'impact des interférences entre nœuds voisins est à prendre en compte pour organiser les communications de manière fiable et efficace en termes de capacité [**?**]. Le nombre plus important de liens possibles permet de réduire l'impact des interférences en choisissant un routage adéquat, plus fiable, au détriment de la capacité. En effet, nous pouvons éviter les conflits en tirant parti de la connaissance des voisinages des nœuds.

L'utilisation de bandes sans licences rend les interférences plus probables dans les deux cas (longue portée ou multi-saut). Le fait d'avoir des liens courts permet d'ajuster la configuration du réseau en temps et en fréquence à l'environnement local de chaque nœud et ses évolutions. Par exemple, l'opérateur peut cesser d'utiliser un canal dont la qualité s'est dégradée à un endroit donné du réseau.

2.2.4 Adéquation aux exigences industrielles

Dans l'industrie, nous pensons que l'automatisation des procédés par les boucles de contrôle radio ne requiert généralement pas de très longues portées (centaine de mètres). Mais les applications ont des exigences de *QoS*, en termes de délai et de fiabilité [**?**]. En particulier, l'asymétrie des communications dans les *LPWAN* rend difficile la mise en place de boucles de contrôle. En effet, le contrôle d'un actionneur demande un temps de réponse inférieur à deux secondes [**?**].

En multi-saut, le délai de transit de l'information est variable, car chaque trame peut être gardée en mémoire plus longtemps dans les nœuds. En effet, ceux-ci doivent maintenir un rapport cyclique d'activités réduit pour assurer une durée de vie raisonnable (10 à 15 ans [**?**]) et respecter la régulation en termes d'émissions électromagnétiques [**?**]. Le temps de transit inclut donc les périodes d'inactivité de chaque nœud intermédiaire. Les communications doivent être organisées pour répondre aux besoins en délais.

En multi-saut, des retransmissions adaptées localement peuvent permettre de respecter un taux de livraison de bout-en-bout. Elles sont moins coûteuses en énergie et plus efficaces que des retransmissions de bout-en-bout d'un message entier telles qu'avec *LoRa*.

Toutes ces raisons favorisent l'étude des solutions multi-sauts industrielles, comme *WirelessHART* ou *ISA100.11a* [**?**]. *WirelessHART* est la version radio du protocole industriel *Highway Addressable Remote Transducer (HART)*. En utilisant la bande des 2.4 GHz, une entité centrale de gestion du réseau planifie des communications radio multi-sauts entre les capteurs (capables de réexpédier le trafic d'autres nœuds) et une passerelle. *WirelessHART* évite les goulets d'étranglement et les pertes de connectivité en choisissant des routes alternatives, et contrôle les canaux de fréquences utilisés en évitant les interférences.

ISA100.11a est un standard publié en 2009 par l'*ISA* (*International Society of Automation*) ayant les mêmes objectifs concernant la fiabilité. *ISA100.11a* est plus flexible que *WirelessHART* [**?**] :

1. la taille des intervalles de temps est ajustable ;
2. l'architecture comprend une épine dorsale (ou *backbone*) de nœuds qui permet d'interconnecter plusieurs sous-réseaux ;
3. *WirelessHART* privilégie l'usage du protocole *HART* pour la couche applicative tandis qu'*ISA100.11a* ne favorise aucun protocole particulier ;
4. *ISA100.11a* implémente plusieurs algorithmes de saut de fréquences.

6TiSCH englobe les travaux de *WirelessHART* et *ISA100.11a*, en permettant leur interopération [**?**].

Dans notre cas, la nécessité pour l'opérateur d'offrir des garanties de *QoS* correspond aux objectifs énoncées dans la charte de *6TiSCH* [**?**] : doté des prévisions de trafic et de l'observation du réseau, l'opérateur peut modifier et contrôler le comportement du réseau vis-à-vis des exigences des applications [**?**].

2.2.5 Coûts et risques de déploiement

Le coût de déploiement initial des solutions de *LPWAN* est moindre : pour des topologies en étoiles, le nombre de nœuds, équipés pour le bas débit, est moins important.

Dans le cas multi-saut, la gestion de la transmission de l'information à chaque relais est plus difficile : il faut prendre en compte l'impact de la qualité de chaque lien sur les chemins multi-sauts. Le risque est que le déploiement ne rende pas la connectivité possible pour tous les nœuds du réseau [**?**].

En revanche, en multi-saut, l'opérateur peut dimensionner son réseau de manière redondante (déploiement de plus de relais que nécessaire), afin de prévoir de nouveaux trafics, des changements dans l'environnement radio, la défaillance d'un relais, etc. En longue portée, une panne sur une passerelle non redondée entraîne une perte de connectivité pour tous les nœuds qui lui sont attachés.

Les technologies industrielles et *6TiSCH* offrent un accès déterministe au réseau. Les blocs temps-fréquences, qui seront détaillés en Section 2.4, correspondent aux ressources minimales de communication dans *6TiSCH*. Ceux-ci permettent à l'opérateur de quantifier l'usage des ressources, de les organiser de manière déterministe et d'anticiper des problèmes futurs.

2.3 Quelle *QoS* dans les protocoles d'accès au médium ?

Une fois la topologie choisie, il faut assurer la transmission des informations d'un nœud à un autre, en respectant les critères attendus pour chacune des applications. La couche d'accès au médium radio a un impact majeur sur la *QoS* dans les réseaux radio multi-sauts, et fait l'objet d'études nombreuses [**?**] :

1. un nœud a un ensemble de trames en mémoire, à transmettre à un ou plusieurs voisins ;
2. il doit choisir quelles sont les trames à envoyer en premier ;
3. il doit choisir quel canal de fréquences utiliser parmi les possibilités offertes par la couche physique ;
4. il doit éviter les collisions avec d'autres émetteurs ;

5. il ne sait pas si l'ensemble des récepteurs visés est prêt à recevoir une trame ;
6. il doit vérifier que chaque trame a bien été reçue.

Nous évoquons les solutions présentes dans l'état de l'art en présentant celles que nous choisissons et pourquoi.

2.3.1 Protocoles d'accès à préambule

L'assurance du succès d'une transmission est l'enjeu de l'accès au canal radio et le principal paramètre sur lequel l'opérateur peut s'appuyer pour garantir de la *QoS*.

Jusqu'à présent, l'usage des réseaux de capteurs se fait majoritairement pour de faibles demandes de trafic. La faible occupation du canal et le fait que les arrivées de trafics ne soient pas corrélées (sauf dans les cas d'applications de type alarme) justifie l'accès direct au canal : une tentative de transmission est initiée dès qu'un nœud a une trame à transmettre. Cette approche impose au récepteur une écoute continue coûteuse en énergie.

Une première famille de protocoles d'accès au médium pour les réseaux de capteurs sans fil définit un accès direct au canal avec préambule [**?**]. L'***échantillonnage de préambule*** permet une consommation d'énergie faible pour des trafics faibles. Un nœud voulant transmettre une trame l'annonce en émettant un préambule long. Les nœuds voisins, qui scrutent périodiquement le canal (échantillonnage du canal), ne se préparent à recevoir une trame que si de l'énergie est détectée (écoute d'un préambule). Le préambule doit donc être plus long que la période d'échantillonnage afin de garantir sa détection.

Cette première amélioration (l'***échantillonnage de préambule***) est mise en place au début des années 2000 sur les réseaux de capteurs sans fil [**?**]. Elle supporte de faibles trafics mais rend les collisions nombreuses pour des voisinages et des trafics denses, car rien n'empêche plusieurs nœuds de transmettre en même temps.

2.3.2 Protocoles d'accès à contention

Une deuxième amélioration consiste à imposer aux émetteurs d'écouter le canal avant leur tentative de transmission. Chaque nœud vérifie la disponibilité du canal en amont de toute tentative de transmission en utilisant son module radio (utilisation du *Clear Channel Assessment (CCA)* de la couche physique). ***L'écoute préalable*** est prise en compte par l'intégration aux réseaux de capteurs sans fil du protocole *CSMA*, en particulier de *CSMA/CA* (*Carrier Sense Multiple Access with Collision Avoidance*), présenté par *Phil Carl* en 1990 [**?**] puis intégré dans les spécifications du *Wi-Fi* (IEEE 802.11, 1995). Le *Wi-Fi* n'est pas spécifié pour les réseaux contraints (les débits, puissances d'émission et l'activité permanente de la radio le rendent trop énergivore). Les deux améliorations (***échantillonnage de préambule***, ***écoute préalable***) peuvent être mises en œuvre conjointement.

Ensuite, le mécanisme probabiliste de ***transmission différée*** permet de limiter les collisions : si le canal est occupé, ce mécanisme d'attente de durée aléatoire (*random backoff*) permet à plusieurs nœuds souhaitant transmettre une trame de séparer leurs futures tentatives d'accès en les étalant aléatoirement dans le temps.

Ces trois mécanismes ne garantissent pas l'absence de collisions car deux nœuds éloignés mais ayant un destinataire commun dans leurs voisinages peuvent ne pas détecter leurs transmissions vers celui-ci (problème du nœud caché [**?**]). De plus, en cas de fort trafic, le temps d'attente, cumulé éventuellement sur plusieurs tentatives infructueuses avant de pouvoir transmettre, ne peut pas être garanti borné.

Le nœud récepteur peut compléter l'information de disponibilité du canal par sa propre vision locale. Le mécanisme *RTS/CTS* formalise cet arbitrage :

1. un nœud émet une requête (information courte nommée *Request To Send (RTS)*) indiquant son intention de transmettre ;
2. la disponibilité du canal est validée par la réponse unique du récepteur (réponse courte nommée *Clear To Send (CTS)*) :
   (a) si plusieurs nœuds veulent transmettre sur une même période de temps, le récepteur choisit un unique émetteur (le gagnant dans la contention) ;
   (b) si le récepteur n'est pas disponible (e.g. en transmission, en limite de mémoire, en veille), il y a échec de la tentative.

L'échange *RTS/CTS* constitue une synchronisation réactive des nœuds car ceux-ci s'accordent sur la date et la durée de la transmission à venir [**?**].

À l'inverse, afin de limiter le temps d'écoute du récepteur, *IEEE Std 802.15.4-2015* propose de synchroniser les nœuds de manière proactive (notamment avec la diffusion périodique de balises) et de restreindre la contention à une fenêtre de temps dédiée (nommée *Contention Access Period (CAP)*). Le choix du mécanisme de synchronisation réactive/-proactive dépend du trafic attendu (le coût relatif de la synchronisation devient faible pour des trafics importants). Pendant la *CAP*, le récepteur reçoit les trames (si plusieurs nœuds transmettent en même temps leurs trames, il y a collision et échec de leurs tentatives et les nœuds diffèrent leurs transmission).

Un nœud nommé *coordinator* organise les transmissions de chaque émetteur de manière à éviter les collisions : il prévoit des intervalles de temps garantis sans collisions (nommés *Guaranteed Time Slots (GTS)*) puisque séparés dans le temps [**?**]. Les *GTS* forment une période sans contention (nommée *Contention Free Period (CFP)*). Le choix des tailles de *CAP*, de *CFP* et de période de réveil définissent le rapport cyclique d'activités des nœuds (*CAP* et *CFP* forment une structure appelée *superframe*).

Le mécanisme d'arbitrage par contention Lorsque plusieurs émetteurs souhaitent utiliser le canal pour émettre, il faut choisir lesquels le peuvent et dans quel ordre.

Cet arbitrage peut être ajusté afin de différencier les services en termes d'accès : suivant la trame à transmettre, les nœuds peuvent modifier les paramètres comme la durée de l'écoute préalable, la fenêtre de contention, la moyenne de la durée d'attente aléatoire [**?**]. Toutefois, ces modifications ne permettent pas d'offrir des garanties de *QoS* : la probabilité de gagner une contention (i.e. de pouvoir transmettre) reste assujettie aux aléas de l'échange *RTS/CTS*, et à la concurrence entre nœuds, indépendamment de la priorité accordée aux trafics.

En présence de trafics et voisinages denses, l'accès par contention devient moins performant. La multiplication des tentatives infructueuses augmente le surcoût en ressources de communications (retransmissions) et en délai (transmission différée). Il faut dans ce cas privilégier les protocoles de contrôle d'accès au médium à synchronisation [**?**].

Le mode *Deterministic and Synchronous Multichannel Extension* (*DSME*) du standard *IEEE Std 802.15.4-2015* [**?**] ajoute à la structure de *superframe* la dimension multi-fréquentielle [**?**]. En plus de déterminer quel intervalle de temps allouer à chaque transmission, le canal de transmission est choisi pendant la *CAP*. En ajoutant un degré de liberté (le choix du canal) pour l'arbitrage des opportunités de transmission, *DSME* améliore la capacité du standard. *DSME*, en conservant une période de contention, ne permet pas de garantir une *QoS* déterministe car en cas de forte densité de trafic, les concurrents non-prioritaires n'obtiennent pas de *GTS* et sont sujets à des délais non contrôlés.

2.3.3 Protocoles d'accès déterministes

Par opposition aux mécanismes de contention qui introduisent une part d'aléatoire dans l'organisation des opportunités de transmission, la famille des protocoles d'accès déterministes prévoit *a priori* les ressources allouées à chaque trafic. Ces approches supposent souvent les demandes de trafic connues et l'environnement stable (interférents extérieurs, topologie).

La planification des transmissions impose la synchronisation des nœuds [**?**]. Elle repose sur la division de l'accès au médium en intervalles de temps réguliers. Ce multiplexage temporel, dénommé *Time Division Multiple Access* (*TDMA*), a été développé sur couche *MAC* pour réseaux de capteurs mais aussi pour des technologies filaires et cellulaires (mobiles). Il permet d'isoler des flux de trafic et de réserver le long d'un chemin, des ressources de communication [**?**]. On limite ainsi le risque d'interférences entre voisins puisque les communications sont planifiées selon la connaissance des conflits potentiels. Le surcoût provient alors de la mise en place et du maintien de la synchronisation entre les nœuds.

Notons qu'il est possible d'utiliser un mécanisme de contention dans certains intervalles de temps, ce qui rend plus flexible la planification et profite des avantages des deux approches. Le standard *IEEE Std 802.15.4-2015* le propose pour les intervalles utilisant *CSMA/CA*, au détriment de la garantie de *QoS*. Cette solution n'est pas déterministe sur les intervalles concernés.

En combinant le choix du support fréquentiel avec le multiplexage temporel, à l'instar de *DSME*, on découpe l'accès au médium en une matrice d'opportunités de transmissions à trois dimensions :

1. la dimension fréquentielle ;
2. la dimension temporelle ;
3. la dimension topologique : l'ensemble des liens non interférents.

Ce multiplexage *Frequency-Time Division Multiple Access* (*FTDMA*) en blocs temps-fréquences est utilisé et standardisé pour l'industrie (e.g. dans *WirelessHART*) et dans les spécifications ouvertes pour l'internet (avec *6TiSCH*). Il impose au protocole d'accès la connaissance de la topologie et la maîtrise des interférences. En effet, sur un bloc temps-fréquence, deux communications interférentes sont susceptibles de générer des collisions et réduire la *QoS*.

Le choix des blocs temps-fréquences peut se faire de manière proactive ou réactive. Pour l'ordonnancement (choix proactif), de même que pour le routage, *6TiSCH* étudie conjointement des solutions distribuées d'ordonnancement ou des solutions centralisées, qui répondent aux exigences des applications (industrie, maison connectée, bâtiment connecté, réseaux d'énergie (les *Smart Grids*)) telles que perçues par l'*IETF* et discutées au sein du groupe *ROLL* [**?**]. L'ordonnancement centralisé permet de connaître les paramètres globalement (le contrôle des ressources utilisées, donc le rapport cyclique d'activités des nœuds, leurs charges de trafic).

Nous détaillons cet accès en *FTDMA* avec *IEEE Std 802.15.4-2015 mode TSCH* en Section 2.4.

2.3.4 Gestion des transmissions des trames à chaque nœud

En parallèle de l'accès au médium, le traitement des files de trames au sein des nœuds joue un rôle important dans la différenciation des applications.

Ordre de priorité des trames dans la mémoire Dans la mesure où les nœuds ne peuvent envoyer qu'une trame à la fois, l'accumulation non contrôlée de trames en mémoire augmente le délai subi par celles-ci. Il est donc préférable d'attribuer des priorités aux trames suivant les exigences de leurs applications.

Une première différenciation de service peut être mise en place sur l'ensemble des trames en mémoire. Les critères de différenciation sont multiples, soit statiques (e.g. la source de la trame) soit dynamiques (e.g. la distance parcourue) [**?**]. Ces critères sont ajoutés et lus dans les en-têtes des trames (e.g. le champ *DSCP* du protocole *IP* définissant la classe de service). L'ordonnancement des trames peut modifier le comportement simple par défaut (première trame entrée, première sortie, *First In First Out* (*FIFO*)) pour prendre en considération un de ces critères : e.g. en répartissant le nombre de trames sortantes en le pondérant pour chaque classe de service (ordonnancement cyclique pondéré, *Weighted Round-Robin (WRR)*). Si le modèle de différenciation de service (*DiffServ*) favorise les applications les unes par rapport aux autres, il ne permet pas de garantir le respect d'une contrainte de *QoS* (e.g. respect d'un délai de bout-en-bout sur un chemin).

Il est préférable d'isoler les flux applicatifs en plaçant leurs trames dans plusieurs espaces de mémoire contrôlés de manière indépendante. On peut alors ordonnancer les trames dans chaque file suivant les contraintes (e.g. *FIFO* avec effacement des trames obsolètes pour faciliter le respect d'une contrainte de délai). L'isolation permet de respecter l'accès déterministe différencié pour chaque trafic isolé : les ressources (opportunités de transmission) sont spécifiques à chaque trafic afin de garantir la *QoS*.

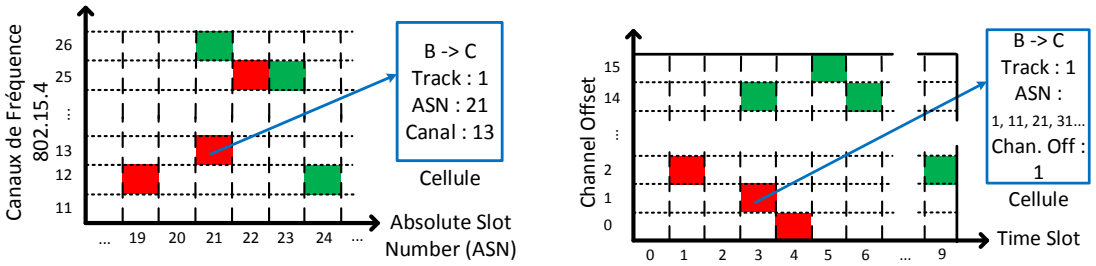
Garantie du succès des transmissions Afin de vérifier que les trames sont bien reçues au niveau du récepteur, le protocole d'accès repose sur l'émission d'acquitements. Un acquittement, de par sa taille réduite, subit peu de collisions, d'autant moins que le canal est déjà réservé par l'émetteur. La probabilité de perte d'acquittements et donc de faux positifs (diagnostiquer une perte de trame alors que c'est l'acquittement qui est perdu), est donc faible. La taille réduite permet en plus d'en réduire l'impact sur la consommation de ressources. En *TDMA*, les intervalles sont dimensionnés pour inclure une trame de données et un acquittement.

En cas de détection d'une perte de trame, la mise en place de retransmissions permet aux nœuds d'améliorer la robustesse face au bruit et aux interférences. Elles augmentent le délai, la fiabilité et l'occupation mémoire des nœuds. Un compromis doit donc être trouvé selon les exigences de *QoS*.

2.4 Le multiplexage temps-fréquence dans *IEEE Std 802.15.4-2015* mode *TSCH*

IEEE Std 802.15.4-2015 mode TSCH [**?**] est un standard de couche *MAC* se focalisant sur les réseaux sans fil industriels [**?**]. Ceux-ci requièrent une haute fiabilité, ce qui correspond également aux besoins de l'opérateur pour le multi-service.

L'amendement *IEEE 802.15.4e* (première version en 2007) regroupe différents modes qui améliorent le fonctionnement général du standard (*IEEE 802.15.4*) et répondent à des besoins d'applications spécifiques (e.g. pour le *RFID*). L'amendement est dorénavant intégré au standard *IEEE Std 802.15.4-2015*. Plus que les autres modes présentés dans le standard (comme *DSME*, ou *LLDN*), le mode *Timeslotted Channel Hopping* (*TSCH*) de *IEEE Std 802.15.4-2015* permet de planifier l'organisation des communications, pour restreindre la compétition entre chaque nœud pour l'accès au médium.



(a) Le multiplexage *FTDMA* sur *IEEE Std 802.15.4-2015 mode TSCH* à 2.4 GHz

(b) Un échéancier pour *IEEE Std 802.15.4-2015 mode TSCH* à 2.4 GHz

FIGURE 2.5 – La mise en place des échéanciers dans *IEEE Std 802.15.4-2015 mode TSCH*

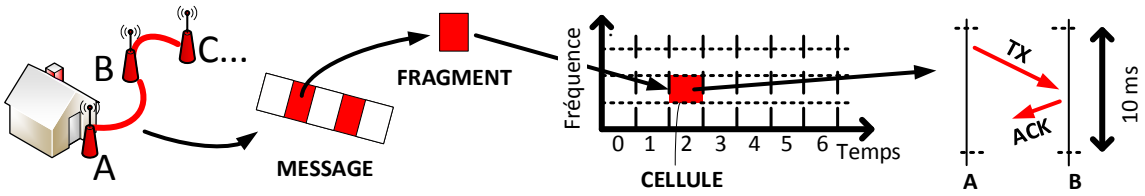


FIGURE 2.6 – Transmission sur un échéancier *IEEE Std 802.15.4-2015 mode TSCH*

En particulier, dans un contexte multi-applicatif induisant une forte densité de trafic, le protocole d'accès tire parti de l'utilisation simultanée de plusieurs canaux de communication. En effet, la bande de fréquences est divisée en plusieurs canaux de fréquences orthogonaux. Ce multiplexage fréquentiel rend plus robuste l'utilisation des ressources radio en réduisant le risque d'interférences internes et de collisions [**?**].

Le choix d'un canal particulier dans la bande de fréquences se fait de manière aléatoire pour *SIGFOX*, cyclique pour le saut de fréquences dans *TSCH*. Il permet d'éviter l'utilisation d'un canal de mauvaise qualité du fait du bruit externe. Une fois choisi, le canal doit être partagé entre émetteur et récepteur(s) : le choix peut être configuré par une entité tierce ou négociée de manière distribuée entre les nœuds.

TSCH permet la construction d'échéanciers de communication (ou *schedules*) pour un réseau de capteurs multi-saut. La Fig. 2.5 montre la division en temps et en fréquences (Fig. 2.5a) : chaque bloc temps-fréquence, ou *cellule*, constitue une ressource de communication, qui peut être utilisée par plusieurs liens. Cette division du médium prend la forme d'un échéancier (Fig. 2.5b) où toutes les cellules sont identifiées :

1. pour une application ou un client avec la notion de *track*, ressources dédiées pour un flux donné, cf. Section 2.6.3 ;
2. pour plusieurs liens entre plusieurs nœuds (généralement 1 lien entre 2 nœuds) ;
3. temporellement par un *timeslot* donné, cf. Section 2.4.1 ;
4. en termes de fréquence par un *Channel Offset* donné, cf. Section 2.4.3.

Sur chaque cellule une transmission d'informations entre deux nœuds voisins peut avoir lieu. La Fig. 2.6 montre que sur une cellule, deux nœuds peuvent réaliser l'échange d'une trame (un fragment d'un message) et de l'acquittement correspondant.

2.4.1 Organisation temporelle

Dans le but de mettre en place le multiplexage temporel du canal, *IEEE Std 802.15.4-2015 mode TSCH* tire parti des expériences en *TDMA* [**?**] : tout au long de la durée de fonctionnement du réseau de capteurs, le temps est ainsi divisé en intervalles réguliers (les

timeslots ou *slots*). Les nœuds doivent donc maintenir une synchronisation afin d'identifier de manière commune les mêmes intervalles de temps et ainsi pouvoir organiser les communications.

Pour partager un échéancier, l'ensemble des nœuds doivent être synchrones [**?**]. L'encapsulation d'informations temporelles sur les échanges de données permet de synchroniser les horloges des nœuds. En absence de données, *IEEE Std 802.15.4-2015 mode TSCH* permet le maintien de cette synchronisation en utilisant la diffusion d'informations de contrôle.

Chaque intervalle est estampillé de manière unique sur l'ensemble du réseau : l'estampillage, ou horodatage, prend la forme d'un *Absolute Slot Number* (*ASN*), incrémenté à chaque période [**?**], et permet aux nœuds d'identifier chaque instant avec précision.

On construit ainsi un ensemble d'allocations de cellules, déterminées dans le temps par l'*ASN*. Chaque bloc temps-fréquence, ou *cellule*, compose l'accès au médium sous la forme d'une matrice (cf. Fig. 2.5a).

On représente une portion de cette matrice temps-fréquence sous la forme d'un échéancier. L'ensemble des intervalles qui constituent l'échéancier est nommé *slotframe*. Par exemple, la Fig. 2.5b montre un échéancier sur une *slotframe* de 10 intervalles (de 0 à 9). La *slotframe* ainsi définie est répétée de manière périodique. La taille de la *slotframe* est un paramètre modulable du protocole.

Le standard prévoit que la durée d'un intervalle de temps soit suffisante pour la transmission d'une trame et de son acquittement correspondant (Fig. 2.6). La taille des trames doit donc être limitée, en fonction du débit de transmission. Les valeurs typiques proposées dans les mises en œuvre existantes de *IEEE Std 802.15.4-2015 mode TSCH* sont :

1. des *timeslots* de durée 10 ms ;
2. taille maximale de trame de 127 octets ;
3. débit de 250 kbps ;
4. taille de *slotframe* de plusieurs centaines de *timeslots*.

Pour que la transmission se déroule correctement dans un intervalle de taille fixe, les messages applicatifs trop longs doivent être fragmentés pour que chaque fragment constitue une trame de taille inférieure au maximum, qui sera transmise sur une cellule (Fig. 2.6). Chaque trame est constituée de la donnée applicative en elle même (la *payload*) et des informations nécessaires à son identification le long de son parcours sur le réseau (les entêtes des couches supérieures, e.g. ceux liés à la fragmentation, cf. Fig. 2.7). C'est la sous-couche d'adaptation *IPv6 Low power Wireless Personal Area Networks* (*6LoWPAN*) [**?**] qui permet cette fragmentation.

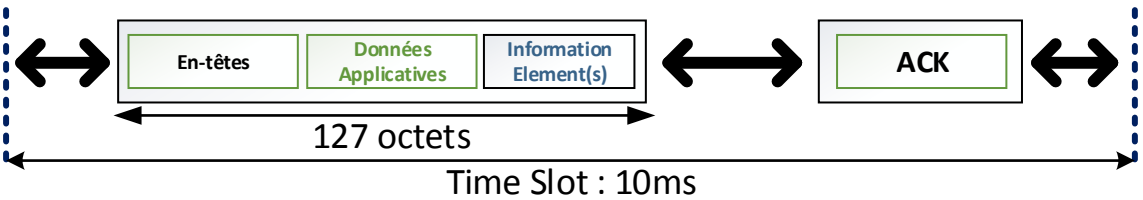


FIGURE 2.7 – Une cellule de communication dans *IEEE Std 802.15.4-2015 mode TSCH*

2.4.2 Organisation fréquentielle

En plus du multiplexage temporel, *IEEE Std 802.15.4-2015 mode TSCH* divise le médium en plusieurs canaux de fréquences indépendants. Le standard envisage pour la couche

physique l'utilisation de plusieurs bandes de fréquences selon les pays et les technologies, eux mêmes à diviser en canaux [**?**] :

1. 868–868.6 MHz : bande utilisée par *SIGFOX* ;
2. 902–928 MHz : division possible en 10 canaux ;
3. 2400–2483.5 MHz : division recommandée en 16 canaux ;
4. 314–316 MHz, 430–434 MHz et 779–787 MHz en Chine ;
5. 950–956 MHz au Japon.

La Fig. 2.8 extraite de [**?**], permet de comparer l'usage de la bande de fréquences de 2.4 GHz par différentes technologies radio, la forme et la distribution fréquentielle des canaux utilisés, et d'observer les potentielles interférences entre les différentes technologies.

Les écarts de fréquence entre les différents canaux (e.g. 75 MHz séparent le canal 11 du canal 26 pour *IEEE Std 802.15.4-2015*) n'ont pas d'impact important sur l'organisation temporelle (e.g. taille des intervalles). Deux émissions voisines sur deux canaux différents n'entrent pas en conflit. Plusieurs trames peuvent être transmises en même temps sur des canaux indépendants. *FTDMA* complète ainsi les avantages des technologies multi-canaux en réduisant l'impact des interférences [**?**].

Avec *FTDMA* l'allocation de ressources est déterminée car contrairement à ce qui a lieu dans *CSMA*, il n'y a pas de processus de contention à l'accès au médium (tous les canaux sont orthogonaux). L'allocation de ressources est maîtrisée, contrairement à la compétition pour le canal dans *CSMA*.

Le multiplexage étant fréquentiel et temporel, la gestion des ressources est un problème bidimensionnel, plus complexe, contrairement au *TDMA* qui ne s'appuie que sur la dimension temporelle.

2.4.3 Le mécanisme de saut de fréquences

IEEE Std 802.15.4-2015 mode TSCH définit un mécanisme de saut de fréquences (*Channel Hopping*) permettant de réduire l'impact du problème d'interférences externes. En outre, dans un environnement radio donné, certains canaux peuvent être sujets à une moins bonne qualité de transmission que d'autres (utilisation de la bande par une autre technologie, atténuation plus forte, etc.). Les cellules des échéanciers sont choisies à un intervalle de temps donné et à un identifiant de canal donné, les *Channel Offsets* [**?**]. Les *Channel Offsets* vont de 0 à 15 pour les 16 canaux de la bande des 2.4 GHz (Fig. 2.5b).

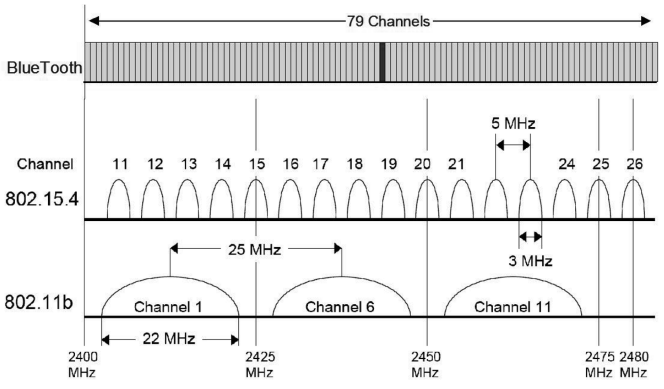


FIGURE 2.8 – Utilisation de la bande des 2.4 GHz [**?**]

À chaque intervalle de temps, le mécanisme de *Channel Hopping* associe à chaque *Channel Offset* un canal de fréquences différent. La relation entre les canaux et leurs identifiants est donnée dans le standard [?] :

$$CH = macHoppingSequenceList[i]$$
$$\text{avec } \; i = (macASN + channelOffset) \;\; mod \;\; macHoppingSequenceLength \quad (2.1)$$

Le canal physique CH est choisi selon l'ASN et le *Channel Offset*, parmi les éléments d'une liste prédéfinie ($macHoppingSequenceList$).

Ainsi, à chaque répétition de l'échéancier, une fréquence généralement différente est utilisée. Par exemple, la cellule présentée sur la Fig. 2.5, à l'intervalle 3, *Channel Offset* 1 de l'échéancier (Fig. 2.5b), est instanciée à l'ASN 21 et sur le canal physique 13 (Fig. 2.5a).

En conséquence, si un canal est en mauvaise condition, le risque de perte de trame lié à son utilisation sera compensé par l'utilisation d'un canal différent à chaque fois. Le *Channel Hopping* tend donc à dissocier les variations de *PER* des choix de *Channel Offset*. Il améliore la fiabilité du réseau en étalant le risque d'utilisation de mauvais canaux sur l'ensemble de la bande de fréquences.

2.4.4 Mise en place de l'échéancier : l'ordonnancement

IEEE Std 802.15.4-2015 mode TSCH permet à des mécanismes de couches supérieures de mettre en place les échéanciers périodiques pour tous les nœuds [?], [?]. Ainsi, les communications sont organisées en multi-saut pour permettre le transfert des informations de leur source jusqu'à leur point de collecte.

L'opérateur construit les échéanciers globalement ou localement en allouant les cellules à chaque flux applicatif, de manière à partager les ressources de communication (les cellules) entre les clients. La construction d'échéanciers (l'ordonnancement ou le *scheduling*) peut être centralisée ou distribuée sur les nœuds [?].

L'ordonnancement centralisé présente plusieurs défauts :

1. d'abord, l'entité chargée de l'ordonnancement est susceptible de tomber en panne ou de perdre sa connexion, la stratégie centralisée augmente les risques d'échecs (problème du *single point of failure*) ;
2. pour calculer des échéanciers de manière globale sur le réseau, il faut en connaître finement l'état. Or la connaissance centrale peut être biaisée (observation ancienne ou erronée). Ainsi, la construction d'échéanciers, déjà complexe du fait de la quantité d'information, peut être pertubée ;
3. le fait de centraliser le calcul augmente le délai d'installation de l'échéancier sur les nœuds, et de mise à jour des informations d'observation permettant de le construire.

En revanche, avec un ordonnancement centralisé, une entité centrale de construction d'échéanciers pourra prévoir les opportunités de transmission pour chacun des liens radio, en évitant les collisions du fait de la connaissance globale du réseau. Les algorithmes centralisés d'ordonnancement sont nombreux. Nous décrirons plus amplement par la suite l'algorithme de *Traffic-Aware Scheduling Algorithm (TASA)* [?], qui prend comme paramètre d'entrée une distribution de trafic connue et dont l'échéancier vise à l'écouler. *TASA* est optimal en termes de compacité [?] : les cellules sont allouées proches les unes des autres, limitant le délai maximal s'il n'y a pas de pertes de trames ; *TASA* alloue de manière centralisée des ressources (cellules) étant donnée une charge de trafic pour chaque nœud [?]. *TASA* ne distingue pas de flux applicatifs mais est performant pour le mono-applicatif à forte densité de trafic [?].

Les solutions distribuées, où les décisions sont prises sur chacun des nœuds en fonction des informations obtenues auprès de leurs voisinages, sont plus flexibles et moins coûteuses en calcul et en temps d'adaptation. Elles ne permettent cependant pas de prendre des décisions en tenant compte de la vision globale du réseau, et donc leurs choix peuvent s'avérer inefficaces ou contradictoires, en termes de consommation de ressources et de respect des contraintes de *QoS*. Les principaux travaux étudiés sont *Disca* [**?**], *D-SAR* [**?**], et plus particulièrement l'ordonnancement réactif avec *Scheduling Function Zero (SF0)* [**?**]. Les allocations de cellules sont négociées entre les nœuds voisins afin de s'adapter au mieux à l'environnement des nœuds et aux demandes de trafic et de *QoS*.

L'échéancier peut aussi introduire des cellules partagées où les nœuds participent à une contention d'accès s'ils ont besoin de transmettre une trame.

2.4.5 Les *Information Elements (IE)*

Le standard définit des *Information Elements* (*IE*) : on utilise chaque trame d'information transitant sur le réseau pour apposer, dans les en-têtes ou avec les données applicatives, des portions délimitées d'information de contrôle (Fig. 2.7). Le format des *IE* comprend, dans l'ordre [**?**] :

1. un type ;
2. un champ *taille de la donnée* ;
3. leur contenu.

Les informations véhiculées dans les *IE* sont multiples : on y trouve par exemple la mise en place de l'échéancier, la synchronisation, des informations de performance ou des commandes de contrôle. On peut aussi utiliser les *IE* pour des besoins spécifiques (e.g. une information liée à des fonctionnalités propriétaires, ou à une couche supérieure, ou à des opérations de contrôle de l'opérateur). Les *IE* permettent d'apposer des informations dans les trames de données jusqu'à atteindre la taille maximale de trame. La Fig. 2.7 montre un exemple où un *IE* contenant des informations (e.g. d'ajustement de synchronisation) est ajouté à l'en-tête d'une trame de données.

Les *IE* transitent de proche en proche, chaque nœud intermédiaire sur un chemin peut récupérer l'information pour lui-même, ou la transmettre de manière transparente (en laissant l'*IE* sur la trame), ou encore modifier, compléter ou ignorer l'information (ajout ou modification des *IE* existants) [**?**].

2.5 Protocoles de routage à Qualité de Service dans les réseaux radio multi-sauts

L'acheminement des données applicatives sur les réseaux radio multi-sauts dépend des choix de routage et de l'accès au médium. Les chemins doivent être choisis afin de respecter les exigences de *QoS* des applications et de prolonger la durée de vie du réseau. Les communications doivent être contrôlées à chaque saut.

De nombreuses solutions de routage existent pour permettre de respecter des contraintes de *QoS* sur réseaux radio multi-sauts [**?**]. Le routage est assujetti aux exigences des applications comprenant notamment la fiabilité, le débit, le délai, dans les environnements industriels [**?**].

2.5.1 Routage réactif

Dans les premières études, les mécanismes de routage considèrent une topologie à plat, c'est-à-dire où tous les nœuds ont le même rôle et la même configuration dans le réseau. Issus des recherches voisines dans les réseaux mobiles ad-hoc (ou *MANET*), *AODV* [**?**] ne convient pas aux opérateurs puisque ceux-ci doivent manœuvrer une topologie hiérarchique convergente (nœuds sources, relais, passerelles). En effet, *AODV* construit des routes dynamiquement par diffusion de requêtes. Cette construction ne serait pas efficace vers un nombre limité de passerelles. Le surcoût et les interférences perturberaient le fonctionnement des nœuds.

L'approche réactive classique (de *AODV* [**?**], *LOADng* [**?**] ou *DSR* [**?**]) permet à un nœud voulant émettre de commander la construction dynamique d'une route vers une destination donnée. Même si celui-ci est établi pour un temps donné, l'inondation de messages de contrôle permettant la construction de chaque route est coûteuse. S'il connaît les caractéristiques de trafic, l'opérateur adopte au contraire un routage peu fréquent. De plus, l'installation d'une nouvelle route requiert un temps de mise en place. Ainsi, ce type d'approche semble inadapté pour notre scénario d'utilisation.

2.5.2 Routage hiérarchique

La hiérarchie en cliques ou *cluster* telle que proposée dans *LEACH* [**?**] ne permet pas non plus de répondre aux besoins de l'opérateur. Le protocole change en effet régulièrement de premier relais (appelé *cluster head*) de manière aléatoire dans le but de répartir l'effort en énergie. Cette technique n'est pas adaptée puisqu'elle ne permet pas de choisir la route empruntée en fonction de la *QoS* attendue. Elle n'offre pas non plus de flexibilité quand au choix du nombre de sauts (par construction, il n'y a qu'un seul relais sur chaque chemin). Enfin, le partage de ressources entre tous les relais pose le problème du passage à l'échelle. *LEACH* a inspiré un grand nombre d'améliorations [**?**]. Celles concernant la fiabilité proposent une approche de différenciation de services, de redondance multi-chemin, ou de stratégie d'évitement des nœuds à trop faible énergie ou des liens à trop faible qualité. Aucune ne permet de garantir de la *QoS* en termes de taux de livraison ou de délai.

2.5.3 Routage géographique

Les nœuds ont la possibilité d'échanger leurs positions géographiques respectives. Ainsi, la décision de routage peut être prise en fonction de la position de la destination et des voisins. Cela corespond à construire un gradient de distance réelle : la distance exprime le positionnement géographique des nœuds par rapport à la passerelle la plus proche.

Cependant, le routage géographique ne permet pas de choisir les liens radio de bonne qualité, ou d'éviter les zones sans relais. La présence d'obstacles ou de sources d'interférences externes réduit les performances du réseau [**?**].

Comme avec d'autres métriques, construire le routage sur la métrique de distance favorise la mauvaise répartition de charges : l'algorithme choisit systématiquement les plus courts chemins en termes de distance.

2.5.4 Routage centralisé

Le routage peut être mis en place par les nœuds de manière autonome (routage distribué) ou par une entité de calcul extérieure (routage centralisé). La collecte d'information globale de routage par tous les nœuds est coûteuse en ressources. Ainsi, le routage distri-

bué se limite en général à construire des routes vers une seule passerelle, à partir des vues locales de chaque nœud, ce qui en réduit la précision.

Au contraire, un routage centralisé collecte et rassemble les informations de l'ensemble du réseau afin de construire globalement la topologie de routage [**?**]. La mise en place du routage pour trafic descendant peut ensuite être réalisée par inondation initiée depuis les points de sortie, ou en descendant depuis un nœud la topologie construite.

Les exigences des applications et le positionnement des nœuds sont exprimées de manière centralisée auprès de l'opérateur, dans les *Service Level Agreements (*SLA*)* [**?**]. La complexité du calcul de routes sur des critères variés de *QoS*, et la quantité d'informations qu'il représente justifient l'approche centralisée : chaque nœud n'a pas les ressources nécessaires pour supporter un algorithme de construction de topologie pour chaque trafic.

L'*Élément de Calcul de Chemin* (*Path Computation Element* (*PCE*) [**?**] est une entité externe au réseau considéré, qui a pour fonction de mettre en place les chemins multi-sauts pour les flux de trafic, sur la base de contraintes à respecter. Ces chemins peuvent aussi être utilisés dans les réseaux en présence d'un protocole de commutation d'étiquettes (comme *MPLS*) afin d'y router les flux de trafic. Dans le cas des *LLN*, le *PCE* pourrait occuper la place de gestionnaire de réseau externe [**?**]. Le *PCE* prévoit son propre protocole d'échange d'information de contrôle (*PCEP*) transporté sur *TCP* [**?**] permettant par exemple d'installer les chemins sur chaque nœud. Nous envisageons le transport d'informations de contrôle sur *UDP* en utilisant *Constrained Application Protocol* (*CoAP*) afin de répondre à la fois aux contraintes des *LLN* et de *QoS*. Le *PCE* est la solution que nous privilégions car il facilite un contôle externe du réseau et peut prendre en compte des exigences de *SLA*.

2.5.5 Routage proactif

Le point de vue de l'opérateur et l'approche des *SLA* motivent la construction du routage de manière proactive (avec des protocoles comme *Routing Protocol for Low-power and Lossy Networks* (*RPL*) [**?**], *CTP* [**?**], ou des entités externes comme un *PCE* [**?**]). En effet, les clauses contractuelles des *SLA* engagent l'opérateur et les clients sur les sources, quantités et exigences des flux applicatifs. On peut donc anticiper les besoins globalement sur l'ensemble de la topologie et construire, en ayant une certaine connaissance de l'environnement, les routes adéquates.

Choix d'un routage par gradient La construction de la topologie de routage (l'ensemble des chemins) peut prendre en compte des critères locaux (le routage choisit le prochain saut de chaque nœud en fonction de son voisinage, e.g. meilleur *PER* vers un voisin). Un routage prenant en compte des critères globaux (e.g. voisin possédant la meilleure route vers un point de sortie en termes charge maximale supportée par les liens la composant) est plus performant car il évite des choix locaux non pertinents (e.g. un lien de bonne qualité vers un voisin isolé). L'algorithme de *Bellman-Ford* fonctionne ainsi sur critères de route.

Étant donné que le trafic de données converge en majorité vers un ensemble restreint de points (trafic *multipoint-à-point (MP2P)*), les réseaux de l'*IoT* motivent la mise en place d'un routage par gradient [**?**] : on associe à chaque nœud un rang qui représente une distance (physique ou logique) au point de sortie le plus proche, représentant par exemple le nombre de sauts nécessaires pour atteindre une passerelle, la qualité des liens, la charge de trafic, ou l'énergie résiduelle des nœuds [**?**].

Le choix des chemins se fait donc *a priori* à partir d'une combinaison de ces critères. Pour chaque trafic, un critère de choix de route différent peut être appliqué, par exemple :

1. pour les flux ayant de fortes exigences de taux de livraison, le routage choisit les liens à *PER* faible ;
2. pour les flux ayant de fortes exigences de délai, le routage choisit les liens les moins chargés et évite ainsi les congestions au niveau des nœuds intermédiaires (trames en attente en mémoire).

Collection Tree Protocol (CTP) [**?**] est un exemple de protocole de routage par gradient basé sur la métrique de *Nombre Attendu de Transmissions*, (ou *Expected Transmission Count (ETX)*) [**?**]. Cette métrique estime, le long d'un chemin, le nombre moyen de transmissions nécessaire à la livraison d'une trame. L'*ETX* est donc directement liée au *PER* de chacun des liens traversés. *CTP* favorise donc les routes les plus fiables en termes de qualité des liens. De même que pour les gradients géographiques, *CTP* concentre le trafic sur les meilleures routes sans prendre en compte d'autres paramètres (répartition de charges, délais).

Le groupe *ROLL (Routing Over Low-power and Lossy networks)* a proposé le protocole de routage distribué, *RPL* [**?**]. *RPL* est un protocole à vecteur de distance pour les *LLN*, qui construit des graphes de routage orientés et acycliques (*Destination-Oriented DAG*, *DODAG*) reliant l'ensemble des nœuds en multi-chemin à un des points de collecte du réseau. *RPL* reprend et approfondit les idées de *CTP*. La mise en place d'un routage par gradient y est initiée à chaque point de collecte (une passerelle, nommée aussi *DAG root*) et fonctionne par diffusions successives de messages (les *DODAG Information Object (DIO)*) annonçant les routes construites suivant la métrique choisie et les rangs des nœuds calculés localement. Chaque nœud peut ainsi choisir ses parents préférés, de rangs inférieurs, ordonnés suivant la métrique de route, formant de fait un *DODAG* multi-chemin. De manière à pouvoir créer une route inverse, chaque nœud pourra ensuite annoncer son identifiant à ses parents via un message (*Destination Advertisement Object (DAO)*) remonté jusqu'au *DAG root*.

RPL peut former une route pour chaque flux considéré et ses exigences, ou pour pallier un changement de la connectivité (choix de parents secondaires, à utiliser en cas de défaillance du lien préféré). Ainsi, plusieurs routes coexistent au niveau d'un même nœud. Le fait de sélectionner plusieurs parents possibles, couplé au choix du gradient, aboutit à la formation des *DODAG* dirigés vers les points de sortie [**?**]. Pour chaque application, un *DODAG* peut être construit dans le but de respecter un ensemble d'exigences. La topologie multi-chemin résultante répond aux exigences de chaque application : plusieurs instances de routage peuvent cohabiter sur le même réseau et créer un *DODAG* différent, enraciné à chaque passerelle, pour chaque application.

Pour établir une première connectivité sur un réseau *6TiSCH*, le groupe propose une configuration minimale [**?**], où le routage se met en place de manière distribuée avec *RPL* [**?**]. Au démarrage du réseau (avant tout autre contrôle de la topologie), *RPL*, par la collecte locale d'informations (choix distribué), permet de mettre en place une première connectivité à bas coût en ressources. Cependant, dans notre cas multi-applicatif, les exigences sont parfois antagonistes : la superposition de chemins choisis sur critères distincts peut entraîner une saturation des ressources, ou l'impossibilité de répondre aux exigences de tous. Nous préférons donc une construction centralisée du routage. Au niveau de chaque nœud, les choix d'ordonnancement de trames et des transmissions reviennent à la couche d'accès au médium. L'opérateur doit choisir les applications à traiter en priorité, d'une part, et il doit prendre la répartition des trafics comme critère dans les métriques de routage, d'autre part. L'équilibrage de charge permet en effet une consommation plus homogène de ressources

et prolonge donc la durée de vie du réseau. Nous proposons une réponse à ces besoins de l'opérateur au Chapitre 6.

2.6 Mécanismes de gestion des réseaux de l'*IoT* multi-sauts opérés

Le maintien de la qualité de service dépend de la surpervision et du contrôle mis en place par l'opérateur sur son réseau [**?**]. La notion de *Software Defined Networking* (*SDN*) structure l'étude de la gestion des réseaux [**?**] notamment sans fil. En particulier, les *SDN* permettent de définir une architecture logicielle de gestion de réseau pour l'*IoT* [**?**], de manière efficace en termes de surcoût de contrôle [**?**].

2.6.1 Garanties de *QoS* par flux applicatif

La traduction des exigences de qualité de service d'une application donnée en termes de paramètres de gestion de réseau est difficile : en effet, les trames provenant de nœuds différents ne rencontreront pas le long de leurs chemins multi-sauts les mêmes difficultés d'acheminement (congestion, pertes de trames). Pour la même exigence de *QoS*, la traduction en termes de ressources nécessaires est donc différente pour chaque source de trafic.

Les *SDN* favorisent la gestion de trafic par flux, sur les réseaux en général et par extension pour les réseaux radio multi-sauts [**?**]. *OpenFlow* [**?**], qui est le représentant populaire et industriel des *SDN* [**?**], laisse la définition de flux assez libre suivant les mises en œuvre. La spécification des flux fait l'objet d'études sur les réseaux *IP* [**?**] et de standardisation [**?**]. Dans notre cas, un flux applicatif correspond à :

1. un nœud source ;
2. un nœud de destination ou une direction (e.g. un ensemble de points de sortie potentiels dans le cas d'un réseau comprenant plusieurs passerelles) ;
3. une caractérisation temporelle de génération de trafic (nombre et taille de messages générés par périodes et échelles de temps) ;
4. un ensemble d'exigences de *QoS* exprimées de bout-en-bout, de la source à la destination (e.g. délai maximum de bout-en-bout des messages du flux).

Ainsi, les exigences d'une application peuvent être exprimées en exigences par flux (e.g. une contrainte de délai sur une application de supervision de la pollution se traduit en autant de contraintes de délai par source). Le trafic de chaque application est multi-flux, initié à chaque source. Cela permet de prendre en compte dans la gestion les exigences extérieures au réseau (e.g. signature d'un nouvel *SLA*) et endogènes (évolution des propriétés des nœuds et des liens) qui ont été largement étudiées [**?**].

Il faut donc coordonner les mécanismes de réseaux (e.g. le choix des routes par flux [**?**]) à la gestion des applications et du réseau. C'est un des objectifs couverts au Chapitre 4.

2.6.2 Architecture de gestion des réseaux de l'*IoT*

Du fait de leur diversité, le développement des réseaux de l'*IoT* implique d'en abstraire les fonctionnalités de gestion [**?**] (gestion commune de l'ensemble du réseau quelles que soient les technologies sous jacentes). En effet, l'abstraction favorise l'interopérabilité entre les différentes technologies.

L'architecture abstraite promue dans l'approche *SDN* distingue deux couches fonctionnelles [?] :

1. le plan de données, qui comprend les mécanismes d'acheminement du trafic (e.g. le routage d'une trame sur un nœud) ;
2. le plan de contrôle, qui concerne la prise de décision de gestion de trafic (e.g. l'installation des règles de routage sur les nœuds). Le plan de contrôle comprend les mécanismes d'observation (ou *monitoring*), d'analyse et de configuration du réseau.

Nous avons étudié le plan de données pour les réseaux de l'*IoT* dans les sections précédentes. Nous décrivons ici les trois aspects du plan de contrôle qui caractérisent la gestion de réseaux pour l'*IoT*.

Observer le réseau L'observation du réseau consiste à collecter des informations sur les performances des nœuds et liens et à détecter d'éventuelles défaillances (perte de connectivité, congestion, etc.). Dans les réseaux de l'*IoT*, la difficulté est d'échanger les informations d'observation qui sont distribuées sur l'ensemble des nœuds en limitant le surcoût en communication et en mémoire [?], et donc en limitant l'impact sur le plan de données. Nous détaillons au Chapitre 5 les solutions existantes et notre contribution sur le sujet.

Analyser le comportement du réseau L'analyse consiste à vérifier à partir des observations le respect des exigences des applications et le bon fonctionnement du réseau (e.g. en termes de consommation d'énergie). L'analyse permet aussi de prévoir les changements nécessaires dans la configuration du réseau (e.g. calculer de nouvelles routes et allocations de ressources par flux). L'*IoT* impose par le nombre de nœuds et l'hétérogénéité des scénarios considérés, et par la nature variable de l'environnement radio, une analyse complexe (nécessitant une importante capacité de calcul sur de nombreux paramètres) et fréquente (adaptée aux variations du réseau) [?]. La prise en compte de l'ensemble des paramètres afin de garantir globalement la *QoS* pour chaque flux, rend donc plus facile une centralisation de l'analyse en dehors du réseau (sur un entité centrale appelée *SDN controller* dans le contexte des *SDN*, *PCE* dans le contexte de l'*IETF*, ou *SLA Manager* au Chapitre 4).

Adapter le comportement du réseau La configuration consiste à adapter le fonctionnement du plan de données en fonction de l'analyse (e.g. en modifiant les règles de routage ou d'allocation de ressources pour chaque flux). La diffusion sur le réseau des informations de contrôle souffre des contraintes de l'*IoT* (fiabilité, délai, occupation de ressources). En effet, l'information de contrôle doit transiter sur le réseau afin d'atteindre le ou les nœuds concernés. Puis chaque nœud doit modifier son comportement (e.g. appliquer un nouvel échéancier). C'est le rôle du *SLA Enforcer* au Chapitre 4.

2.6.3 Exemple : la gestion de réseau dans la pile protocolaire *6TiSCH*

Le groupe de travail *6TiSCH* de l'*IETF* travaille à organiser les communications radio pour les réseaux de capteurs contraints. Suivi par les opérateurs et les autres acteurs majeurs du domaine, *6TiSCH* fait le lien entre *IEEE Std 802.15.4-2015 mode TSCH*, la couche d'accès au médium, pour réseaux de capteurs, que nous avons décrite, et les protocoles de l'*IETF* basés sur *IPv6* [?]. *6TiSCH* intègre les efforts déjà réalisés pour les *LLN* (e.g. l'utilisation de *RPL* en couche réseau) [?].

La gestion des réseaux classiques [?] est également facilitée par *IPv6* dans les contextes industriels [?] et de l'*IoT* [?].

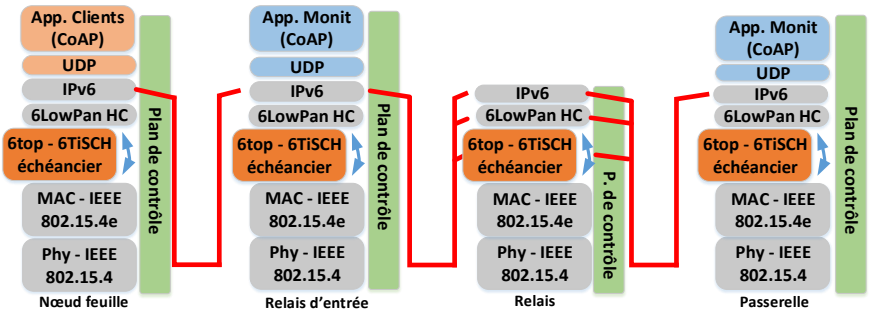


FIGURE 2.9 – La pile protocolaire proposée par *6TiSCH*

La pile proposée par le groupe (Fig. 2.9) permet d'isoler les flux de trafic applicatifs issus de capteurs, et en même temps d'optimiser et de rendre plus fiable l'utilisation du canal radio, en s'appuyant sur *IEEE Std 802.15.4-2015 mode TSCH*.

6TiSCH s'interface avec les travaux des autres groupes de travail de l'*IETF* :

1. *ROLL*, pour les tâches de routage distribué, avec *RPL* [**?**] ;
2. *Core*, pour le format d'échange de données applicatives avec *CoAP* [**?**] ;
3. *DetNet* [**?**], pour la construction de réseaux hétérogènes contraints déterministes , notamment le lien avec la construction externe de routes (avec d'autres entités, e.g. un *PCE* [**?**]) ;
4. *6lo*, pour l'adaptation des paquets *IP* aux trames pouvant circuler sur les réseaux contraints (*6TiSCH* réutilise *6LoWPAN* [**?**] afin de réduire le surcoût en termes de communications lié aux en-têtes *IP*, et d'adapter la taille des paquets transmis au standard *MAC*).

6TiSCH est une des piles protocolaires utilisées dans les mises en œuvre récentes en multi-saut. Elle est adaptée en code source ouvert sur une dizaine de types de nœuds [**?**] et en particulier sur les nœuds de la plate-forme de tests *FIT/IoT-LAB* [**?**]. Ainsi, les propositions de *6TiSCH* sont pertinentes pour l'opérateur pour réseaux de l'*IoT* multi-sauts.

L'ordonnancement par flux : la notion de *track* dans *6TiSCH* *6TiSCH* précise la notion de *track* : un *track* est un ensemble de blocs temps-fréquences le long d'une ou plusieurs routes, qui sont dédiés à certaines trames seulement [**?**]. Sur chaque nœud, les trames sont transmises selon l'identifiant de leur *track*. Ainsi, les cellules associées sont clairement identifiées. On peut donc associer un *track* à chaque flux applicatif et ainsi permettre la gestion de réseau par flux. Les ressources allouées à chaque flux sont alors clairement identifiées par chaque nœud, ce qui facilite l'ingénierie de trafic, par analogie avec *MPLS* : du point de vue du plan de données, l'identifiant de chaque *track* correspond à une étiquette permettant de choisir les cellules utilisées vers le prochain saut [**?**].

La Fig. 2.10 illustre la mise en place d'un *track* sur un chemin composé de trois liens. Les vues locales à chaque lien de l'échéancier (en haut de la figure) sont agrégées en une vue par *track* (en bas). À noter, une vue globale par *track*, en multi-chemin, peut éventuellement comporter des cellules superposées. En effet, il est possible d'utiliser la même cellule temps-fréquence pour les communications sur deux liens distants du même réseau (hors de portée radio). Par exemple, les liens $C \rightarrow D$ et $I \rightarrow H$ dans la Fig. 2.4 présentant peu de risques d'interférences, l'opérateur peut choisir de planifier une même cellule (même intervalle de

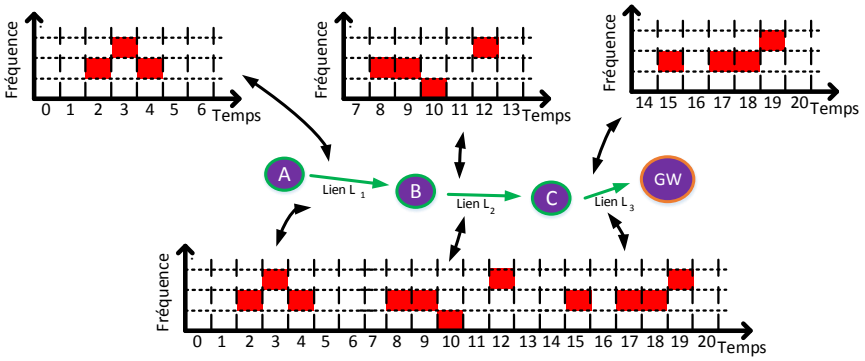


FIGURE 2.10 – Composition d'un *track* le long d'un chemin dans *6TiSCH*

temps, même *ASN*) pour leurs émissions. Cette réutilisation spatiale de fréquences optimise l'usage de la bande passante.

6TiSCH favorise la mise en place de *QoS* différenciée sur chaque cellule, notamment en donnant à chaque trame ou flux une certaine priorité dans la mémoire partagée des nœuds. L'allocation de cellules permet donc une gestion des flux en isolation (choix du *track* et priorité des trames).

Dans l'organisation des ressources de communication, l'ordonnancement doit donc trouver un compromis entre l'occupation de la bande passante, le respect des contraintes des clients et la longévité de la solution.

Du fait de l'isolation des flux par *track* et de la modélisation des ressources, l'opérateur peut détecter des contraintes de *QoS* antagonistes entre plusieurs flux. Cela facilite l'anticipation des problèmes et leur résolution (e.g. placer un relais supplémentaire, ou refuser les conditions d'un client).

La gestion locale de l'ordonnancement avec *6TiSCH Operation Sublayer* (*6top*)
Chaque nœud comprend une fonction d'ordonnancement, définie dans la sous-couche protocolaire de gestion de réseau, *6top* [**?**], qui fait partie de son plan de contrôle [**?**]. Elle lui permet de gérer ses allocations au niveau de la couche *MAC* (commandes adressées aux nœuds voisins, réception et application d'échéanciers, comptage et statistiques de performances, etc.). Cette fonction contient les mécanismes de contrôle de l'échéancier et d'échanges nécessaires à la collecte d'information et à l'ordonnancement.

6top rend possible à la fois l'ordonnancement distribué et l'ordonnancement centralisé. Le groupe de travail étudie la manière de définir les interfaces avec *TSCH* d'une part (remontées d'informations locales, notifications de changements d'échéancier), et des autres protocoles de couches supérieures d'autre part :

1. *RPL* pour les données permettant les calculs de routage, l'agrégation de métriques ;
2. des données de contrôle provenant d'autres entités (e.g. des consignes d'ordonnancement en *PCEP* ou *CoAP*).

Par exemple, *6top* permet à un nœud d'utiliser la qualité d'un lien avec un de ses voisins pour adapter son échéancier (le contrôle est alors distribué). Dans le cas d'un contrôle centralisé, la sous-couche *6top* est en charge de la mise en place de nouveaux échéanciers sur les nœuds, selon les prérequis reçus de l'entité de contrôle. Dans notre cas, l'opérateur doit construire une route pour chaque flux et allouer les cellules sur chaque lien de la route afin de respecter les *KPI*.

6top définit des commandes et un modèle générique en utilisant le langage de modélisation *YANG* [**?**]. Ce modèle donne accès aux métriques (e.g. l'état d'une cellule), à la configuration de *TSCH* et aux procédures de contrôle, ainsi que le support pour les différents mécanismes d'allocation. Le modèle sera aussi utile pour les définitions de métriques d'observation du réseau. Le groupe travaille actuellement à la spécification du protocole d'échange, *6P*, correspondant à l'interface de *6top* [**?**].

Stratégie de planification dans *6TiSCH*

6TiSCH prévoit la mise en place proactive ou réactive de l'ordonnancement. Pour la planification réactive de ressources, il est possible d'organiser l'acheminement des trames à la volée, selon des demandes dynamiques de trafic initiées par un nœud source de trafic. *Scheduling Function Zero (SF0)* se positionne dans ce scénario, en suivant la même logique que *RSVP*, en réservant des ressources de communication directement le long d'un chemin en le parcourant [**?**]. Un acquittement de bout-en-bout valide la réservation de ressources, et lorsque celui-ci revient à la source de trafic, les trames peuvent être directement envoyées. De même que pour l'accès à contention, ce scénario de planification réactive ne prend pas en compte la connaissance *a priori* du trafic que l'opérateur obtient des *SLA*.

Si les stratégies de routage doivent être appliquées avant la construction des échéanciers, il est possible de rassembler les deux fonctions de contrôle (choix du routage et ordonnancement). En effet, le choix des blocs temps-fréquences est directement lié au choix du prochain saut. Il n'est donc pas nécessaire de dé-corréler les deux couches protocolaires au niveau de chacun des nœuds : dans *6TiSCH*, la couche *6top* peut assumer à la fois les choix de routage avec les choix de blocs temps-fréquences. Dans ce cas (présenté sur le nœud relais en Fig. 2.9), la commutation des trames n'est pas prise en charge par les couches supérieures (e.g. *IPv6*) mais directement dans *6top*. Cette problématique, résumée sous la notion de *Route-over vs Mesh-under*, a un impact sur le coût en ressources de communication (la stratégie influe sur les en-têtes des trames), sur la complexité du code embarqué sur les nœuds et sur la flexibilité du problème.

En faisant l'hypothèse d'un ordonnancement statique, pré-configuré sur les nœuds, le groupe a abouti à la rédaction de règles d'une configuration minimale du réseau permettant son démarrage et l'intégration de nouveaux nœuds en cours d'utilisation [**?**].

Échange d'informations de contrôle dans *6TiSCH*

6TiSCH prône l'utilisation de *CoAP* (Constrained Application Protocol) [**?**] comme protocole de transfert de données applicatives pour les *LLN*. *CoAP* est une solution efficace pour l'échange de données de contrôle parce que le protocole s'inscrit dans une architecture *REST (Representational State Transfer)*, tout en prenant en compte les contraintes des nœuds (e.g. réduction du code embarqué sur les nœuds). En effet, il inclut :

1. des messages *GET* permettant de formuler une requête ;
2. des messages *POST*, permettant de reporter régulièrement des données nouvelles ;
3. une stratégie définie dans *Observe* [**?**], qui permet de notifier directement d'un changement d'un paramètre applicatif (e.g. sur un événement).

De plus, *CoAP* spécifie un format de données pour les applications qui permet leur intégration à la pile protocolaire *6TiSCH* en réduisant le coût en termes de ressources de communication (taille des messages). De fait, *CoAP* repose sur *UDP* et ainsi permet à une diversité d'applications d'être transportées le long du réseau, en permettant l'identification des données de chacune. Une variété de trafic peut ainsi être transportée en différenciant les flux et donc la *QoS* qui leur est offerte.

CoAP étant mis en place au dessus d'*IPv6*, l'opérateur peut transporter classiquement les données de ses clients. *CoAP* peut également être utilisé pour les échanges d'une application d'observation du réseau (e.g. sur un relais routeur d'entrée en Fig. 2.9). En effet, c'est le protocole actuellement utilisé pour transporter les informations formatées de gestion des nœuds (i.e. le *Device Management*), comme *COMI* [**?**] ou *OMA Ligthweight M2M* [**?**]. Le code embarqué peut donc être réemployé efficacement. *CoAP*, parmi les protocoles d'échange de données, est plus compact qu'*HTTP* [**?**], même si son efficacité doit encore être améliorée [**?**].

2.7 Synthèse du chapitre - choix de topologie, réseau et ordonnancement

Nous avons discuté le choix d'une topologie multi-saut. Celui-ci offre une grande latitude dans la configuration des paramètres du réseau en multipliant le nombre de relais et donc de chemins multi-sauts fiables possibles. Les leviers pour exercer le rôle d'opérateur y sont étudiés.

Dans ce contexte applicatif et ces contraintes liées à l'environnement radio, la difficulté pour un opérateur réside donc dans l'organisation des communications radio multi-sauts. Notre étude met en avant trois axes pour parvenir à garantir de la *QoS* sur réseaux radio multi-sauts :

1. en tirant profit des protocoles de niveau *MAC* à planification proactive : comment réguler les communications pour chaque lien ?
2. en utilisant la couche réseau : comment établir les meilleures routes entre nœuds éloignés, en termes de *QoS* et de durée de vie du réseau ?
3. en faisant le lien entre la couche réseau et les applications : comment coordonner la génération de trafic avec le routage et l'ordonnancement ?

Nous avons montré en détail comment le cadre de *IEEE Std 802.15.4-2015 mode TSCH* et les efforts du groupe de travail *6TiSCH* répondent à ces attentes. Une telle utilisation du médium doit être contrôlée pour l'ensemble du réseau. *6TiSCH* étudie en particulier l'observation du réseau, la construction d'échéanciers de communication entre les nœuds, et les procédures de configuration et de démarrage du réseau.

L'ancrage des travaux dans *IP* est avantageux pour l'opérateur parce que les mécanismes standards et les protocoles s'appuyant sur *IP* peuvent être adaptés à ses besoins (e.g. le transport des données applicatives sur *UDP*, l'adressage, *ICMP*, etc.).

En se basant sur *IEEE Std 802.15.4-2015 mode TSCH*, *6TiSCH* construit le successeur en standard ouvert de *WirelessHART* [**?**], et *ISA100.11a* [**?**] pour les technologies radio déterministes réunies en normes propriétaires et prévues pour des applications industrielles. Le standard est plus récent, ne dépend d'aucune application en particulier, et est plus flexible que ses prédécesseurs [**?**].

6TiSCH est également adéquat pour implémenter des *SLA* parce que *TSCH* permet de prédire la charge de trafic maximale supportée par les nœuds, de manière déterministe [**?**], et donc de prévoir la consommation d'énergie et de ressources. Et les *SLA* sont généralement construits à partir de métriques issues des réseaux *IP* [**?**].

Ce déterminisme est en cours d'étude par *6TiSCH* en collaboration avec le groupe de travail *IETF Deterministic Networking (DetNet)* [**?**], qui prend lui en compte une plus grande hétérogénéité des nœuds et donc les mécanismes nécessaires pour parvenir à l'interopérabilité. En effet, l'usage de bandes de fréquences sans licences impose de rendre plus

robustes et prévisibles les technologies radio existantes à tous les niveaux des protocoles radio.

Le cadre initial de travail (jusqu'à ce que la nouvelle charte soit approuvée en février 2016) se limitait à un routage distribué et un échéancier statique, par exemple pré-configuré sur chaque nœud. Ce cadre de départ permet de tester le fonctionnement du réseau et d'avancer la standardisation. Cependant, cela ne suffit pas à l'opérateur, qui requiert une mise à jour possible des configurations des nœuds.

L'opérateur peut d'ores et déjà tirer parti des règles d'une configuration minimale [**?**] en initiant son réseau (avant la mise en œuvre des premiers *SLA*) et en collectant les informations sur la qualité des liens, réalisant ainsi une premiere observation des métriques et permettant une analyse initiale des performances.

Cette configuration minimale donne, par défaut, un nombre fixe d'intervalles avec contention à l'accès à l'ensemble des nœuds. Cela ne répond ni à une demande de trafic importante, ni à une exigence de garantie de service.

Le fait de pouvoir modifier un échéancier reste une condition nécessaire à l'opérateur car cela permet d'adapter la configuration à de nouveaux trafics et à de nouvelles conditions radio. La nouvelle charte [**?**] oriente les futurs travaux vers un ordonnancement dynamique distribué, et centralisé, en proposant le rapprochement avec *DetNet* et *PCE*. Ce serait suffisant pour mettre en place les échéanciers de communication, construits depuis le système d'information de l'opérateur.

Les prochains travaux devront donc permettre à l'opérateur de mettre en place les échéanciers adaptés aux contraintes applicatives. De plus, *6TiSCH* prévoit un ordre de priorités et l'isolation des trafics par la mise en place d'ensembles dédiés de cellules, les *tracks*.

2.7.1 Opérer les réseaux de l'*IoT* multi-sauts

Selon notre point de vue, la technologie *IEEE Std 802.15.4-2015 mode TSCH* est un bon support pour implémenter une solution d'opérateurs avec des contraintes de *SLA* pour l'*IoT*. En effet, le standard est prévu pour prendre en compte la *QoS* exigée par les différentes applications existantes. Nous supposons que le déploiement d'applications urbaines obligera les opérateurs à utiliser des solutions de ce type, déterministes, pévues pour des réseaux de capteurs contraints et exigeants, adaptés aux besoins de fiabilité dans l'industrie, car ces solutions permettent aussi de mettre en place des garanties multi-services.

La pile protocolaire proposée par *6TiSCH* permet d'offrir des garanties de *QoS* variées grâce au contrôle des ressources en temps et en fréquence. En effet l'approche *FTDMA* permet à l'opérateur de contrôler finement l'allocation de ressources. On peut établir la relation entre les métriques de haut niveau (bande passante, délais) et les paramètres du réseau (e.g. nombre de cellules allouées). L'opérateur sait déterminer la quantité de ressources nécessaire au respect des *KPI*, qui est stable et prévisible contrairement au cas de l'utilisation de mécanismes de contention à l'accès au médium.

6TiSCH inclut aussi des mécanismes d'observation locaux à chaque cellule temps-fréquence et pour chaque nœud, permettant de construire des statistiques de performances. Ceci doit être utilisé pour construire les *KPI* de l'opérateur.

L'architecture fonctionnelle de *6TiSCH* laisse l'opérateur libre de son plan de contrôle : il peut à la fois collecter les informations dont il a besoin par le biais de protocoles distribués (*RPL*), et faire fonctionner une entité de routage et d'ordonnancement externe selon ses propres algorithmes et intérêts (e.g. l'équilibrage de charge, qui permet de distribuer la consommation de ressources tout en réduisant le risque de pertes d'information liés aux goulets d'étranglement).

De plus, avec les *tracks* proposés dans *6TiSCH*, l'opérateur a une vision long-terme de la santé de son architecture.

Plusieurs types de changements peuvent intervenir sur le réseau :

1. certains trafics peuvent apparaître ou disparaître : un client peut modifier les données qu'il souhaite collecter ;
2. certains nœuds peuvent apparaître ou disparaître : la raison peut en être une perte totale d'énergie, une panne, ou de nouveaux déploiements ;
3. les contraintes applicatives peuvent changer (relâchement ou renforcement des exigences des clients) ;
4. l'environnement radio est amené à changer : de nouvelles constructions limitent une portée, de nouvelles émissions perturbent le canal, les nœuds sont déplacés, etc.

L'opérateur doit pallier ces problèmes et faire évoluer l'architecture. Dans le contexte de *6TiSCH*, l'opérateur peut détecter un problème de performance grâce aux statistiques locales fournies par la sous-couche *6top*. On peut envisager de maintenir actif un protocole de routage (e.g. *RPL*) dans le but d'obtenir et de construire des métriques supplémentaires, favorisant l'observation du réseau. Le premier défi consiste à faire en sorte que cette information distribuée soit reçue de manière centralisée par l'opérateur, tout en ne consommant pas trop de ressources. Nous proposons au Chapitre 5 d'apposer ces informations sur les trames de données afin qu'elles circulent de bout-en-bout.

Muni de ces informations, régulières, l'opérateur peut reconstruire périodiquement un nouvel échéancier *FTDMA*, adapté aux nouvelles conditions radio. De manière centralisée, le *Path Computation Element* (*PCE*) [**?**] peut calculer les allocations nécessaires, *track* par *track*, pour des nouveaux flux à installer dans les échéanciers des nœuds. De plus, l'allocation peut se faire en répartissant la charge sur le réseau, de manière à retarder la perte définitive du premier nœud, liée à une surconsommation de ressources de communication. Nous le proposons par la suite au Chapitre 6.

Ce sont ces mécanismes de contrôle (observation, allocation de ressources) qui permettent de garantir les *KPI*, dans le cadre des *SLA*. Les caractéristiques des applications influencent le choix des bons algorithmes, protocoles et paramètres de communication. Mais aussi précis que soit le contrôle, il est difficile de donner des garanties *a priori* sur la qualité de service offerte à chaque application dans un environnement trop peu stable. En effet, si le médium radio évolue (et il le fait d'autant plus sur les bandes non licenciées) plus fréquemment que la période de ré-ordonnancement, les garanties apportées risquent de ne pas être respectées. *DetNet* propose d'étudier la question de la gestion de multiples applications avec différentes contraintes de *QoS*, sur environnement peu fiable et pour une architecture hétérogène.

Dans le prochain chapitre, nous montrons comment spécifier de manière non équivoque les flux et la *QoS* attendus, dans des *SLA* spécifiques qui répondent aux besoins des opérateurs pour réseaux de l'*IoT*. Cette spécification permet ensuite d'articuler une architecture de gestion de l'*IoT* multi-service pour un opérateur (Chapitre 4). Nous nous appuierons sur des mécanismes d'observation (Chapitre 5) et d'allocation de ressources (Chapitre 6).

Chapitre 3

Spécification des *SLA* pour les réseaux de l'*IoT*

Dans ce chapitre, nous proposons un modèle de spécification de contrats de *QoS* (les *SLA*) pour opérer l'Internet des Objets (*IoT*). En effet, chaque entreprise cliente a des besoins applicatifs spécifiques concernant la transmission de messages entre ses capteurs et son système d'information.

L'opérateur, tout en mutualisant son infrastructure radio, doit respecter ces exigences de Qualité de Service (*QoS*) spécifiées dans un contrat : le *SLA*. Dans le domaine des *web services*, les *SLA* ont fait l'objet d'études multiples, et il en a été donné une spécification en *XML* : il s'agit du modèle *WSLA* [**?**].

Nous étendons cette spécification en y intégrant des éléments répondant aux contraintes des réseaux de l'*IoT*. En particulier, nous donnons la possibilité de respecter la *QoS* pour des ensembles particuliers de capteurs (e.g. sur une zone donnée). Nous permettons la construction de métriques complexes qui rendent compte de la performance du réseau.

3.1 Introduction

Nous avons introduit au Chapitre 2 le rôle d'opérateur de réseau de l'*IoT*. Nous évitons ainsi de réaliser un déploiement dédié pour chaque application, et d'en faire la maintenance séparément comme c'est le cas aujourd'hui. Les réseaux multi-sauts sans fil mutualisés, gérés par un opérateur, permettent d'opérer un nombre croissant de flux applicatifs depuis les capteurs et jusqu'aux systèmes d'information (SI) des clients et vice-versa. La gestion de ces réseaux est un point clé qui en détermine la performance. En particulier, dans le contexte des *Smart Cities* [**?**], les canaux de communication radio doivent être partagés spatialement et temporellement parmi les nœuds comprenant les capteurs mesurant divers éléments : la température, la présence de voitures, déclenchant des alarmes sur le niveau de pollution ou remontant des index de consommation de gaz, etc.

La mutualisation de l'infrastructure est un point clé de la réduction de coûts : un seul déploiement est nécessaire. L'opérateur acquiert une vision d'ensemble de son réseau :

1. la maintenance est plus aisée et passe mieux à l'échelle pour une large couverture (la mutualisation permet de gérer plusieurs applications à la fois) ;
2. les ressources de communication sont plus efficacement utilisées en étant partagées entre usagers, réduisant les coûts car le système est mieux dimensionné (moins de gaspillage de ressources) ;

3. l'opérateur permet aux nœuds d'accéder de manière structurée au médium, au lieu de les laisser en concurrence et en interférences.

L'opérateur se charge donc de la gestion des communications radio, en laissant aux clients leur expertise sur les choix de fonctionnement des applications (e.g. contenu et fréquence des messages générés).

L'opérateur fiabilise l'*IoT*, pour les clients qui l'utilisent, en fournissant des garanties sur la qualité de service (*QoS*) de chaque flux. L'opérateur doit satisfaire les contraintes des applications clientes en termes de délais et de taux de livraison. Il doit être capable de prouver que les conditions contractuelles sont respectées.

Par exemple, une compagnie de gaz souhaitant collecter des index de compteurs de gaz peut utiliser l'infrastructure radio de l'opérateur. L'application cliente requiert qu'au moins un message (parmi les deux qui sont produits chaque jour) soit collecté, et ce pour 95% des compteurs des usagers. En outre, le client souhaite que ses messages soient collectés en moins d'une heure, afin de répondre à des exigences de facturation. L'opérateur doit démontrer que sa solution fonctionne, jour après jour. Il doit négocier les conditions dans lesquelles il donne ses engagements : position des nœuds, nombre de messages applicatifs, etc.

Pour ce faire, nous proposons une architecture de gestion de réseaux de l'*IoT* opérés au chapitre suivant. Plusieurs mécanismes permettent de différencier la *QoS* dans les réseaux de capteurs sans fil [**?**]. Cependant, la nouveauté de notre contribution réside dans le lien que nous proposons entre les exigences des applications des clients et le système de gestion de réseau. L'architecture que nous proposons [**?**] donne la possibilité aux clients de négocier leurs exigences dans les *SLA*. Elle permet également à l'opérateur de réseaux de l'*IoT* de gérer les performances de son infrastructure de réseau et d'informer les clients du respect des clauses des *SLA*.

Dans ce chapitre, nous proposons une méthode de construction de *SLA* pour l'*IoT*. Nous nous appuyons sur une spécification au format *XML* qui permet à un client et à un opérateur de négocier les contenus possibles des *SLA*. Le schéma *XML* proposé intègre les contraintes et les paramètres requis pour opérer les réseaux de l'*IoT* multi-applicatifs.

3.1.1 Énoncé du problème

Les *Service Level Agreements* (*SLA*) sont des contrats de *QoS* entre un fournisseur de services, un client et de potentiels acteurs tiers. Ils précisent la façon dont le service est proposé et les obligations auxquelles chaque acteur doit répondre. Ils ont été utilisés pendant longtemps, dans plusieurs domaines informatiques et en particulier sur *IP* [**?**]. Par exemple, ils ont été mis en place pour le service de téléphonie, puis généralisés aux services de voix sur *IP*, afin de spécifier la qualité de la transmission vocale, ou la quantité d'appels simultanés possibles. Les relations contractuelles entre les opérateurs (i.e. les fournisseurs de services de télécommunications) et les clients sont régies par ces *SLA*.

Souvent, les exigences applicatives pour la collecte des données sur les réseaux de l'*IoT* n'ont pas de traductions concrètes dans la configuration du réseau : elles sont vérifiées *a posteriori* en analysant les données collectées passivement (cf. le paradigme de *Big Data*). Au contraire, nous visons à fournir un cadre pour les réseaux de l'*IoT*, qui permette de répondre aux différentes exigences applicatives par la gestion de la couche d'accès au médium et la couche réseau. Le contenu des *SLA* doit être traduisible dans la mise en œuvre des technologies telles que *RPL* [**?**] ou *6TiSCH* [**?**].

Les services que nous décrivons ici concernent la transmission de flux applicatifs entre les capteurs des clients et leurs systèmes d'information. Notre objectif est de spécifier de

façon non-ambiguë et contractuelle les paramètres de bout-en-bout des couches de réseau, comme le taux de livraison des messages ou le délai de transmission. Les applications envisagées (e.g. la télérelève, la surveillance de la pollution, etc. [**?**]) dépendent de ces paramètres pour fonctionner correctement. Les clients doivent restreindre le trafic qu'ils génèrent suivant le profil de trafic maximal négocié avec l'opérateur dans le *SLA*.

Dans ce contexte, nous devons faire apparaître des informations spécifiques à l'*IoT* dans les *SLA*, qui constituent la base de la négociation entre l'opérateur et chacun de ses clients. Nous devons spécifier :

1. la quantité et la géolocalisation 3D des nœuds : l'opérateur doit vérifier la capacité de son infrastructure et la connectivité des nouveaux nœuds ;
2. chaque ensemble spécifique de nœuds (e.g. les nœuds d'un bâtiment, ou les nœuds incluant des capteurs de température d'une zone géographique, ou 2 nœuds tous les 50 mètres carrés dans une ville, etc.) sur lesquels s'appliquent les exigences du *SLA* ;
3. des opérateurs mathématiques spécifiques (e.g. la façon de calculer un taux de livraison pour une période donnée) ;
4. des critères de performances simples, univoques, fournis par l'opérateur ;
5. l'évaluation de performances sur des périodes glissantes ou à date fixe.

Tous ces éléments sont détaillés en Section 3.4.

3.1.2 Cas d'utilisation

Afin d'illustrer l'utilisation de *SLA*, nous donnons l'exemple d'une application de télérelève de compteurs de gaz. Le *SLA* présenté est le résultat de la négociation entre trois acteurs :

1. l'opérateur *WSNOperator*, en charge de l'infrastructure de réseau et de l'acheminement des messages ;
2. le client, une compagnie de gaz, nommé *GasCompany23*, qui possède des capteurs générant des messages liés aux usages de gaz ;
3. le prestataire *WSNMeasurements*, un acteur tiers pour vérifier la validité du *SLA* en cours.

Le *SLA* spécifie trois services différents pour cette application :

1. le service de collecte journalière d'index de consommation (pour lequel le principal *KPI* est le taux de livraison) ;
2. le service de configuration des capteurs ;
3. le service d'alarmes (pour lequel le principal *KPI* est le délai).

La collecte journalière Le *SLA* définit l'ensemble des nœuds concernés par le service, leur identifiant, leur position géographique. Le client précise le trafic attendu : nombre de messages générés sur chaque nœud source, par période de temps, ainsi que leur taille maximale.

Le *SLA* exprime les conditions dans lesquelles doit s'effectuer la collecte : l'opérateur garantit un taux minimum de livraison des messages (pour le trafic attendu) et des conditions sur le délai de transit des messages (e.g. délai moyen sur chaque période). Les conditions peuvent différer selon les jours et les heures (e.g. heures ouvrables, week-end).

Le *SLA* définit l'acteur chargé de démontrer le bon fonctionnement du réseau et le respect des garanties contractuelles. Cet acteur peut être l'opérateur ou une entité indépendante (ici *WSNMeasurements*). Le *SLA* précise la période de supervision des *KPI*.

Enfin, le *SLA* définit les actions à effectuer en cas de dégradation du service (garanties non respectées) et les pénalités.

Le service de configuration Il concerne les paramètres de *SLA* et les métriques qui correspondent à la transmission des messages depuis les systèmes d'information des clients vers leurs nœuds sources.

Le client peut en effet modifier le comportement de ses nœuds (e.g. fréquences d'émission, demande d'information complémentaire, changement de format des messages).

Le trafic est moins important et les garanties de délai et de livraison diffèrent. Le format de spécification reste identique.

Le client peut distinguer un ensemble de nœuds devant être mis à jour plus souvent que les autres. Le *SLA* permet de définir chaque service pour différents ensembles de nœuds.

Le service de remontée d'alarmes Les messages d'alarmes de gaz (e.g. détection d'une fuite ou d'un problème d'acheminement du gaz) ont des contraintes spécifiques : le trafic attendu est moins important, ponctuel, mais la collecte requiert de plus faibles délais.

Afin d'offrir des garanties, l'opérateur doit allouer des ressources en prévision d'une alarme potentielle. La définition du trafic d'alarme doit prendre en compte les exigences et les corrélations (e.g. plusieurs nœuds détectent la même rupture d'approvisionnement).

3.1.3 Intérêt de la mise en place de *SLA*

La diversité des acteurs proposant des solutions de connectivité pour l'*IoT* accroît les exigences attendues. Les *SLA* représentent un moyen de définir explicitement quels sont les services offerts par chaque acteur en spécifiant l'ensemble des paramètres possibles. Les contrats doivent être rédigés en termes non-ambigus. Ainsi, un client est en mesure de comparer les services proposés à travers les *SLA* et de choisir ce qui convient le mieux à ses besoins.

En suivant un modèle de *SLA*, les acteurs veillent à ce que le contrat qu'ils rédigent soit complet et cohérent. De plus, opérateur et clients sont en mesure de réutiliser les clauses de *SLA* existants et de les adapter à leurs besoins (e.g. les actions garanties en cas de dégradation de service).

En résumé, les *SLA* offrent de la clarté, de l'interopérabilité, de l'exhaustivité et de la flexibilité d'expression, ainsi que des exemples et des modèles que tout le monde peut utiliser.

Le reste du chapitre est organisé comme suit : nous détaillons les travaux antérieurs en section d'état de l'art (Section 3.2). Puis nous expliquons les outils de construction des *SLA* en Section 3.3. Nous explicitons les extensions nécessaires au modèle de *SLA* pour le cas de l'*IoT* en Section 3.4. Nous intégrons ces extensions dans la structure générale des *SLA* en Section 3.5 et nous concluons en Section 3.6.

3.2 État de l'art

Dans cette section, nous replaçons en premier lieu la notion de *SLA* dans la littérature (Section 3.2.1). Nous expliquons les mécanismes de *QoS* associés (Section 3.2.2). Puis nous introduisons la proposition de *WSLA* [**?**] (Section 3.2.3). Nous expliquons le contexte de la proposition et en quoi elle répond également à notre problématique. En Section 3.2.4, nous mettons en évidence les éléments spécifiques aux *web services*, et ce qui manque pour

les réseaux de l'*IoT*. En Section 3.4.1, nous détaillerons la composition des *SLA* construits avec *WSLA*.

3.2.1 Un aperçu des *Service Level Agreements (SLA)*

Quand un client contractualise des services à un fournisseur, (e.g. un opérateur de télécommunications), la Qualité de Service (*QoS*) doit être définie précisément afin que ceux-ci soient correctement réalisés. Les *Service Level Agreement* (*SLA*) définissent alors les engagements techniques et financiers des deux parties [**?**]. Ils définissent aussi les pénalités devant être acquittées par l'opérateur en cas de non respect des contraintes sur lesquelles il s'est engagé. Les opérateurs de réseau, lorsqu'ils signent des *SLA*, s'engagent à collecter ou à faire collecter par un prestataire des métriques de performance qui prouveront que la *QoS* est maintenue. Il existe plusieurs stratégies [**?**] :

1. des *SLA* statiques : l'opérateur ne modifie pas son système ; il prend un risque statistique de rupture du contrat dans le futur, à la manière des modèles de prédiction pour les compagnies d'assurance ;
2. des *SLA* réservés : l'opérateur dédie des ressources déterministes pour chaque *SLA* et garantit que celles ci seront toujours conformes aux caractéristiques requises ;
3. des *SLA* adaptatifs : l'opérateur est capable de modifier en temps réel la configuration du réseau. Il est capable de s'adapter aussi bien à des incidents qu'à des dégradations progressives de la performance du réseau. Les *SLA* sont dynamiquement maintenus.

Cette dernière approche implique que les *SLA* définissent une durée maximum de dégradation du service. Ce temps de *SLA* dégradé [**?**] doit être court : l'opérateur doit agir rapidement sur son infrastructure pour rétablir le niveau de performance normal.

Les *SLA* se décomposent en objectifs (les *Service Level Objectives* (*SLO*)), qui sont des sous parties du *SLA* correspondant à une exigence de *QoS* particulière. Par exemple, *Au moins un index quotidien de consommation doit être collecté chaque semaine pour 99% des compteurs de gaz*. Les *SLO* sont définis sur une trame temporelle, par exemple *chaque jour ouvré entre 7h et 20h*.

3.2.2 Mécanismes de *QoS* pour la gestion des *SLA*

Dans le but de respecter les exigences des *SLA*, l'opérateur peut utiliser les mécanismes suivants de *QoS*, en particulier sur une technologie sans fil [**?**] :

1. le contrôle d'admission ou *Call Admission Control (CAC)* : l'opérateur décide d'admettre ou non un message sur son réseau (il évalue le risque d'impact sur le reste du trafic) ;
2. le contrôle de ressources (e.g. dans les réseaux radio, le *Radio Resource Management (RRM)*) : il s'agit d'utiliser les couches d'accès et de routage afin de garantir un partage des ressources ;
3. l'observation de réseau ou *Network Monitoring* : c'est ce qui permet de superviser le réseau et de valider le respect des clauses du *SLA* ;
4. l'anticipation de *QoS* : l'analyse permettant de prévoir les changements du réseau ;
5. la gestion de configuration : l'opérateur contrôle son infrastructure en modifiant les paramètres.

Dans les réseaux radio, la propriété de diffusion médium radio requiert des protocoles évitant les collisions et limitant les interférences. En particulier dans les réseaux de l'IoT, on peut planifier les communications, selon des algorithmes comme *Traffic-Aware Scheduling Algorithm (TASA)* [**?**], qui construit des échéanciers temps-fréquences de communication, étant données la topologie des nœuds et la charge de trafic requise.

Dans les réseaux de capteurs, l'estimation de la qualité des liens est cruciale pour savoir si la *QoS* sur laquelle les acteurs se sont accordés est atteinte [**?**]. Elle permet avec d'autres mesures brutes, comme l'énergie résiduelle [**?**] ou les statistiques de trafic, d'élaborer les métriques complexes définies dans les *SLO*. Certaines de ces métriques démontrent que l'exigence de qualité de service de chaque *SLO* est remplie et guident les actions de contrôle en temps réel sur le réseau. D'autres sont définies en tant qu'indicateurs clés de performance (*KPI*) et sont envoyées aux clients et à l'opérateur pour leur permettre de construire des statistiques de réseau de haut niveau.

3.2.3 *SLA* pour les *web services* : le modèle *WSLA*

La notion de *SLA* a été relancée il y a dix ans avec l'apparition des *web services* et de l'architecture orientée services (SOA). En 2001-2003 , *Keller* et *Ludwig* ont présenté *WSLA*, un travail pionnier décrivant en *XML* la construction de *SLA* pour les *web services* [**?**]. Bien qu'il existe d'autres efforts de spécifications de *SLA*, celles-ci sont soit trop spécifiques à un domaine donné [**?**], et ne pourraient pas être adaptées au cas de l'IoT, soit trop génériques et moins applicables à des cas concrets d'utilisation [**?**]. On ne pourrait pas en proposer une mise en œuvre dans l'architecture du Chapitre 4.

Le schéma *XML* de la spécification *WSLA* définit tous les composants des *SLA*[1] [**?**]. De même, nous définissons pour notre scénario un ensemble de paramètres et conditions qui composent les contrats entre les clients (e.g. collectivités, fournisseurs de gaz, etc.) et l'opérateur, pour un service spécifique (e.g. la télérelève sur une zone d'une ville).

Keller et *Ludwig* ne font pas d'hypothèses sur la façon de construire ces documents, qui peuvent être le résultat d'une négociation. Les services peuvent aussi être définis au préalable par l'opérateur, alors que les valeurs et les seuils peuvent être laissés à la négociation.

Le modèle *WSLA* divise le *SLA* en 3 blocs :

1. les acteurs (*Parties*) comprenant les fournisseurs, les clients et les parties tierces de support ;
2. des métriques complexes, définies comme paramètres de *SLA* (*SLA Parameters*), construites à partir de mesures simples et des fonctions ;
3. les engagements (*Obligations*), comprenant les *SLO*, les notifications, les garanties, les intervalles de temps et les échéanciers à respecter.

En combinant les trois blocs dans un contrat, à l'aide d'une grammaire riche de métriques, de références temporelles et d'opérateurs mathématiques, l'opérateur est capable de proposer une gamme de *SLA* adaptés et précis.

L'interprétation du code *XML* pour le déploiement de la configuration du système selon un nouvel *SLA*, ne fait pas partie du cadre d'étude de *WSLA*. Des solutions propriétaires (e.g. logiciel privateur) sont à prévoir pour la négociation de *SLA*, la communication des *KPI* entre les systèmes d'information des clients et de l'opérateur.

1. Notez que les éléments de code issus de ce modèle de [**?**], sont spécifiés Copyright 2001, 2002, 2003 IBM Corp.

3.2.4 Limites du modèle *WSLA* pour les réseaux de l'*IoT* opérés

Dans une large mesure, le cadre *WSLA* répond au problème des réseaux de l'*IoT* opérés. En effet, le cadre de *WSLA* prévoit :

1. Une structure qui correspond à nos hypothèses sur les *SLA* ;
2. La possibilité d'écrire ses propres mesures simples et univoques (directives de mesure), fournies par le système d'information ;
3. L'évaluation de paramètres sur un support de séries temporelles (*time series*) ;
4. La définition de *garanties d'actions*, à l'interface entre le contrôle du système, et l'évaluation des *SLA* (e.g. `Si alarme > seuil, déclencher l'envoi de SMS.`).

Keller et *Ludwig* ont démontré que le modèle *WSLA* est extensible en définissant eux-même des éléments spécifiques à leurs domaines. Par exemple, ils ont écrit des extensions du type d'action et du type d'opération pour leur cas d'utilisation avec *WSDL* et *SOAP* [**?**].

Les fonctions mathématiques sont seulement nommées dans le document, et précisées et mises en œuvre séparément. Ce fait donne également la liberté de définir des fonctions qui conviennent à notre cas d'usage de l'*IoT* opéré. Par exemple, la fonction *RateOfChange* [**?**], définie dans le cadre de *WSLA*, correspond à un besoin spécifique dans les *web services*. Nous étendons le modèle de la même manière.

Une différence majeure entre les deux domaines réside dans la multiplicité des nœuds pour l'*IoT*. Les définitions de service varient selon les nœuds auxquels ils se réfèrent, à quel endroit et à quel moment. Dans le domaine des *web services*, le système d'information est facile à définir et à atteindre. Dans le domaine de l'*IoT*, les nœuds et leurs capacités doivent apparaître dans les *SLA*.

Nous proposons d'étendre la structure du *SLA*, donnée dans le cadre de *WSLA*, en y ajoutant un quatrième bloc, *Devices*, qui nous permet d'inclure les informations manquantes concernant les nœuds et la topologie de réseau. Cette extension permet de répondre à notre problème sans perdre la grande quantité de contributions qui conviennent à l'*IoT* opéré dans le modèle *WSLA*.

3.3 Outils de construction de *SLA*

Dans cette section nous détaillons les outils de spécification sur lesquels s'appuient le modèle *WSLA* et que nous utilisons dans le cas de l'*IoT* opéré.

3.3.1 Automatisation et spécification *XML*

La spécification des *SLA* dans le modèle *WSLA* est basée sur le langage *XML* [**?**].

```
1  <!-- Commentaire -->
2  <!-- Elem1 contient une balise et un champ de texte -->
3  <Elem1> champ de texte </Elem1>
4
5  <!-- Elem2 contient un texte et un exemple -->
6  <Elem2 attribute="attributeValue">
7    texte pour Elem2
8    <Elem3 name="exemple"/>
9    <AutreElem> exemple de texte </AutreElem>
10
11 </Elem2>
```

Listing 3.1 – La syntaxe *XML*.

Fonctionnement du langage *XML* Plus qu'un langage, *Extensible Markup Language* (*XML*) est un format d'expression situé à l'interface entre la compréhension humaine et le traitement automatisé de l'ordinateur [**?**]. *XML* définit des éléments, construits à partir de plusieurs composants :

1. les *champs de texte* sont les éléments de base d'une spécification *XML* : ils définissent les valeurs de tous les paramètres exprimés ;
2. les *balises*, délimitées par < et >, identifient chaque élément d'un fichier *XML* : elles peuvent être interprétées de manière spécifique par un logiciel. Elles peuvent ne contenir aucune valeur, contenir un champ de texte, ou bien contenir d'autres sous-éléments délimités par des balises ;
3. les *attributs* : les balises peuvent aussi inclure des attributs, qui précisent leur spécification (e.g. un nom). Chaque attribut prend la forme d'un paramètre et d'une valeur ;
4. les *commentaires* peuvent être insérés par l'auteur pour une meilleure compréhension humaine de la structure du document.

L'extrait de code 3.1 illustre la logique *XML*, il comprend :

1. des commentaires (e.g. lignes 1, 2, 5) ;
2. des balises (e.g. `AutreElem` ligne 9) ;
3. des attributs (e.g. lignes 6, 8) ;
4. des champs de contenu (e.g. ligne 3).

```
1  <!-- Elem1 est simple : une chaine et pas d'attributs -->
2  <xsd:element name="Elem1" type="xsd:string"/>
3
4  <!-- Elem2 contient un ensemble complexe de variables (une liste) et un texte -->
5  <xsd:element name="Elem2">
6    <!-- Elem2 est complexe et de type mélangé -->
7    <xsd:complexType mixed="true">
8      <xsd:sequence>
9        <xsd:element name="Elem3">
10         <xsd:complexType>
11           <xsd:attribute name="name" type="xsd:string"/>
12         </xsd:complexType>
13       </xsd:element>
14       <xsd:element name="AutreElem" type="xsd:string"/>
15     </xsd:sequence>
16   </xsd:complexType>
17 </xsd:element>
```

Listing 3.2 – Le schéma *XML*.

La mise en place du schéma *XML* Les fichiers et la syntaxe *XML* laissent l'auteur complètement libre de créer des documents structurés à partir d'éléments et de valeurs choisies. Afin d'imposer des restrictions sur les documents, il est possible d'utiliser des fichiers de schéma *XML* qui régissent le contenu d'un fichier *XML*, en spécifiant :

1. les éléments qui peuvent apparaître dans le document ;
2. le type de contenu du texte (e.g. chaîne de caractères, nombre flottant, entier, date, etc.) ;
3. les balises attendues pour chaque élément et leur nombre d'occurrences ;
4. les attributs attendus pour chaque balise et leur nombre d'occurrences.

La syntaxe utilisée dans un schéma *XML* est la même que dans un document *XML*. Des éléments *XML* spécifiques y sont définis :

1. les éléments simples sont des éléments qui ne contiennent qu'un champ de texte ;
2. les types simples, tels que des chiffres, des dates ou des chaînes de caractères, sont prédéfinis dans *XML*. Pour définir des éléments avec des attributs, ou des éléments qui incluent d'autres éléments, le schéma *XML* prévoit des éléments complexes. On peut créer des types complexes en les combinant entre eux ;
3. les extensions et restrictions : les attributs permettent de définir des extensions (un élément devient alors l'extension d'un autre) ou des restrictions (e.g. valeur de paramètre limitée à 10) ;
4. les séquences et les choix : un élément *XML* peut comprendre une séquence de sous-éléments ordonnés. Les types d'éléments peuvent être mélangés (e.g. contenir à la fois des champs de texte, d'autres éléments, etc.). Ils peuvent aussi contenir un sous-élément choisi dans une liste. Certains éléments peuvent contenir zéro, une, ou plusieurs instances du même élément (le nombre d'instances est spécifié dans le schéma *XML*).

L'extrait de code 3.2 est un exemple de schéma *XML* correspondant à l'instance *XML* présentée dans l'extrait 3.1. Il montre la déclaration de l'élément simple `Elem1` à la ligne 2. Il montre la déclaration de l'élément complexe `Elem2` à la ligne 5. `Elem2` inclut une séquence des 2 éléments, `Elem3` et `AutreElem` (ligne 14). `Elem3` est de type complexe, il contient un attribut, `name`, qui prend comme valeur une chaîne de caractères (ligne 11).

Pourquoi choisir *XML* pour le modèle de *SLA* ? Le format *XML* est couramment utilisé, en partie du fait de l'importance croissante du *web* et de son utilisation dans *HTML*. *XML* est :

1. expressif : il permet d'écrire des documents sous une forme lisible et structurée ;
2. interopérable : la définition d'un schéma *XML* identique pour tous les modèles de *SLA* facilite la compréhension entre l'opérateur et les clients ;
3. non-ambigu : le schéma *XML* spécifie plusieurs contraintes sur les séquences d'éléments, leurs attributs et valeurs. Et ce document est écrit dans le but d'éviter les mauvaises interprétations des fichiers *XML*. Par conséquent, les clients, les opérateurs et les développeurs ne devraient pas avoir de doutes au moment d'interpréter le code *XML*.

3.3.2 Définition des garanties et conditions dans les *SLA*

La description du service et des performances attendues représente la partie principale des *SLA*. Nous montrons comment l'opérateur et les clients se mettent d'accord sur les paramètres et les valeurs à garantir. Ils définissent la façon d'évaluer les paramètres, quelles sont les métriques correspondantes, et qui est chargé de les fournir.

Dans le modèle *WSLA*, un schéma *XML* de *SLA* est proposé, dans lequel plusieurs éléments peuvent se combiner afin de définir chaque service. Un document *SLA* particulier constitue une instance *XML* spécifiée par le schéma.

Chaque paramètre de *SLA* correspond à une variable qui est d'abord définie, puis évaluée et utilisée dans les *SLO* et les actions qui sont décrits en Section 3.5.3. Les paramètres de *SLA* sont construits comme une séquence de fonctions appliquées sur les métriques, ayant cours pendant un temps spécifié par des échéanciers.

Constructions des métriques complexes *Keller* et *Ludwig* proposent la construction de métriques complexes dans les *SLA*. L'extrait de code 3.3 montre que les métriques sont composées :

1. d'une *source*, ligne 4, indiquant l'acteur en charge de fournir la métrique ;
2. d'un choix entre plusieurs éléments constitutifs, ligne 6 :
   (a) soit d'une *directive de mesure*, définie ligne 23 ;
   (b) soit d'une fonction de métriques (e.g. somme, moyenne, etc.) ;
3. de quatre attributs *name*, *type*, *unit* et *counter*. Ceux-ci caractérisent les propriétés de la métrique.

Les directives de mesure (ou *Measurement Directives*) sont des instructions de mesure de base, adressées au système d'information de l'opérateur, pour la collecte de métriques de performances. Ce sont les éléments formels permettant de définir toutes les autres métriques. Le type *MeasurementDirectiveType* comprend :

1. un élément *ReadingSchedule*, ligne 25, qui définit l'échéancier régissant l'évaluation de la métrique brute ;
2. des extensions nécessaires : il faut inclure d'autres éléments dans les directives (e.g. une adresse faisant référence à une source externe, un facteur de déclenchement de la mesure, etc.).

Chaque métrique ainsi définie permet d'évaluer les performances du réseau sur un aspect donné. Elle est évaluée par l'entité de gestion des *SLA* (le ***SLA Manager***) [?].

```
1  <!-- Copyright 2001, 2002, 2003 IBM Corp -->
2  <xsd:complexType name="MetricType">
3    <xsd:sequence>
4      <xsd:element name="Source" type="xsd:string"/>
5      <xsd:element name="MetricURI" type="xsd:anyURI" minOccurs="0"/>
6      <xsd:choice>
7        <xsd:element name="MeasurementDirective" type="wsla:MeasurementDirectiveType"
       />
8        <xsd:element name="Function" type="wsla:FunctionType"/>
9        <!-- Version 2003 -->
10       <xsd:element name="MeasurementDirectiveVariable" type="wsla:MDVariableType"/>
11     </xsd:choice>
12   </xsd:sequence>
13   <xsd:attribute name="name" type="xsd:string"/>
14   <xsd:attribute name="type" type="wsla:Type"/>
15   <xsd:attribute name="unit" type="xsd:string"/>
16   <xsd:attribute name="counter" type="xsd:boolean" use="optional"/>
17 </xsd:complexType>
18
19 <xsd:element name="Metric" type="wsla:MetricType"/>
20
21 <!-- Measurement directives -->
22
23 <xsd:complexType name="MeasurementDirectiveType" abstract="true">
24   <xsd:sequence>
25     <xsd:element name="ReadingSchedule" type="xsd:string" minOccurs="0"/>
26   </xsd:sequence>
27   <xsd:attribute name="resultType" type="wsla:Type"/>
28 </xsd:complexType>
```

Listing 3.3 – Schéma *XML* de métriques complexes de *SLA*, extrait de [?].

Mise en place des paramètres de *SLA* Les éléments suivants dans la définition de service sont les paramètres de *SLA* (*SLA Parameters*). Ils agrègent les métriques complexes : ils permettent ainsi de définir la structure de haut niveau du *SLA*.

```
1  <!-- Copyright 2001, 2002, 2003 IBM Corp -->
2  <xsd:element name="SLAParameter" type="wsla:SLAParameterType"/>
3
4  <xsd:complexType name="SLAParameterType">
5    <xsd:sequence>
6      <xsd:element name="Metric" type="xsd:string"/>
7      <xsd:element name="Communication" type="wsla:SLAParameterCommunicationType"
       minOccurs="0"/>
8    </xsd:sequence>
9    <xsd:attribute name="name" type="xsd:string"/>
10   <xsd:attribute name="type" type="wsla:Type"/>
11   <xsd:attribute name="unit" type="xsd:string"/>
12 </xsd:complexType>
```

Listing 3.4 – Schéma *XML* des paramètres de *SLA*, extrait de [**?**].

L'extrait de code 3.4 montre qu'un paramètre de *SLA* est construit sur :

1. une métrique (une chaîne de caractères faisant référence à une métrique préalablement définie) ;
2. un élément de communication facultatif : il définit la façon dont chaque paramètre de *SLA* est transmis aux acteurs. Par exemple, un client peut exiger la publication périodique d'un *KPI* ;
3. trois attributs *name*, *type* et *unit*. Ceux-ci caractérisent les paramètres de *SLA* pour les utiliser dans d'autres sections du document.

Les *SLO* et les garanties d'actions sont en effet construits sur des *prédicats* : ils comparent un paramètre de *SLA* à une valeur (l'objectif), sur la base d'un opérateur mathématique de comparaison.

Spécification du temps Les éléments qui spécifient la trame temporelle dans la définition de service sont les échéanciers (ou *schedules*). Ce sont des séries d'intervalles qui divisent une période de temps donnée. Le schéma *XML* permet d'en spécifier les paramètres (durée des intervalles, nombre d'intervalles dans une période, nom de l'élément). Les échéanciers servent dans la définition des directives de mesure, des *SLO* et des garanties d'actions.

```
1  <!-- Copyright 2001, 2002, 2003 IBM Corp -->
2  <xsd:element name="Schedule" type="wsla:ScheduleType"/>
3  <xsd:complexType name="ScheduleType">
4    <xsd:sequence>
5      <xsd:element name="Period" type="wsla:PeriodType"/>
6      <xsd:element name="Interval" type="wsla:IntervalType"/>
7    </xsd:sequence>
8    <xsd:attribute name="name" type="xsd:string"/>
9  </xsd:complexType>
```

Listing 3.5 – Schéma *XML* pour l'échéancier, extrait de [**?**].

L'extrait de code 3.5 montre que les *schedules* sont des séries d'intervalles (ligne 6) qui divisent une période de temps donnée (*Period*, ligne 5) [**?**]. Par exemple, un échéancier peut être défini pour chaque intervalle d'une heure, sur la période de 2014 à 2016.

Keller et *Ludwig* définissent une fonction à partir des échéanciers : le constructeur de série de temps (*TSConstructor*). Les séries temporelles (ou *Time Series*) sont des listes de valeurs associées à un ensemble d'intervalles de temps. Elles permettent d'introduire des conditions complexes dans les *SLA* (e.g. calcul d'un paramètre de *SLA* sur une moyenne temporelle pendant une période donnée).

```
<!-- Copyright 2001, 2002, 2003 IBM Corp -->
<xsd:complexType name="TSConstructor">
  <xsd:complexContent>
    <xsd:extension base="wsla:FunctionType">
      <xsd:sequence>
        <xsd:element name="Schedule" type="xsd:string"/>
        <xsd:choice>
          <xsd:element name="Metric" type="xsd:string"/>
          <xsd:element name="Function" type="wsla:FunctionType"/>
        </xsd:choice>
        <xsd:element name="Window" type="xsd:long" minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Listing 3.6 – Schéma *XML* pour le constructeur de série de temps, extrait de [?].

L'extrait de code 3.6 montre que cette extension du type *FunctionType* agrège :

1. un échéancier, ligne 6, qui donne le cadre temporel permettant de construire les *time series* (l'échéancier définit quand prendre les mesures) ;
2. une référence à une fonction ou à une métrique, qui donne les valeurs dans chaque intervalle ;
3. une fenêtre de temps, ligne 11, qui donne le nombre d'intervalles pris en considération. Par exemple, 8 signifie *les huit dernières valeurs*. On peut définir des conditions sur des fenêtres glissantes ou sur des fenêtres fixes (de date à date).

Spécification du trafic Afin de limiter le risque de saturation du réseau par du trafic en excès par rapport au contrat, le *SLA* doit inclure la définition des paramètres de génération maximale de trafic depuis chacun des nœuds. Ceux-ci peuvent être spécifiés sur la base d'une métrique complexe (taille et quantité de messages) et d'une série temporelle (période minimale de génération).

3.3.3 Outils mathématiques et formels

Dans cette section, nous décrivons tous les outils utilisés dans le document *XML* d'un *SLA* dans le contexte des applications de l'*IoT*. Nous donnons des détails formels des types des éléments et des fonctions, utilisés lors de la construction des paramètres de *SLA*. Nous nous concentrons ici sur les outils qui peuvent être ambigus pour le lecteur.

Les prédicats Les prédicats sont des évaluations booléennes d'une comparaison entre un paramètre de *SLA* et une valeur définie. Voici un exemple de construction d'un prédicat.

```
1  <!-- Copyright 2001, 2002, 2003 IBM Corp -->
2  <xsd:complexType name="Greater">
3    <xsd:complexContent>
4      <xsd:extension base="wsla:PredicateType">
5        <xsd:sequence>
6          <xsd:element name="SLAParameter" type="xsd:string"/>
7          <xsd:element name="Value" type="xsd:double"/>
8        </xsd:sequence>
9      </xsd:extension>
10   </xsd:complexContent>
11 </xsd:complexType>
```

Listing 3.7 – Schéma *XML* pour le prédicat *Greater*, extrait de [**?**].

L'extrait de code 3.7 montre que *Greater* étend le type *PredicateType* avec une séquence de deux éléments :

1. en premier lieu, une chaîne de caractères faisant référence à un paramètre de *SLA*, ligne 6 ;
2. en second lieu, une valeur de type *double*.

Voici un exemple de ce prédicat : *Delay Greater than 10 seconds*.
Plus précisément :

> **Greater** est le booléen correspondant à l'opérateur mathématique commun $>$. Il prend deux opérandes A et B, qui appartiennent au même ensemble ordonné. *Greater* correspond au booléen $A>B$.
>
> **Less** est le booléen correspondant à l'opérateur mathématique commun $<$. Il prend deux opérandes A et B, qui appartiennent au même ensemble ordonné. *Less* correspond au booléen $A<B$.

Les expressions logiques Les expressions logiques sont des opérateurs de prédicats. Ils prennent comme paramètres un ou deux prédicats A et B. Par exemple, *And* permet de composer le prédicat *A and B*.

```
<!-- Copyright 2001, 2002, 2003 IBM Corp -->
<xsd:complexType name="LogicExpressionType">
  <xsd:sequence>
    <xsd:choice>
      <xsd:element name="Predicate" type="wsn_sla:PredicateType"/>
      <xsd:element name="And" type="wsn_sla:BinaryLogicOperatorType"/>
      <xsd:element name="Or" type="wsn_sla:BinaryLogicOperatorType"/>
      <xsd:element name="Not" type="wsn_sla:UnaryLogicOperatorType"/>
      <xsd:element name="Implies" type="wsn_sla:BinaryLogicOperatorType"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>
```

Listing 3.8 – Schéma *XML* pour les expressions logiques, extrait de [**?**].

L'extrait de code 3.1 montre les opérateurs logiques nommés de manière non-ambiguë, dans le modèle *WSLA* : *And*, *Or*, *Not* et *Implies*.

Les fonctions Les fonctions sont utilisées pour combiner des métriques et récursivement des fonctions de métriques. Deux exemples simples :

> **Plus** est la fonction correspondant à l'opérateur mathématique commun +. Il prend deux opérandes A et B de même type. *Plus* est la fonction $A + B$.
>
> **Mean** est la fonction correspondant à l'opérateur mathématique de la moyenne. Elle prend une liste de K valeurs numériques $v_1, v_2, ..., v_K$ Elle a pour résultat $\sum_{i=1}^{K} v_i/K$.

La spécification ne fournit pas l'interprétation par le système d'information de chaque fonction. Par exemple *Plus* est une fonction nommée *Plus* et dispose de 2 opérandes, mais il n'y a pas ici de relation spécifiée avec l'outil mathématique +. Elles sont simplement nommées dans le schéma *XML*, mais elles doivent être définies et spécifiées à un autre endroit pour pouvoir être utilisées sans ambiguïté. L'opérateur doit fournir une description formelle des notions ambiguës.

3.4 Extension du modèle *WSLA* pour l'*IoT* opéré

Nous venons de voir les outils de construction et les éléments constitutifs des *SLA*, tels que spécifiés dans le modèle *WSLA*. Dans cette section, nous détaillons les éléments de structure de *SLA* que nous proposons pour adapter le modèle *WSLA* aux contraintes des réseaux de l'*IoT*.

3.4.1 Structure des *Service Level Agreement (SLA)*

Un document *SLA* consiste en quatre principaux blocs distincts :

1. les acteurs ou *Parties* : ce sont les organisations (entités juridiques, principalement des entreprises) qui sont citées dans le document *SLA* ;
2. les définitions des services, où sont spécifiés les paramètres de *SLA* et les métriques ;
3. les *Service Level Objectives* (*SLO*) qui déterminent les niveaux de service attendus ;
4. les *garanties d'actions* définissant le comportement des acteurs lorsque les *SLO* ne sont pas satisfaits.

La Fig. 3.1 illustre la composition d'un *SLA* pour les réseaux de l'*IoT*. Elle ajoute aux blocs précédemment listés les structures temporelles (les échéanciers et les séries temporelles définis en Section 3.3.2) et physiques (caractéristiques et positionnement des nœuds). Ces structures sont la base de la construction des métriques spécifiques à l'*IoT*.

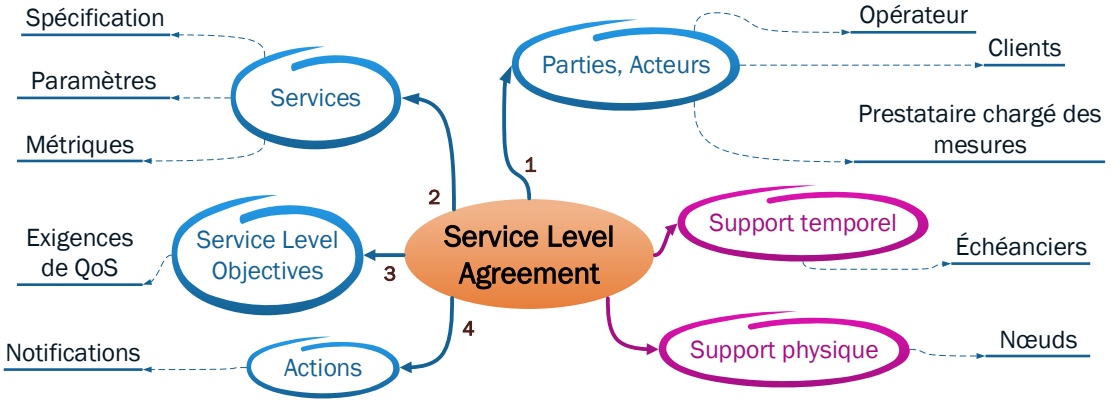


FIGURE 3.1 – Composition des *SLA*.

Exigences des réseaux de l'IoT opérés En effet, le trafic est constitué de multiples flux provenant de nombreuses sources différentes. Les *SLA* doivent s'appliquer sur ces flux et donc les spécifier sans ambiguïté.

Après avoir éxaminé plusieurs contrats entre un opérateur de télécommunications et ses clients dans le domaine de la ville intelligente [2], nous considérons que les principaux *KPI* attendus sont le taux de livraison de bout-en-bout et le délai de transmission des messages reçus de bout-en-bout. Nous imaginons des *SLA* basés pour toutes les applications sur ces deux critères.

3.4.2 Spécification de la liste de nœuds

Nous devons identifier les nœuds et leurs caractéristiques associées. Les informations que le client doit fournir sur chacun de ses nœuds lors de la négociation du *SLA* sont :

1. un identifiant unique (cet *ID* peut être proposé par l'opérateur (e.g. une adresse *IPv6*)) ;
2. la géolocalisation du nœud (latitude, longitude, altitude) qui permet à l'opérateur de (re)dimensionner son réseau si nécessaire ;
3. une liste de *caractéristiques* : celles-ci correspondent aux *activités* possibles du nœud. Par exemple, les nœuds peuvent contenir à la fois des capteurs d'humidité et de température. Ainsi, des *SLA* pour plusieurs applications peuvent être définis.

Les nœuds des clients qui ne sont pas déployés au moment de l'élaboration du contrat doivent être déclarés lors de la négociation : les acteurs peuvent ainsi prévoir l'évolution du service. Par exemple, un client peut déployer ses nœuds feuilles en plusieurs étapes, après avoir contractualisé un *SLA* pour l'ensemble des nœuds.

2. Documents internes à *Orange* dans le cadre d'appels d'offres nationaux. Documents confidentiels non diffusables ni reproduisibles même partiellement.

```
1 <!-- Version 1.0 G. Gaillard, A collaboration between Orange Labs / INSA Lyon, June
      2014 -->
2 <xsd:complexType name="SensorType">
3   <xsd:sequence>
4     <xsd:element name="ID" type="xsd:string"/>
5     <xsd:element name="Position" type="wsn_sla:PositionType"/>
6     <xsd:element name="Feature" type="xsd:string" minOccurs="0" maxOccurs="
      unbounded"/>
7   </xsd:sequence>
8 </xsd:complexType>
```

Listing 3.9 – Schéma *XML* pour la spécification des nœuds.

L'extrait de code 3.9 montre les éléments *XML* d'un nœud, un *ID*, une *Position*, et zéro, une ou plusieurs caractéristiques (*Features*).

```
1  <!-- Version 1.0 G. Gaillard, A collaboration between Orange Labs / INSA Lyon, June
       2014 -->
2  <Devices>
3    <Sensor>
4      <ID>21</ID>
5      <Position>
6        <Latitude>45.15123N</Latitude>
7
8        <Longitude>5.70798E</Longitude>
9        <Height>1.26</Height>
10     </Position>
11     <Feature>GasMeter</Feature>
12     <Feature>GasAlarm</Feature>
13   </Sensor>
14   ...
```

Listing 3.10 – Exemple *XML* d'un nœud.

L'extrait de code 3.10 montre l'exemple *XML* du nœud 21. Le nœud 21 a une position 3D et deux caractéristiques, *GasMeter* et *GasAlarm*, correspondant respectivement à une application de télérelève et à une application d'alarme de gaz.

Ces éléments, considérés pour un *SLA* donné, sont inscrits dans un nouveau bloc que nous proposons, nommé *Devices*, qui liste les nœuds et définit la structure physique du *SLA*, juste avant le bloc de définition de service.

3.4.3 Construction d'ensembles de nœuds

L'objectif de la création d'ensembles de capteurs, ou *Sensor Sets*, est de fournir la possibilité de définir des paramètres de *SLA* sur une collection spécifique de nœuds feuilles. Chaque *Sensor Set* peut être traité comme une liste (*list*), comme un intervalle (*range*), ou comme une *combinaison* (intersections et unions logiques) de *lists* et de *ranges*. Par exemple, le système peut considérer les nœuds :

— ayant la caractéristique *pollution* ;
— ayant leurs identifiants dans la liste (24, 42, 107, ou 1457) ;
— ayant leur latitude dans l'intervalle (10.04N, 10.05N).

Nous présentons la façon de spécifier un ensemble de capteurs, à travers l'élément *XML* *Sensor Set*.

L'extrait de code 3.11 montre que les *Sensor Sets* peuvent utiliser, ligne 4 :

1. des *ranges* (ligne 11) qui définissent une borne inférieure et une borne supérieure pour un paramètre ordonné donné (ligne 16), (e.g. la latitude). Si les identifiants de nœuds constituent un ensemble ordonné (e.g. adresses *IP*, *MAC*), ils peuvent constituer une plage d'*ID* (e.g. *Les ID entiers entre 0 et 99*) ;

2. des *lists* (ligne 19) qui définissent une liste de valeurs possibles pour un paramètre donné. On peut ainsi sélectionner l'union des nœuds décrits par un de leurs paramètres dans la liste. Par exemple, on peut définir des listes d'identifiants, des listes de caractéristiques, des listes d'ensembles de capteurs, etc.

Par conséquent, la collection de nœuds définie par un *Sensor Set* est l'intersection des ensembles définis par ses éléments de type *range* et *list*. Le *Sensor Set* peut représenter une zone géographique, une caractéristique commune, ou même un capteur d'un seul nœud.

```
1  <!-- Version 1.0 G. Gaillard, A collaboration between Orange Labs / INSA Lyon, June
       2014 -->
2  <xsd:element name="SensorSet" type="wsn_sla:SensorSetType"/>
3  <xsd:complexType name="SensorSetType">
4    <xsd:sequence>
5      <xsd:element name="Range" type="wsn_sla:NodeRangeType" minOccurs="0" maxOccurs=
       "unbounded"/>
6      <xsd:element name="List" type="wsn_sla:NodeListType" minOccurs="0" maxOccurs="
       unbounded"/>
7    </xsd:sequence>
8    <xsd:attribute name="name" type="xsd:string"/>
9  </xsd:complexType>
10
11 <xsd:complexType name="NodeRangeType">
12   <xsd:sequence>
13     <xsd:element name="Min" type="xsd:string"/>
14     <xsd:element name="Max" type="xsd:string"/>
15   </xsd:sequence>
16   <xsd:attribute name="ParameterType" type="xsd:string"/>
17 </xsd:complexType>
18
19 <xsd:complexType name="NodeListType">
20   <xsd:simpleContent>
21     <xsd:extension base="wsn_sla:StringList">
22       <xsd:attribute name="ParameterType" type="xsd:string"/>
23     </xsd:extension>
24   </xsd:simpleContent>
25 </xsd:complexType>
```

Listing 3.11 – Schéma *XML* des ensembles de capteurs (*Sensor Sets*).

```
1  <!-- Version 1.0 G. Gaillard, A collaboration between Orange Labs / INSA Lyon, June
       2014 -->
2  <SensorSet name="First Sensor Set example: the Urban Pollution Monitoring">
3    <Range ParameterType="Latitude">
4      <Min>10.04N</Min>
5      <Max>10.05N</Max>
6    </Range>
7    <List ParameterType="Feature">
8      pollution
9    </List>
10   <List ParameterType="ID">24 42 107 1457</List>
11 </SensorSet>
```

Listing 3.12 – Exemple *XML* d'un ensemble de capteurs (*Sensor Set*).

L'extrait de code 3.12 montre l'expression du *Sensor Set* donné en exemple au début de cette Section 3.4.3. Cette définition de *Sensor Sets* permet aussi de spécifier les paramètres de performances concernant la diffusion vers un ensemble de nœuds donné. Par exemple, on peut définir le pourcentage de messages de configuration reçus pour tous les capteurs d'un bâtiment.

Dans la Section 3.3.2 nous avons expliqué la création de paramètres de *SLA* en combinant des métriques en fonction de métriques de base. Nous avons décrit la fonction *TSConstructor* dans l'extrait de code 3.6, qui permet de construire des séries temporelles d'un paramètre donné. Dans le contexte de l'*IoT*, les paramètres de *SLA* s'expriment de

manière distribuée sur un ensemble de nœuds. Nous avons donc besoin d'une fonction qui permette de construire une déclinaison d'un paramètre sur un ensemble de nœuds (e.g. la liste des valeurs de délai pour les capteurs d'un bâtiment).

Nous proposons de construire des séries d'ensembles de nœuds, ou *Sensor Series*, à l'aide d'un constructeur nommé *Sensor Series Constructor*. Les *Sensor Series* sont des listes de valeurs pour un *Sensor Set* donné. Le schéma *XML* du constructeur est très similaire à celui du *TSConstructor* (extrait de code 3.6) : nous y incluons un élément *SensorSetName* qui indique quel *Sensor Set* est considéré. Ainsi, il est possible de construire une métrique donnant la liste des résultats d'une directive de mesure appliquée sur un *Sensor Set*.

3.4.4 Métriques particulières des réseaux de l'*IoT* opérés

Pour mettre en place les *SLA* pour l'*IoT*, il faut définir des directives de mesure spécifiques. Dans cette sous-section, nous listons les métriques de base dédiées aux réseaux de l'*IoT* que nous utilisons pour spécifier les *SLA*. Les mesures résultent de l'observation de la performance du réseau. Le système d'information de l'opérateur les fournit quand un acteur a besoin d'évaluer la satisfaction des *SLA*.

Nous composons ces métriques de base grâce à l'ensemble des fonctions disponibles, afin de construire des métriques complexes et les paramètres de *SLA* correspondants.

Métriques basiques de livraison pour les *SLA* de l'*IoT* Dans le modèle *WSLA* [**?**], *Keller* et *Ludwig* proposent des directives de mesure que nous avons décrites en Section 3.3.2. Nous adoptons leur formalisme pour spécifier les métriques de base dédiées aux réseaux de l'*IoT*.

Device Generated Message Counter Le compteur de messages générés par les nœuds, ou *Device Generated Message Counter (DGMC)*, correspond au nombre de messages générés par chaque nœud d'un *Sensor Set*. Il est défini :

1. sur un ensemble de nœuds sources du client ;
2. au cours d'une période de temps donnée ;
3. pour une caractéristique donnée.

Le *DGMC* peut être calculé en comptant les messages à leur génération (sur le nœud feuille) ou à leur entrée sur le réseau de l'opérateur (sur un routeur d'entrée).

Device Received Message Counter Le compteur de messages reçus par nœud, ou *Device Received Message Counter (DRMC)*, correspond au nombre de messages reçus par chaque nœud d'un *Sensor Set*. Il est défini de la même manière que le *DGMC*. Le *DRMC* peut être calculé en comptant les messages une fois reçus.

System Generated Message Counter Le compteur de messages générés depuis le système d'information, ou *System Generated Message Counter (SGMC)*, correspond au nombre de messages générés depuis le système d'information du client vers chaque nœud d'un *Sensor Set*. Il est défini de la même manière que le *DGMC*. Le *SGMC* peut être calculé en comptant les messages à leur génération (évaluation par le client) ou à leur réception sur une passerelle de l'opérateur.

System Received Message Counter Le compteur de messages reçus par le système d'information, ou *System Received Message Counter (SRMC)*, correspond au nombre de messages reçus par le système d'information du client depuis chaque nœud d'un *Sensor Set*. Il est défini de la même manière que le *DGMC*. Le *SRMC* peut être calculé au passage des messages sur une passerelle de l'opérateur, ou à leur réception au SI du client.

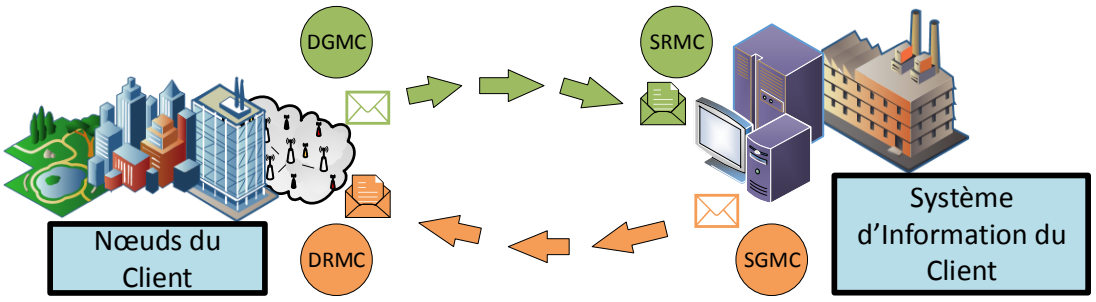


FIGURE 3.2 – Les métriques basiques de livraison.

La Fig. 3.2 illustre les quatre compteurs introduits :

1. le *DGMC* et le *DRMC* sont évalués sur les nœuds du client ou sur les relais d'entrée ;
2. le *SRMC* et le *SGMC* sont évalués sur les passerelles ou au niveau du système d'information du client.

Muni de ces quatre métriques de base, l'opérateur peut spécifier une gamme de *KPI* liés au taux de livraison de bout-en-bout, dans les deux sens de communication. L'évaluation de ces métriques doit être précisée dans les *SLA*. En effet, la méthode de mesure et l'endroit impactent les valeurs prises : par exemple dans le cas de pertes de trames entre un nœud feuille et un relais d'entrée, les *DGMC* évalués de part et d'autre diffèrent. L'opérateur évalue les garanties qu'il peut donner de bout-en-bout dans le cas où il contrôle les communications des nœuds feuilles (il évite ainsi une surexploitation frauduleuse du médium).

Dans les chapitres suivants, nous focalisons notre étude sur le trafic de remontée (collecte). Au Chapitre 5, nous restreignons notre étude au réseau radio de l'opérateur : la mesure du nombre de trames entrantes est réalisée sur les routeurs d'entrée, et en sortie sur les passerelles de l'opérateur. Au Chapitre 6, nous étudions l'allocation de ressources en incluant les nœuds feuilles, les routeurs d'entrée, les relais intermédiaires et les passerelles (en réception).

```
1  <!-- Version 1.0 G. Gaillard, A collaboration between Orange Labs / INSA Lyon, June
       2014 -->
2  <!-- Device Generated Message Counter-->
3  <xsd:complexType name="DGMC">
4    <xsd:complexContent>
5      <xsd:extension base="wsn_sla:MeasurementDirectiveType">
6        <xsd:sequence>
7          <xsd:element name="SensorSetName" type="xsd:string"/>
8        </xsd:sequence>
9      </xsd:extension>
10   </xsd:complexContent>
11 </xsd:complexType>
```

Listing 3.13 – Schéma *XML* pour le *DGMC*.

```
1  <!-- Version 1.0 G. Gaillard, A collaboration between Orange Labs / INSA Lyon, June
       2014 -->
2  <Metric name="SouthMinuteDGMC" type="integer" unit="messages">
3    <Source>WSNoperator</Source>
4    <MeasurementDirective xsi:type="DGMC" resultType="integer">
5      <ReadingSchedule>Minutely</ReadingSchedule>
6      <SensorSetName>South of Grenoble, France</SensorSetName>
7    </MeasurementDirective>
8  </Metric>
```

Listing 3.14 – Exemple *XML* d'un *DGMC*.

L'extrait de code 3.13 montre le schéma *XML* du *DGMC*. Il s'agit d'une extension du type *MeasurementDirectiveType* [**?**], definie en Section 3.3.2, extrait de code 3.3, ligne 23. Le *DGMC* utilise l'élément *ReadingSchedule* du type *MeasurementDirectiveType* comme structure de temps. Il ajoute un élément *SensorSetName*, ligne 7, qui indique l'ensemble de nœuds considérés.

L'extrait de code 3.14 montre un exemple d'utilisation du *DGMC*. Il spécifie un échéancier par minute, *Minutely*, ligne 5, et un ensemble de nœuds, *South of Grenoble, France*, ligne 6.

Métriques basiques de délai pour les *SLA* de l'IoT Ce paragraphe traite de l'évaluation de base du délai de transmission des messages de bout-en-bout.

> **Device to IS Delay** Le *Device to IS Delay (DID)* correspond à la liste des temps entre la génération de tous les messages de chaque capteur d'un ensemble de nœuds, et leur réception par le système d'information au cours d'une période de temps donnée.
>
> Le *DID* pourrait être calculé en incluant un horodatage sur les messages lors de leur génération, en faisant l'hypothèse que les nœuds sont synchronisés (voir Chapitre 5).
>
> **IS To Device Delay** Le *IS To Device Delay (IDD)* correspond à la liste des temps entre la génération de tous les messages reçus du système d'information et leur réception par chaque nœud au cours d'une période de temps donnée.
>
> L'*IDD* pourrait être calculé de la même manière que le *DID*.

De même que pour les métriques basiques de livraison, les *SLA* doivent spécifier la méthode et le lieu de mesure. Dans les chapitres suivants, nous restreignons notre évaluation du délai au réseau radio : nous considérons le temps écoulé entre l'émission d'une trame sur un nœud feuille et sa réception sur une passerelle.

3.5 Mise en place de *SLA* pour les réseaux de l'*IoT* opérés

Nous venons de proposer des extensions au modèle *WSLA* permettant de l'utiliser dans le cadre de l'*IoT*. Dans cette section, nous détaillons la constitution des blocs de *SLA* que nous avons listés à la Fig. 3.1.

3.5.1 Les acteurs de l'*IoT*

Dans le modèle *WSLA*, trois sortes d'acteurs sont définis comme extension du type *wsla :SignatoryPartyType* :

1. le fournisseur de services : l'acteur qui offre un service aux clients ;
2. le demandeur de services : le client ;
3. les prestataires tiers nommés *Supporting Parties* : d'autres acteurs chargés de tâches spécifiques définies dans le *SLA*. Par exemple, ils peuvent fournir des valeurs de certains paramètres de performance, ou réaliser des actions spécifiques (e.g. *Notification*).

Dans le cas de l'*IoT*, nous envisageons les mêmes rôles pour l'opérateur, le client et des parties tierces qui peuvent par exemple réaliser des campagnes de mesures de performance du réseau indépendamment de l'opérateur.

3.5.2 La définition des services

Ce bloc contient la description de chaque service, à partir des éléments que nous avons construits en Section 3.3.2 :

— les échéanciers temporels ou *schedules* ;
— les ensembles de nœuds ou *sensor sets* ;
— les métriques complexes composées à partir des métriques de base pour l'*IoT* (qui sont définies en Section 3.4.4) ;
— les paramètres de *SLA* ou *SLA Parameters* : ceux-ci correspondent au plus haut niveau de définition des services (extrait de code 3.4).

La construction des paramètres de *SLA* est adaptée aux réseaux de l'*IoT* opérés :

1. elle donne accès aux éléments essentiels : construction à partir de métriques de base prenant en compte la multiplicité des nœuds et des applications ;
2. elle donne la liberté à l'opérateur de choisir le degré de précision des paramètres sur lesquels il s'engage ;
3. elle donne de la liberté à l'opérateur dans les restrictions sur les valeurs possibles des paramètres.

3.5.3 L'expression des besoins des clients : les *Service Level Objectives (SLO)*

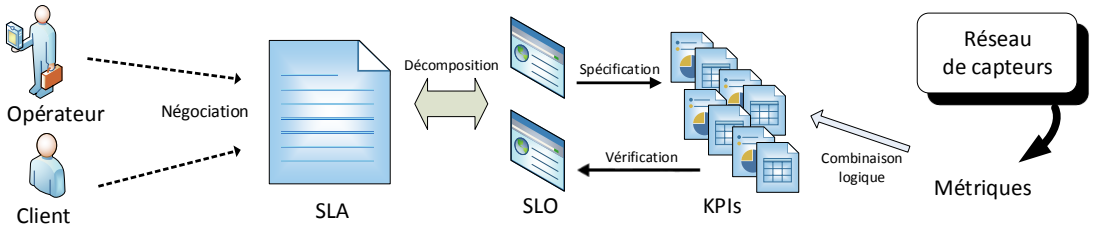


FIGURE 3.3 – Mise en place des *SLO* et lien avec les métriques de réseau.

Pour l'opérateur, le *SLA* correspond à une composition logique de plusieurs niveaux de services, ou *Service Level Objectives* (*SLO*) [**?**]. Les *SLO* constituent la forme aboutie de la spécification des *SLA*, qui fait l'objet d'une négociation entre un client et l'opérateur.

```
<!-- Copyright 2001, 2002, 2003 IBM Corp -->
<xsd:complexType name="ServiceLevelObjectiveType">
  <xsd:sequence>
    <xsd:element name="Obliged" type="xsd:string"/>
    <xsd:element name="Validity" type="wsla:PeriodType" maxOccurs="unbounded"/>
    <xsd:element name="Expression" type="wsla:LogicExpressionType"/>
    <xsd:choice>
      <xsd:element name="EvaluationEvent" type="wsla:EvaluationEventType"/>
      <xsd:element name="Schedule" type="xsd:string"/>
    </xsd:choice>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string"/>
</xsd:complexType>
```

Listing 3.15 – Schéma *XML* pour les Service Level Objectives, extrait de [**?**].

Chaque *SLO* correspond à un *Key Performance Indicator* (*KPI*), une valeur attendue et un ensemble de nœuds concernés. La Fig. 3.3 montre les éléments de la décomposition d'un

SLA en *SLO*. Les acteurs s'accordent sur chaque *KPI* à observer (e.g. le taux de livraison de bout-en-bout) et sur les valeurs attendues (e.g. un *SLO* pourrait spécifier une valeur minimale de 80% pour le taux de livraison des messages provenant des nœuds feuilles d'un immeuble une fois par jour).

Les *Service Level Objectives* (*SLO*) expriment des conditions en utilisant les paramètres de *SLA* définis dans les définitions de service. L'expression de ces conditions comprend des prédicats ou des combinaisons de prédicats. Les *SLO* donnent un résultat booléen (s'ils sont validés ou non, e.g. *MeanDelay_Parameter_1 Less than 150 s*).

L'extrait de code 3.15 montre le contenu d'un *SLO*, et comment combiner les prédicats pour former les conditions du *SLO*. Dans le modèle *WSLA*, un *SLO* contient :

— l'acteur responsable *obliged party*, ligne 4, qui correspond à l'acteur en charge du respect du *SLO* ;
— une période de validité, au cours de laquelle le *SLO* est applicable ;
— une expression, ligne 6 : elle peut être un prédicat, ou une combinaison logique de prédicats. Elle peut inclure des opérateurs logiques (cf. Section 3.3.3) ;
— un cadre de temps d'évaluation, qui définit la manière et le moment où le *SLO* est évalué. Il peut être périodique ou s'appliquer lorsqu'une nouvelle valeur est disponible.

```
1  <ServiceLevelAgreement name="GasComp">
2    <ServiceLevelObjective name="GasDayCR">
3      <Obliged>WSNOperator</Obliged>
4      <Validity>
5        <Start> 2016-01-01 </Start>
6        <End> 2030-12-31 </End>
7      </Validity>
8      <Expression>
9        <Predicate xsi:type="Greater">
10         <SLAParameter> DayCollectRatio </SLAParameter>
11         <Value> 0.95 </Value>
12       </Predicate>
13     </Expression>
14     <EvaluationEvent> Daily </EvaluationEvent>
15   </ServiceLevelObjective>
16   <ServiceLevelObjective name="GasWeekCR">
17     ...
18 </ServiceLevelAgreement>
```

Listing 3.16 – Code source *XML* pour la spécification d'un *Service Level Objective* (*SLO*).

L'exemple de *SLO* illustré par l'extrait de code 3.16 exprime l'exigence de *GasComp*, un distributeur de gaz : 95% des index de ses compteurs de gaz doivent être collectés chaque jour par l'opérateur *WSNOperator*. Le *SLO* comprend :

1. une métrique complexe (un paramètre de *SLA*) : `DayCollectRatio` ;
2. un seuil (l'objectif de service, i.e. la valeur attendue, e.g. 95%) : le champ `Value` ;
3. un opérateur de comparaison interne (entre seuil et valeur de la métrique) : `Greater` ;
4. un cadre temporel définit par :
   (a) la période de validité : `Validity` ;
   (b) la fréquence et le mode d'évaluation : `EvaluationEvent`.

Certains *SLO* définissent les obligations des clients : le comportement des applications est ainsi contractualisé dans les *SLA*, et vérifié en temps réel, afin d'éviter une surconsommation des ressources du réseau.

En définissant plusieurs *SLO* sur le même paramètre de *SLA*, avec différents seuils et cadres temporels, le *SLA* spécifie les objectifs de service dégradé. Par exemple, si l'objectif `DayCollectRatio>95%` n'est pas atteint sur une journée, le *SLA* est dégradé, on évalue le

SLO `DayCollectRatio>80%` les deux jours suivants. Si ce dernier n'est pas non plus atteint, le *SLA* est rompu.

3.5.4 Les garanties d'action

Les éléments de garantie d'action (nommés *ActionGuarantee*) définissent ce que les acteurs doivent faire dans certains cas (e.g. quand le *SLA* est dégradé). Leurs contenus sont similaires à ceux des *SLO* :

```
1  <!-- Copyright 2001, 2002, 2003 IBM Corp -->
2  <xsd:complexType name="ActionGuaranteeType">
3    <xsd:sequence>
4      <xsd:element name="Obliged" type="xsd:string" maxOccurs="unbounded"/>
5      <xsd:element name="Expression" type="wsla:LogicExpressionType"/>
6      <xsd:choice>
7        <xsd:element name="EvaluationEvent" type="wsla:EvaluationEventType"/>
8        <xsd:element name="Schedule" type="xsd:string"/>
9      </xsd:choice>
10     <xsd:element ref="wsla:QualifiedAction" maxOccurs="unbounded"/>
11     <xsd:element name="ExecutionModality" type="wsla:ExecutionModalityType"/>
12   </xsd:sequence>
13   <xsd:attribute name="name" type="xsd:string"/>
14 </xsd:complexType>
```

Listing 3.17 – Schéma *XML* pour les garanties d'action (ActionGuarantee), extrait de [?].

L'extrait de code 3.17 montre les composants d'une garantie d'action :

- l'acteur *obliged party*, qui doit exécuter l'action (ligne 2) ;
- l'élément *expression*, sous forme d'une combinaison logique de prédicats, avec opérateurs logiques, qui reflète la condition qui déclenche l'action (ligne 5) ;
- une structure temporelle d'évaluation, pendant laquelle la condition d'action est évaluée. Celle-ci peut être périodique, ou initiée lorsqu'une nouvelle valeur de l'expression est disponible ;
- certains éléments potentiels, comme le *QualifiedAction*, ligne 10, qui décrivent plus en détails les actions concrètes à mener ;
- les modalités d'exécution de ces actions (e.g. *always*).

Dans le modèle *WSLA*, les auteurs fournissent la spécification *XML* des actions de *notification de violation* et de *notification d'information*. Nous proposons un autre type d'action qui change la granularité de l'observation pour les métriques de base :

```
1  <!-- Version 1.0 G. Gaillard, A collaboration between Orange Labs / INSA Lyon, June
       2014 -->
2  <xsd:complexType name="SetMonitoringSchedule">
3    <xsd:complexContent>
4      <xsd:extension base="wsn_sla:ActionInvocationType">
5        <xsd:sequence>
6          <xsd:element name="MeasurementDirective" type="xsd:string"/>
7          <xsd:element name="NewSchedule" type="xsd:string"/>
8          <xsd:element name="CausingGuarantee" type="xsd:string"/>
9          <xsd:element name="SLAParameter" type="wsn_sla:StringList" minOccurs="0"/>
10       </xsd:sequence>
11     </xsd:extension>
12   </xsd:complexContent>
13 </xsd:complexType>>
```

Listing 3.18 – Schéma *XML* pour l'action de changement de fréquence d'observation.

L'extrait de code 3.18 montre le schéma *XML* de l'action *SetMonitoringSchedule*. Celle-ci étend le type *ActionInvocationType* en lui ajoutant trois éléments :

1. l'élément *MeasurementDirective*, ligne 6 : la métrique de base visée ;
2. l'élément *NewSchedule* : le nouvel échéancier à appliquer ;

3. l'élément *CausingGuarantee* : le *SLO* qui n'a pas été respecté.

Avec ce type d'actions, les documents *SLA* permettent de préciser la stratégie de contrôle du réseau de l'*IoT*.

Voici un exemple d'application :

```
<!-- Version 1.0 G. Gaillard, A collaboration between Orange Labs / INSA Lyon, June 2014 -->
<QualifiedAction>
  <Party>WSNMeasurements</Party>
  <Action actionName="ChangeMonitoring" xsi:type="SetMonitoringSchedule">
    <MeasurementDirective>DeviceToISDelay</MeasurementDirective>
    <NewSchedule>Minutely</NewSchedule>
    <CausingGuarantee>SouthDailyDeviceToISDelay</CausingGuarantee>
  </Action>
</QualifiedAction>
```

Listing 3.19 – Exemple *XML* d'une action de changement de fréquence d'observation.

L'extrait de code 3.19 montre la description de l'Action. Elle explique ce qu'un tiers *WSN-Measurements* doit faire si le *SLO SouthDailyDeviceToISDelay* n'est pas respecté. Ici, la période d'évaluation de la directive de mesure *DeviceToISDelay* doit être réduite à une minute.

3.6 Conclusion et perspectives

Dans ce chapitre, nous avons abordé la question de la spécification de *SLA* pour les réseaux de l'*IoT* opérés. Nous avons détaillé comment un opérateur peut spécifier un contrat de qualité de service avec chaque client, sous la forme d'un *Service Level Agreement (*SLA*)*. Les *SLA* obligent l'opérateur à respecter les exigences spécifiées pour les clients et négociées en fonction de leurs applications. Chaque *SLA* exprime les limites des garanties que chaque acteur fournit, ainsi que les actions qui doivent être exécutées si les objectifs de niveau de service (*Service Level Objectives* (*SLO*)) ne sont pas respectés.

Dans cet objectif, nous avons étendu le modèle *WSLA* pour le rendre utilisable dans le domaine de l'*IoT* opéré. Nous avons spécifié des ensembles de nœuds et la façon de traiter une liste donnée de nœuds. Nous avons proposé des métriques de base que nous considérons fournies par les mécanismes de gestion du réseau : les compteurs de messages et la façon de les utiliser, en particulier sur un ensemble de capteurs donné. Nous avons également défini des mesures de délai de transit pour les messages reçus. Ceux-ci peuvent être évalués soit par des entités tierces d'analyse de la performance du réseau, soit par l'opérateur lui-même.

Nous donnons un exemple complet de *SLA* spécifié pour une application de télérelève de gaz [**?**]. La mise en œuvre effective de *SLA* rédigés en langage *XML* fait partie des travaux futurs. Elle sera facilitée par le fait que *XML* est répandu et facile à intégrer au SI d'un opérateur.

D'autres métriques peuvent être intégrées aux *SLA* de manière à prendre en considération d'autres aspects de l'*IoT* (e.g. la sécurité des transmissions).

Les aspects économiques et juridiques (e.g. montant des dédommagements en cas de dégradation du service) ne sont pas le sujet de ce chapitre, mais doivent toutefois faire partie de la stratégie de l'opérateur lors de la mise en place des *SLA*.

Enfin, nous devons intégrer la spécification des *SLA* dans l'architecture de gestion que nous proposons [**?**]. En particulier, l'interprétation des *SLA* en *XML* pour le contrôle d'admission, l'évaluation de la viabilité d'un *SLA* et la configuration du réseau doivent être mises en œuvre. Le travail décrit dans le prochain chapitre s'appuie sur cette spécification.

Chapitre 4

Architecture de gestion opérée de réseaux de l'*IoT*

Notre point de vue est celui d'un opérateur de l'Internet des Objets (*IoT*) désireux de partager une infrastructure de réseau de capteurs entre plusieurs clients. Nous envisageons le cas d'un opérateur de l'*IoT* qui utiliserait la pile protocolaire *6TiSCH* pour garantir, comme décrit au Chapitre 2, la Qualité de Service (*QoS*) de plusieurs applications ayant des contraintes différentes.

Nous nous plaçons donc dans ce cas, où un ensemble de nœuds échange des messages en utilisant des communications multi-sauts. L'opérateur est alors en charge de l'acheminement des informations de chaque client depuis leur génération jusqu'à leur arrivée dans le système d'information du client (Fig. 4.1). Nous nous concentrons sur la partie multi-saut sans fil du réseau.

4.1 Introduction

Nous tirons profit du cadre existant pour ce type de contrat : les *Service Level Agreements* (*SLA*) [**?**]. Les *SLA* contiennent les clauses contractuelles sur lesquelles l'opérateur et le client se mettent d'accord. Ils fixent les termes de rupture, de dysfonctionnement avéré, de dégradation et de respect des exigences des clients. Ils comprennent aussi les obligations techniques et fonctionnelles (dédommagement, interventions humaines sur le réseau), correspondant à chaque situation possible. Dans le précédent chapitre, nous avons proposé la spécification et la mise en place de *SLA*. Nous positionnons celle-ci dans l'état de l'art spécifique en termes de *SLA*, en particulier pour les réseaux de capteurs sans fil et l'*IoT*.

Les opérateurs de réseaux de l'*IoT* ont besoin d'une technologie robuste et fiable, dans le but de garantir le respect de contraintes de *QoS* dans l'environnement radio, similaires à celles qui sont attendues dans le monde industriel [**?**]. Nous avons vu dans le Chapitre 2 en quoi la technologie *IEEE Std 802.15.4-2015* en mode *TSCH* [**?**] est une bonne candidate pour la mise en place d'une architecture de gestion de *SLA*, dédiée aux réseaux de l'*IoT* contrôlés par un opérateur. En effet, le standard est prévu pour traiter des exigences de *QoS* dans plusieurs scénarios applicatifs [**?**].

Ces opérateurs doivent implémenter toutes les fonctionnalités nécessaires à la gestion de *SLA* : l'observation, le contrôle d'admission, la configuration et le respect de performances. L'objectif est ici de proposer une architecture fonctionnelle qui englobe ces fonctionnalités et les met en relation les unes avec les autres. Afin de réaliser cela :

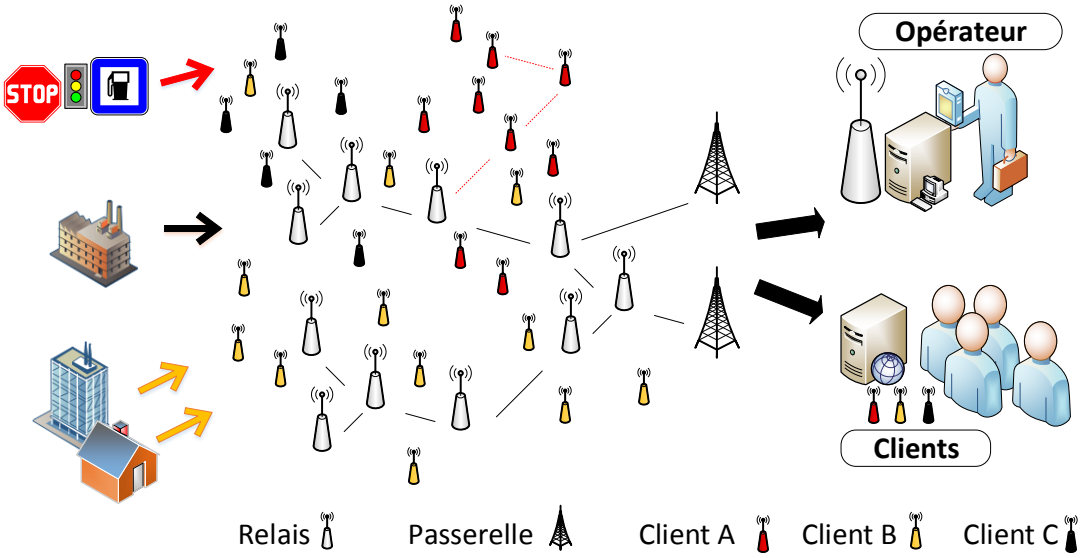


FIGURE 4.1 – Acteurs, topologie physique et types de nœuds d'un réseau de l'*IoT* opéré.

1. nous proposons une architecture fonctionnelle pour gérer les *SLA*, c'est-à-dire les clauses de Qualité de Service (*QoS*) qu'ils contractualisent avec leurs clients ;
2. nous donnons des orientations formelles pour la mise en place d'une telle architecture de *SLA* pour les réseaux de l'*IoT* opérés. Nous distinguons chaque fonction nécessaire pour opérer un réseau de l'*IoT* selon des *SLA*, et déterminons quelles entités fonctionnelles sont nécessairement dépendantes de la technologie ;
3. nous donnons des exemples détaillés, instanciés dans le contexte de *6TiSCH*.

Dans ce chapitre, nous fournissons d'abord le scénario et les défis que nous affrontons (Section 4.2). Nous montrons les principales caractéristiques de l'architecture proposée (Section 4.3). Puis chaque entité fonctionnelle est décrite en détail et nous expliquons la mise en œuvre que nous recommandons en Section 4.4, avant de conclure et de résumer les apports de l'architecture.

4.1.1 Les architectures de gestion des *SLA*

La gestion de *SLA* par l'opérateur nécessite que toutes les fonctionnalités introduites ci-dessus soient combinées dans une architecture de *SLA* complète.

IBM a dédié une partie de ses efforts de recherche aux *web services*, et dans cet objectif, *Keller* et *Ludwig* ont remis au goût du jour la notion de *SLA* en leur donnant une spécification formelle [**?**]. Ils proposent une décomposition des *SLA* basée sur un modèle exprimé en langage *XML*. Nous nous en inspirons pour notre architecture dans le contexte des réseaux de capteurs et de l'*IoT*.

Les fonctionnalités requises pour la gestion de *SLA* dans les réseaux sans fil et en particulier pour les réseaux de capteurs ont fait l'objet d'études et de propositions variées. En 2003, *Linnyer Beatrys Ruiz et al.* proposent *MANNA*, une architecture de gestion de réseaux de capteurs sans fil [**?**]. *MANNA* concentre les fonctionnalités nécessaires pour fournir des services sur de tels réseaux (e.g. la découverte de la topologie, l'observation de l'environnement radio). L'architecture propose un modèle de représentation du réseau et la mise en place d'entités chargées de la gestion (nommés agents ou *WSN Manager*).

Yang et al. établissent une répartition des rôles de fournisseur de services, d'opérateur et d'usagers dans les réseaux sans fil hétérogènes [**?**]. Les auteurs proposent une architecture permettant d'appliquer des *SLA* et d'observer le respect de contraintes de *QoS*. Les services concernent l'usager (e.g. terminaison d'appel) et sont envisagés pour des réseaux cellulaires ou *Wi-Fi* et des terminaux multimédia.

Snoeck et al. [**?**] construisent un exemple d'architecture de gestion avec *SLA*, dédiée aux réseaux de distribution d'électricité. L'architecture intègre la mise en place de *SLA* en négociant avec chaque client sur la base de modèles de *SLA*, qui spécifient les services. Les *SLA* négociés sont intégrés au processus courant. L'architecture supervise le *SLA* pendant tout son cycle de vie en comparant les performances obtenues à celles attendues et en informant le client sur le résultat. Nous souhaitons spécifier une architecture de *SLA* similaire pour les réseaux de l'*IoT* opérés (en particulier, les réseaux de capteurs urbains). L'architecture doit permettre de contrôler la configuration du réseau de manière à s'adapter à de nouveaux *SLA* et de nouvelles conditions.

La mise en œuvre logicielle de la gestion de *SLA* dépend de la technologie employée. *Del Cid et al.* proposent une architecture logicielle légère permettant de distribuer la gestion de *SLA* sur les nœuds des réseaux de capteurs [**?**]. Le projet *VITRO* [**?**] propose une couche logicielle d'abstraction qui rend possible la virtualisation afin de partager les données des capteurs entre plusieurs usagers. Enfin, *Octopus* [**?**] est un outil développé en code source ouvert pour *TinyOS* qui permet l'observation, la visualisation et le contrôle humain des réseaux de capteurs.

Cependant, ces approches ne couvrent pas le maintien de *SLA* garantis : l'observation de ressources de nœuds et du réseau ne suffit pas à fournir des garanties de *QoS* au niveau de la couche réseau. Nous proposons une architecture complète et générique de gestion de *SLA* pour réseaux opérés. Nous décrivons maintenant les acteurs et leurs interactions.

4.2 Les acteurs de l'*IoT* opéré, leurs rôles et fonctionnement

Le développement de l'Internet des Objets augmente les coûts de déploiement des services, en particulier pour les réseaux radio multi-sauts (e.g. déploiements de relais dans les villes, liés à l'amélioration des services urbains par l'*IoT*, pour former les *Smart Cities* [**?**]).

Les compagnies de services aux particuliers (e.g. les fournisseurs ou distributeurs de gaz), tendent à déléguer la gestion des télécommunications à des entreprises spécialisées, les opérateurs de réseaux de l'*IoT*, dont le but est de partager les ressources en bande passante radio, pour tous leurs clients. En particulier, l'environnement radio dans les villes est de plus en plus dense en termes d'occupation de ressources ; l'usage de bandes de fréquences non dédiées et non licenciées rend complexe pour les clients et les fournisseurs la mise en place de contrats de *QoS* sur le service de communication.

4.2.1 Un environnement multi-client et multi-service

Nous précisons l'environnement étudié grâce à la Fig. 4.2. Celle-ci illustre la manière pour l'opérateur de collecter les données des usagers.

Les *nœuds feuilles* des clients Les nœuds des clients sont appelés *nœuds feuilles* car ils se situent au plus proche de l'usager, en extrémité du réseau. Ces nœuds génèrent les messages et émettent les trames : les nœuds feuilles sont les sources du trafic applicatif.

Parfois, un client peut agréger sur un nœud feuille le trafic de plusieurs sources, à l'aide d'une technologie de collecte qui lui est propre. Le client présente alors à l'opérateur un seul nœud feuille comprenant tout le trafic de l'ensemble des capteurs qui lui sont associés (c'est le cas du client A de la Fig. 4.1 qui peut représenter par exemple un répéteur placé au bout d'une rue pour l'ensemble des lampadaires de celle-ci).

L'architecture doit considérer les contraintes liées aux propriétés des nœuds :

1. chaque nœud possède une unique interface radio, à l'alternat ou *half-duplex* (pas de transmission et réception simultanément) ;
2. chaque nœud a une capacité de stockage de trames limitée par sa mémoire.

Chaque feuille supporte une ou plusieurs applications (le nœud correspondant peut contenir un ou plusieurs capteurs physiques, indépendamment). Chaque application génère des messages potentiellement fragmentés en plusieurs trames. La Fig. 4.2 montre trois feuilles, l'une d'entre elles (la feuille A) supportant deux applications.

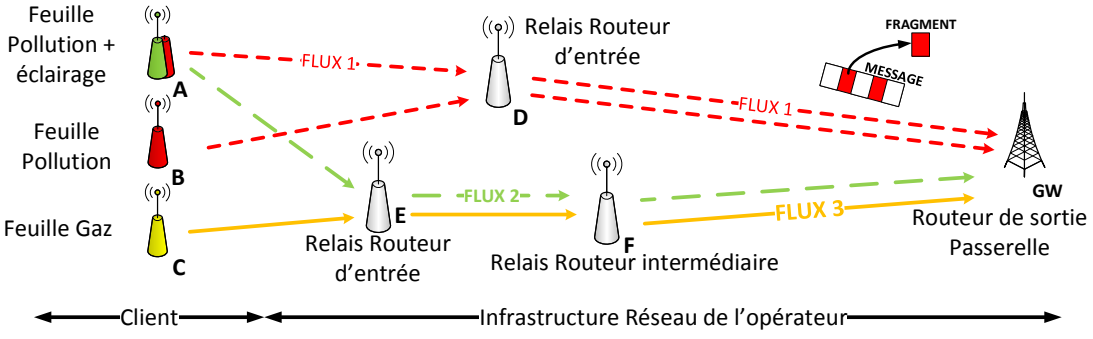


FIGURE 4.2 – Constitution du réseau de l'opérateur

Le trafic client Pour chaque application, un nœud feuille crée un *flux* de messages. Nous considérons que le trafic client est convergent (les relais intermédiaires acheminent les trames vers un ou plusieurs points de collecte). Un flux est caractérisé par sa source, son trafic et l'application à laquelle il se rattache. Le trafic lié à une application donnée se caractérise par un ensemble multi-source de flux.

L'opérateur fournit des garanties de *QoS* sur les flux qui respectent la spécification des *SLA* [**?**]. Nous considérons dans le contexte de *6TiSCH* que pour chaque flux, la source génère un nombre fixé de messages (dont la taille est définie), par période (la taille de l'échéancier). Les feuilles gardent en mémoire les trames correspondantes jusqu'à leur premier intervalle de transmission.

Lors de la configuration logicielle de leurs nœuds, les clients doivent respecter les clauses des *SLA* car la quantité de trafic que chaque nœud client émet a un impact sur la consommation globale d'énergie et sur les communications voisines. L'opérateur doit donc restreindre les garanties de *QoS* au trafic qui est spécifié dans les *SLA*.

Les exigences du client L'opérateur choisit la stratégie d'utilisation des ressources radio en tenant compte des contraintes applicatives des clients. Il faut par exemple prioriser les messages d'une application par rapport à ceux d'une autre afin de respecter un délai. Ces exigences de *QoS* s'expriment par flux sous la forme d'objectifs de niveau de service, les *SLO* [**?**]. Comme les opérateurs de télécommunications, nous nous intéressons majoritairement à deux indicateurs clés de performance (*KPI*) [**?**] : le délai de transit des messages de chaque flux sur le réseau de l'opérateur, et le taux de livraison des mêmes messages de bout-en-bout.

4.2.2 Le réseau radio multi-saut de l'opérateur

Afin de se démarquer en fournissant une connectivité garantie et des performances, l'opérateur doit implémenter des solutions sans fil multi-sauts. Il met en place une infrastructure réseau adaptée aux différents flux applicatifs.

L'organisation du réseau de l'opérateur L'opérateur installe des nœuds relais permettant de construire des chemins multi-sauts en assurant une couverture radio à grande échelle. Les relais jouent deux rôles :

1. celui de routeur d'entrée permettant de connecter les nœuds feuilles des clients à l'architecture et de collecter le trafic applicatif des clients ;
2. celui de routeur intermédiaire permettant d'acheminer les trames saut-par-saut vers les points de sortie du réseau de l'opérateur, pour acheminer le trafic des clients.

Enfin, l'opérateur met en place des nœuds définis comme points de sortie du réseau. Points de convergence des flux applicatifs, ceux-ci constituent un ensemble de passerelles vers les systèmes d'information de l'opérateur et des clients. Les solutions de communication entre les passerelles et les systèmes d'information des clients se construisent sur des réseaux plus conventionnels notamment filaires. Elles ne font donc pas l'objet de notre étude, mais de nombreuses solutions existent (e.g. *VPN* sur *IPv6*, transport des données basé sur *MPLS*). Les passerelles collectent le trafic et les informations supplémentaires qui leur arrivent (e.g. les informations d'observation du réseau).

La Fig. 4.2 présente une portion de topologie comprenant trois nœuds feuilles, trois relais et une passerelle. Les flux applicatifs provenant des différentes sources y empruntent des chemins variés. Les nœuds clients peuvent être mis en place par le client lui même (e.g. des capteurs de télérelève de gaz sont installés chez l'usager) ou par l'opérateur. L'opérateur

doit adapter le positionnement et le nombre de ses relais à leur distribution géographique, leurs conditions radio et leur nombre. Il peut par exemple installer un nouveau relais pour relier les nœuds feuilles d'une rue, ou d'un immeuble.

La mutualisation de l'infrastructure L'opérateur doit prévoir son architecture pour un fonctionnement de longue durée (e.g. 10 ans). En amont de la signature de *SLA*, le premier déploiement de relais doit prendre en compte la disposition des lieux et des potentielles sources de futur trafic [**?**], afin de couvrir les besoins des premiers clients avec qui l'opérateur s'engage. Le déploiement des relais est assujetti aux contraintes de l'environnement (e.g. difficulté d'accès, espace réduit). Nous considérons donc des relais alimentés de manière autonome par pile.

Lorsqu'un nouveau client s'accorde avec l'opérateur, il doit indiquer la distribution géographique de ses feuilles et le trafic maximum attendu. La position des nœuds feuilles de chaque client est donc connue et statique. Ayant connaissance de l'environnement (e.g. le bâti à l'échelle d'une ville), l'opérateur peut alors, de manière centralisée, choisir de déployer de nouveaux relais à des points clés [**?**] ou de réorganiser le plan de données du réseau afin de rattacher les nouvelles sources des flux des clients. En effet, cette connaissance des conditions permet à l'opérateur d'avoir une idée de quelles ressources vont être utilisées, en mettant ces informations en relation avec celles issues des mécanismes de routage (e.g. le rang des nœuds relais dans *RPL* [**?**]). Il peut ainsi prolonger la durée de vie de son infrastructure [**?**].

De plus, du fait de l'augmentation de la demande de trafic et de la difficulté à déployer de multiples réseaux superposés, l'utilisation d'une infrastructure partagée prend tout son sens ici puisque choisir le positionnement des relais permet d'en partager l'usage pour être plus efficace en énergie et en limiter le nombre et le coût.

Nous pensons que l'opérateur réduira ses coûts en partageant son infrastructure pour plusieurs clients. En effet, un opérateur qui fournit de la connectivité radio à différents capteurs peut quantifier l'usage des ressources pour établir des stratégies commerciales contractualisées. Dans le but de gérer les trafics applicatifs clients, ce partage nécessite cependant la mise en place d'une architecture qui inclue toutes les fonctionnalités nécessaires de gestion du réseau. En particulier, les clients doivent être informés du respect des garanties en termes de quantité de ressources allouées, de fiabilité et de délai.

4.3 Un aperçu de l'architecture

Nous introduisons une nouvelle architecture de *SLA* opérée, dans laquelle l'opérateur de réseau de l'*IoT* assure l'observation, la gestion des ressources et l'offre de *SLA* personnalisés pour chaque client. L'opérateur est capable de détecter, d'anticiper et de corriger les potentielles variations de performance. Les usages non contractualisés des ressources (e.g. une feuille génère plus de messages que prévu et surcharge l'environnement radio, ou bien un nœud non identifié dans le *SLA* se fait passer pour une feuille connue) doivent être contrôlés afin de préserver les autres flux clients et l'énergie des nœuds.

4.3.1 Contexte et problème

La tâche de l'opérateur est rendue plus difficile par l'utilisation de la radio. Le médium radio est partagé et évolue suivant les dimensions temporelle, fréquentielle, spatiale. Il doit aussi prendre en considération les contraintes liées aux nœuds (énergie, mémoire, capacité de calcul). Nous surveillons les fréquents changements qui surviennent sur le réseau par des

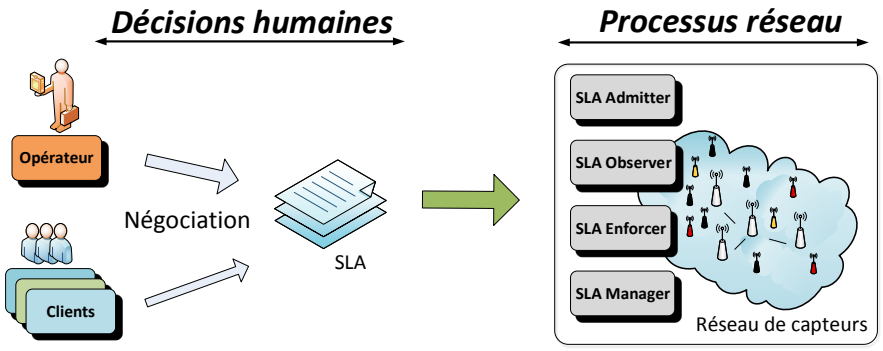


FIGURE 4.3 – Vue globale de l'architecture de *SLA* pour réseaux de l'*IoT* opérés

mécanismes d'observation adaptés (i.e. distribués sur des nœuds contraints). La gestion des *SLA* nécessite des fonctionnalités comme le contrôle d'admission (la décision d'accepter ou non un nouveau flux sur un nouveau contrat), l'observation de la *QoS*, la configuration des nœuds. En multi-saut radio, l'observation, l'analyse de *QoS* et la configuration à distance sont plus difficiles car les fonctionnalités sont distribuées sur les nœuds et dépendent de toutes les couches protocolaires en même temps.

De plus, les nœuds sont contraints en énergie. La charge de trafic participe à leur consommation d'énergie et leur durée de vie. Quand un opérateur de réseau de l'*IoT* connecte de nouveaux nœuds clients il doit être attentif à équilibrer la charge de trafic : il faut en effet éviter la perte totale et prématurée d'énergie à la fois sur ses nœuds et ceux de ses clients.

L'opérateur doit chercher à résoudre les problématiques suivantes :

1. Que se passe-t-il quand un nouveau client se présente et souhaite connecter de nouveaux capteurs ?
2. Quelle quantité de trafic sont capables de supporter les relais formant le réseau de l'opérateur ?

Pour répondre à ces questions, nous proposons que l'opérateur mette en fonctionnement une infrastructure qui collecte des informations de performance et de capacité courante du réseau, afin de lui permettre de décider d'admettre ou non de nouveaux flux clients. Sans cette collecte, l'opérateur pourrait admettre un nouveau client sans que les ressources nécessaires soient disponibles : la *QoS* serait dégradée pour tous les clients, et la durée de vie des relais diminuerait, l'exposant à une violation des *SLA*.

L'opérateur fait face à deux défis :

1. couvrir la plus grande zone possible ;
2. densifier la couverture sur les zones à fort trafic.

La stratégie de l'opérateur consiste donc à faire un compromis entre le nombre et la charge de ses clients. Les choix au moment du déploiement initial sont cruciaux car les déploiements supplémentaires sont coûteux.

Lorsque de nouveaux nœuds relais sont nécessaires à la connexion d'un nouveau client, l'opérateur doit avoir accès à toute l'information qui pourrait faciliter la décision d'un ingénieur : quel est le nombre de nœuds supplémentaires devant être déployés, à quel endroit doivent ils être placés, quelle périodicité du trafic est requise, etc. Dans ce but, l'architecture permet à l'opérateur de comparer les performances observées du réseau avec les besoins exprimés dans les *SLA*.

Enfin, de nouvelles routes et configurations doivent être automatiquement calculées en fonction des nouvelles exigences de *QoS*, d'une part, et de l'énergie restant dans les nœuds d'autre part (celle ci doit être surveillée pour éviter les pannes, et les relais doivent être remplacés avant rupture de service). L'architecture permet cette analyse.

Dans ce scénario, l'architecture introduite en Fig. 4.3 et détaillée en Fig. 4.4, permet donc à l'opérateur de répondre aux besoins que nous avons décrits, en complétant les propositions existantes, comme *VITRO* [**?**] ou *Octopus* [**?**].

Par la suite, nous proposons une description et la mise en œuvre de notre architecture de gestion de *SLA* pour réseaux de l'*IoT* opérés, illustrée Fig. 4.4 :

1. nous définissons les entités de base et leurs interactions, qui sont nécessaires pour supporter les *SLA* dans le contexte des réseaux de l'*IoT* ;
2. nous expliquons comment s'occuper de l'admission potentielle de nouveaux flux sur le réseau. Nous donnons les heuristiques associées ;
3. nous discutons la mise en place de l'observation (ou *monitoring*) en temps réel et de la manière de structurer et de conserver les mesures ;
4. nous distinguons les ensembles dépendant de la technologie de ceux qui sont génériques ; nous fournissons des exemples de mise en œuvre, dans le contexte ouvert et standard de *6TiSCH*.

4.3.2 Les étapes de l'architecture de *SLA* et leurs objectifs

La Fig. 4.4 montre l'architecture de *SLA* que nous proposons : celle-ci est divisée en trois parties, chacune correspondant à une dimension temporelle différente :

1. la partie humaine, qui consiste à (re)-négocier les termes des *SLA* ;
2. la partie de contrôle hors ligne qui se charge des décisions d'admission de nouveaux services sur le réseau ;
3. et la partie exécutive (le contrôle en ligne) qui permet de gérer les processus courants sur le réseau : mise à jour des paramètres de configuration et maintien de la *QoS*.

De cette manière, l'architecture de *SLA* prend en compte les différents aspects de la gestion de *SLA*. La partie gauche de la Fig. 4.4 correspond à des changements hebdomadaires ou mensuels dans les décisions humaines. La partie de droite est réactive à des changements en temps très courts sur le réseau de l'opérateur.

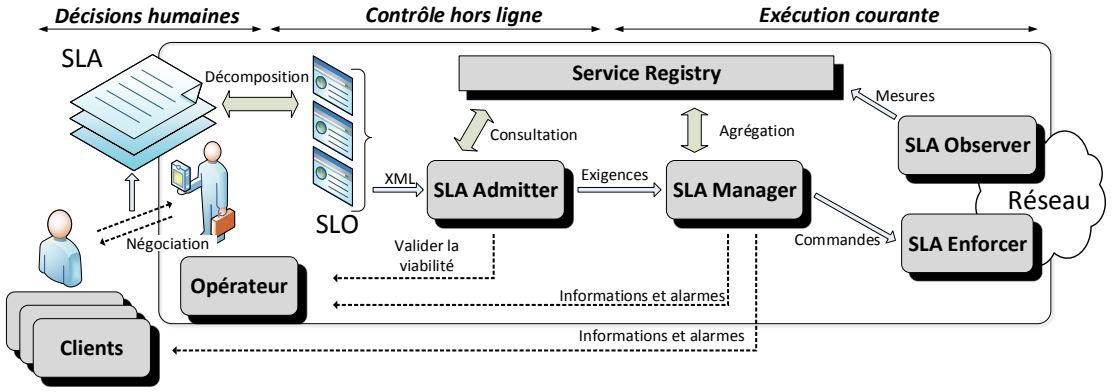


FIGURE 4.4 – L'architecture de *SLA* pour réseaux de l'*IoT* opérés : détail des entités et leurs interactions.

Étape 1 : les décisions humaines La négociation d'un *SLA* entre l'opérateur et un client est la première étape du processus. Les clauses du *SLA* sont spécifiées comme suit (voir les détails au Chapitre 3) :

1. le client décrit ses besoins au niveau applicatif sous la forme de *KPI* ;
2. il fournit la position de ses nœuds feuilles et le trafic maximum attendu ;
3. il spécifie la durée du contrat et les périodes d'application des *KPI*.

L'opérateur intègre alors le *SLA* dans son système d'information. La décomposition du *SLA* en *SLO* est décrite au chapitre précédent. L'ensemble des *SLO* mis en place est alors transmis au ***SLA Admitter***.

Étape 2 : le processus d'admission Le ***SLA Admitter*** extrait les métriques complexes formant les *SLO*, leur combinaison et leur période de validité. Le ***SLA Admitter*** évalue la quantité de ressources requises pour chaque *SLA*. Puis, il consulte l'information d'état du réseau stockée dans le ***Service Registry*** (la base de données des mesures de performance du réseau, voir les détails Section 4.4.5). Il compare la disponibilité courante du réseau avec les besoins globaux en ressources, y compris les exigences des *SLA* préalablement admis. Il conclue et informe l'opérateur de la viabilité du nouvel *SLA* en question.

Si la configuration courante du réseau ne remplit pas les conditions nécessaires à la remontée des nouveaux trafics, le réseau lui-même peut changer en adoptant de nouveaux paramètres (e.g. un nouveau routage) afin de mieux équilibrer les charges et rendre l'admission possible.

Si le ***SLA Admitter*** rejette un nouvel *SLA*, il faut revenir à la première étape (humaine) de l'architecture de *SLA* de l'opérateur : un gestionnaire prendra les décisions appropriées (e.g. rendre l'admission possible en déployant de nouveaux relais, ou en négociant à la baisse les exigences attendues). En cas de succès, les nouvelles demandes de ressources sont transférées à l'entité suivante, le ***SLA Manager***, chargé d'intégrer ce nouvel *SLA* à l'analyse et au maintien des performances du réseau.

Étape 3 : intégration dans le processus courant Lorsqu'un nouvel *SLA* est admis, le ***SLA Manager*** ajoute les exigences des nouveaux flux aux exigences qu'il connaît des autres *SLA* déjà en application. Il utilise cette information pour son analyse temps-réel des performances. Cette analyse permet au ***SLA Manager*** de formuler des instructions qu'il envoie au ***SLA Enforcer*** (entité chargée de la configuration du réseau, voir les détails Section 4.4.3). Une instruction indique par exemple l'allocation périodique de ressources de communication pour une application donnée et un ensemble de nœuds sources.

Le ***Service Registry*** est actualisé par les observations du ***SLA Observer*** (entité responsable de l'observation des performances du réseau, voir Section 4.4.4). Le ***SLA Manager*** consulte le ***Service Registry*** en temps réel afin d'obtenir une information à jour sur l'état du réseau. L'analyse du ***SLA Manager*** consiste à :

- regrouper les mesures brutes (e.g. le nombre de trames reçues d'une feuille pendant une période donnée) entre elles pour former des *métriques complexes* (e.g. le taux de livraison global moyen d'une application pendant une période donnée) ;
- comparer les métriques complexes avec les exigences de chaque flux ;
- construire les instructions à donner aux nœuds.

Le ***SLA Manager*** transmet à l'opérateur et aux clients certaines métriques complexes, qui sont spécifiées sous la forme de *KPI* dans les *SLO*. Cette étape permet aux acteurs de vérifier que le *SLA* est correctement appliqué. Si ce n'est pas le cas, le *SLA* est dit *dégradé* : l'opérateur dispose d'une période fixée dans le *SLA*, lui permettant d'agir sur la

configuration de réseau afin de rétablir un service normal. Les pénalités correspondant à la violation des conditions par l'opérateur sont également définies dans le *SLA*.

Enfin, le ***SLA Enforcer*** met à jour en temps réel la configuration des nœuds, suivant les instructions que le ***SLA Manager*** lui transmet.

Parmi les trois étapes de l'architecture, nous venons d'introduire cinq entités fonctionnelles, chacune correspondant à un rôle spécifique, et interagissant avec les autres par le biais d'interfaces particulières.

Nous décrivons maintenant les spécificités de chaque entité. Les technologies utilisées doivent être robustes aux changements de l'environnement radio afin de pouvoir garantir des niveaux de service durablement. Nous donnons une possible mise en œuvre de chaque entité dans le contexte de *6TiSCH* [**?**].

4.4 Les entités fonctionnelles de l'architecture de *SLA* de l'opérateur

Dans cette section, nous explorons maintenant les détails de l'architecture de *SLA* et les descriptions formelles des différentes entités fonctionnelles et de leurs interfaces. Nous étudions quels mécanismes adopter depuis l'expression des besoins des clients jusqu'à leur mise en œuvre sur le réseau. Nous identifions les problèmes techniques et scientifiques qui concernent la mise en place d'une architecture de *SLA* pour réseaux de l'*IoT* opérés. Nous illustrons également l'architecture de *SLA* proposée dans le cadre de *6TiSCH* et *IEEE Std 802.15.4-2015* [**?**].

4.4.1 Le *SLA Admitter* : contrôle d'admission pour les nouveaux clients

Le ***SLA Admitter*** (Fig. 4.5) prend le rôle d'interface entre les décisions humaines et l'architecture de *SLA* de l'opérateur. Il traite les demandes de mise en place des nouveaux *SLA*, en analysant leur viabilité, au regard de l'état courant du réseau de l'opérateur. Ainsi, il empêche la mise en place de *SLA* qui mettraient en péril l'architecture de *SLA* de l'opérateur.

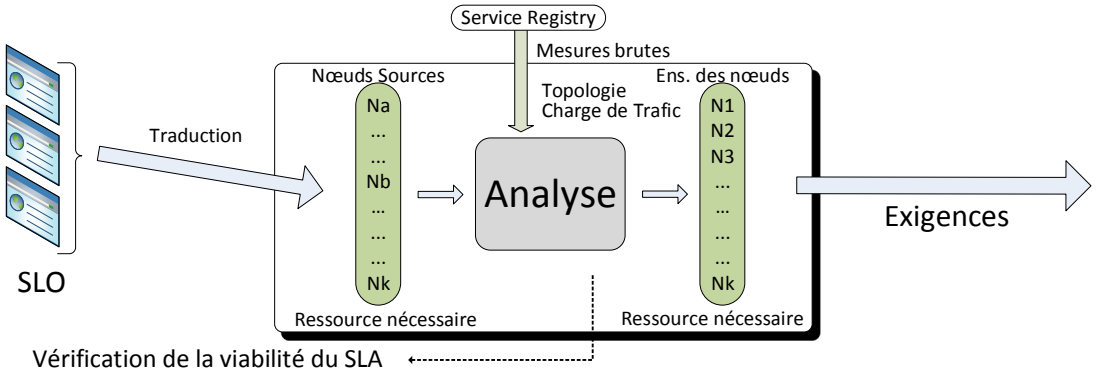


FIGURE 4.5 – Le ***SLA Admitter***.

Le ***SLA Admitter*** s'occupe d'abord de la transcription des *SLO* en termes de besoins en ressources pour chaque nœud relais. Cette transcription inclut :

— l'ensemble des nœuds sources du client à relier à l'opérateur ;
— la quantité maximale de trafic pour laquelle le nouvel *SLA* spécifie des garanties de *QoS* ;

— les exigences de délai des nouveaux flux clients ;
— le cadre temporel d'application des garanties (e.g. les heures ouvrées) et leurs durées.

Une fois la forme de trafic déterminée, le ***SLA Admitter*** consulte le ***Service Registry***(voir détails Section 4.4.5). Sur la base des mesures brutes, il construit l'estimation de l'état courant de réseau :

— la capacité résiduelle sur les nœuds de l'opérateur ;
— les informations de topologie physique (e.g. le graphe de connectivité du réseau) ;
— le délai induit sur chaque route ;
— les variations de ces informations dans le cadre temporel des métriques considérées.

L'analyse doit permettre à l'opérateur de prédire l'impact des nouveaux flux sur les nœuds (en termes d'énergie, d'occupation mémoire, etc.). L'analyse doit considérer des changements possibles sur la configuration du réseau (e.g. de nouveaux chemins, de l'équilibrage de charge sur les relais, etc.).

La granularité de chaque *SLO* peut varier selon la stratégie de coût de l'opérateur. De plus, le choix de la technologie sans fil a un impact sur la précision et la forme des mesures brutes. Pour ces raisons, si la fonction de ***SLA Admitter*** est générique, sa mise en œuvre concrète dépend de la technologie.

Dans les cas où l'analyse empêche la mise en place d'un nouvel *SLA*, le ***SLA Admitter*** fournit un rapport expliquant le rejet du *SLA* à l'opérateur. Cela lui permet de déterminer les problèmes qui ont rendu l'admission impossible, et ce qui peut être réalisé pour modifier la situation et favoriser l'admission (e.g. par le déploiement de nouveaux relais ou en modifiant les *SLO*).

Sinon, le ***SLA Admitter*** accepte une demande de mise en place de *SLA* en transmettant les exigences correspondantes au ***SLA Manager***. En effet, ces informations font partie du résultat de l'analyse du ***SLA Admitter***, et servent à l'organisation des communications au niveau du ***SLA Manager***.

Finalement, le contexte de *6TiSCH* simplifie le processus d'admission. La connaissance des échéanciers des nœuds simplifie les prévisions de *QoS*, parce qu'elle donne une vision claire des ressources disponibles. On a aussi une estimation précise de la consommation d'énergie liée à l'utilisation de ces ressources.

4.4.2 Le *SLA Manager* : maintien et synthèse de l'état du réseau de l'*IoT*

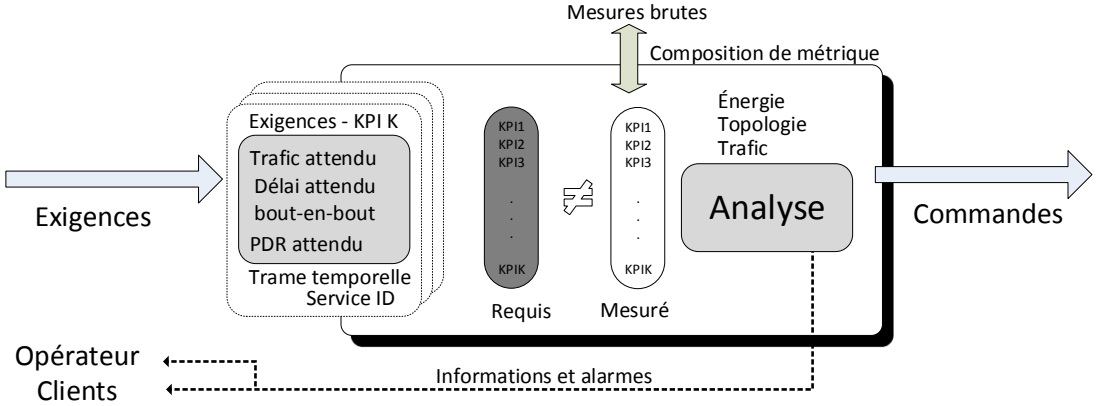


FIGURE 4.6 – Le ***SLA Manager***.

Le ***SLA Manager*** représente l'intelligence du système. Il se situe à l'interface entre la partie de contrôle hors ligne et la partie exécutive de l'architecture de *SLA* de l'opérateur.

Interface avec le *SLA Admitter* Le ***SLA Manager*** obtient comme paramètres d'entrée pour chaque source et chaque flux client entrant sur le réseau les exigences en ressources réseau :

- la charge de trafic maximum attendue ;
- les exigences de *QoS* (e.g. le délai maximal de bout-en-bout pour les messages d'un capteurs sur un nœud feuille donné) ;
- le cadre temporel d'application (i.e. les bornes entre lesquelles les exigences de *QoS* s'appliquent).

Ces exigences sont traduites par flux à partir des *SLA*, dont la spécification est détaillée au Chapitre 3. Le ***SLA Manager*** analyse l'état du réseau à partir des paramètres d'entrée et des informations brutes issues de l'observation.

Analyse des performances courantes Le ***SLA Manager*** collecte les mesures brutes du ***Service Registry***, et les agrège pour former des métriques complexes qui correspondent à l'ensemble des paramètres attendus (les exigences transcrites en ressources). La difficulté consiste à traiter des données provenant de sources multiples, parfois incomplètes (perte de trames contenant des informations d'observation), redondantes ou agrégées.

Par exemple, il évalue le délai de bout-en-bout moyen des trames transitant sur le réseau pendant une période donnée à partir des valeurs de délai pour chaque trame collectée, ou pour les trames collectées de chaque source.

Ensuite, le ***SLA Manager*** compare l'état mesuré du réseau (au travers des métriques complexes) avec l'ensemble des exigences de *QoS* : si celles-ci ne sont pas respectées, le *SLA* correspondant est considéré comme dégradé (l'opérateur dispose d'un temps donné pour rétablir le service, avant que le *SLA* soit rompu).

Avec une telle analyse, le ***SLA Manager*** peut de manière autonome changer la configuration requise du réseau, dans le but de résoudre l'incident. Si aucun paramètre ne peut être modifié avec succès (résolution de l'incident), il déclenche une alarme qui permet une prise de décision humaine (intervention manuelle, re-négociation du *SLA*, etc.). Les mécanismes de mise en place de la *QoS* dans les réseaux de capteurs font encore l'objet de recherches, notamment sur la manière d'allouer des ressources en tenant compte des variations des conditions radios [**?**].

Le ***SLA Manager*** doit aussi surveiller le niveau d'énergie des nœuds relais. Il alarme l'opérateur quand c'est nécessaire (lorsque qu'un certain seuil est franchi).

Les rapports d'état périodiques En plus des alarmes, le ***SLA Manager*** rend compte des *KPI* : il envoie les valeurs de certaines métriques complexes aux clients et à l'opérateur, afin de leur apporter des points de vue spécifiques sur le comportement, ou l'état du réseau. Par exemple, un client peut avoir besoin de superviser le délai applicatif des trames sur certains nœuds, dans le but de surveiller des paramètres non contractuels, d'évaluer ses possibilités d'amélioration du service, ou pour diagnostiquer des dysfonctionnements à certains endroits critiques.

La commande d'action Le ***SLA Manager*** déclenche des changements dans la configuration (dépendante de la technologie) des relais de l'opérateur, par exemple :

```
Donner aux nœuds [25-35] 2 opportunités/jour pour transmettre 200 octets
au point de collecte C, en moins de 300 s.
```

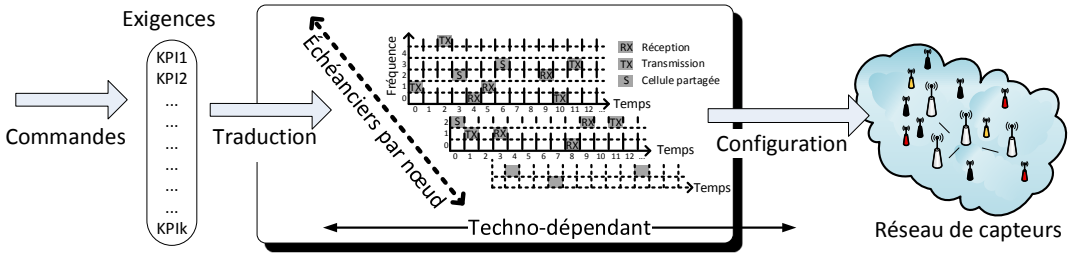


FIGURE 4.7 – Le ***SLA Enforcer***.

L'ensemble générique des instructions inclut :

— l'identifiant des nœuds concernés par ce changement ;
— les caractéristiques génériques du changement ;

Le format d'échange de messages entre le ***SLA Manager*** et le ***SLA Enforcer*** dépend de la nature de ce dernier (e.g. *PCEP*).

4.4.3 Le *SLA Enforcer* : comment configurer le réseau

Le ***SLA Enforcer*** contrôle la configuration en cours : il fournit à chaque nœud les changements à effectuer (e.g. sur le routage ou l'accès au médium), en traduisant les instructions du ***SLA Manager***. Cette entité est en conséquence dépendante de la technologie.

Alors que le ***SLA Manager*** donne des exigences génériques, le ***SLA Enforcer*** les traduit en instructions qui dépendent des protocoles employés. Par exemple, le ***SLA Enforcer*** construira ou modifiera le graphe de routage (e.g. un *DODAG* [**?**]) à partir de certaines métriques directement traduites des exigences du ***SLA Manager***. Ces changements peuvent avoir un impact sur un relais spécifique, un groupe de nœuds, ou encore le réseau entier de l'opérateur.

La Fig. 4.7 montre la mise en œuvre du ***SLA Enforcer*** dans le cadre de *6TiSCH* [**?**]. Les instructions sont traduites en échéanciers (ou *schedule*) du standard *IEEE Std 802.15.4-2015 mode TSCH*, puis ceux-ci sont transmis à chaque nœud.

Le ***SLA Enforcer*** peut être mis en place de deux façons :

— une entité de contrôle **centrale** (e.g. un *Path Computation Element* (*PCE*) [**?**]) génère l'échéancier de cellules à allouer (l'algorithme d'ordonnancement étant par exemple le *Traffic-Aware Scheduling Algorithm (TASA)* [**?**]) en prenant en compte les exigences de service ;
— dans le fonctionnement même du protocole de routage *RPL*, les points de sortie initient des mises à jour **distribuées** de la topologie de routage.

On associe ainsi à chaque flux, selon la *QoS* attendue, un ensemble de cellules dédiées. L'opérateur peut ajouter une certaine flexibilité (non garantie) en mettant en place des cellules partagées supplémentaires.

4.4.4 Le *SLA Observer* : l'observation de *QoS* sur les réseaux multi-sauts de l'*IoT*

Afin de gérer le réseau, il faudra que l'opérateur collecte des informations précises sur la performance des nœuds. Le ***SLA Observer*** (Fig. 4.8) a pour objectif de mesurer ces informations et de les transmettre au ***Service Registry***.

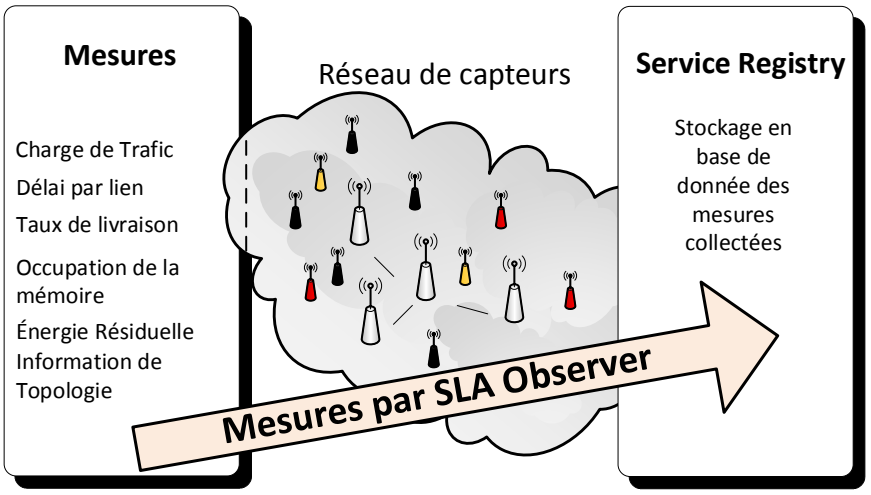


FIGURE 4.8 – Le ***SLA Observer***.

Lors de la mise en place d'un tel algorithme d'observation pour réseau de l'*IoT*, un compromis doit être trouvé entre la consommation d'énergie et la précision de la mesure. En effet, le trafic client est assez hétérogène (les applications peuvent générer d'une transmission par jour à plusieurs trames par heure [**?**]), donc la quantité d'information de mesure peut dépasser la quantité de données sur certains relais. Ainsi, les algorithmes de mesure doivent être correctement dimensionnés (ajustés aux demandes des clients exprimées dans les *SLA*) afin d'en réduire le surcoût en énergie.

Que mesurer ? Les mesures brutes sont construites localement sur chaque nœud. Le ***SLA Observer*** dépend donc de la technologie présente sur le nœud. Par exemple, tous les nœuds n'ont pas la même manière de mesurer la qualité des liens radio. Les indicateurs (*Link Quality Indicators (LQI)*) diffèrent, ainsi que leur analyse : le taux de livraison (*Packet Delivery Ratio (*PDR*)*) peut être calculé de plusieurs manières (e.g. moyenne sur une période fixée, ou estimation à partir des mesures précédentes) [**?**]. Les mesures sont effectuées à différentes couches protocolaires (e.g. le délai au niveau de l'accès au médium (voisinage à un saut), la topologie de routage, l'état d'occupation de la mémoire des nœuds).

L'architecture de *SLA* de l'opérateur s'appuie sur les métriques génériques suivantes (définies localement ou de bout-en-bout) :

— le délai ;
— le taux de perte de trames ;
— l'énergie résiduelle ;
— la qualité de liens ;
— la charge de trafic.

En effet, elles permettent à l'opérateur de contrôler la validité des *SLA* et la durée de vie du réseau. D'autres métriques peuvent être considérées pour des besoins spécifiques (e.g. l'intégrité pour des besoins de sécurité).

Comment mesurer ? L'opérateur doit choisir pour chaque mesure s'il l'effectue de manière périodique ou de manière occasionnelle.

1. Les mesures périodiques permettent une mise à jour régulière de la base de données. Le choix de la période d'échantillonnage de la mesure a une influence sur sa granularité : une période longue (e.g. une mesure toutes les heures) est moins coûteuse en énergie qu'un temps court, mais donne une information moins détaillée.
2. Dans le cas où les exigences des applications sont fortes, la période de mesure doit être courte. La qualité de l'observation de *QoS* est un critère de différenciation entre les opérateurs : ils devront effectuer une observation suffisamment précise de leur réseau pour fournir une meilleure information à leurs clients.
3. D'autres mesures ne doivent être réalisées qu'à des occasions spécifiques (e.g. lorsque la couche applicative requiert la valeur d'un paramètre spécifique) ou selon les mesures précédentes (e.g. en réaction au franchissement d'une valeur seuil, ou si aucun changement n'a eu lieu pendant longtemps).

La stratégie d'observation adoptée est donc essentielle. Le surcoût en termes de communication et donc d'énergie doit être pris en considération par l'opérateur lorsqu'il propose des *SLA*.

Comment les mesures sont-elles collectées ? Le ***SLA Observer*** doit garder les mesures brutes dans le ***Service Registry***. Selon la technologie, il peut :

1. utiliser un canal dédié de communication (la mesure est alors dite *out-of-band*), avec une pile protocolaire indépendante du plan de données. Cela nécessite que le nœud supporte plusieurs technologies. Par exemple, dans les plates-formes de test *SensOrLab* [**?**] ou *FIT/IoT-Lab* [**?**], une interface de supervision, par lien série (*FIT/IoT-Lab*) ou 4G (*SensOrLab*), est prévue pour les besoins des expérimentations. Les mesures pourraient être collectées sur des interfaces génériques ou standards [**?**], dans des messages de protocoles existants (e.g. *CoAP* [**?**]) ;
2. utiliser une observation passive : aucun trafic explicite n'est généré sur le réseau de l'opérateur, un serveur dans le système de l'opérateur extrait de manière centrale les informations inférées des trames en transit ;
3. utiliser une observation active : envoyer des trames dédiées à l'observation, ou bien encapsuler les informations d'observation sur les trames de données. Dans le deuxième cas, l'opérateur devra potentiellement inspecter les trames de données des clients, et créer ainsi des problèmes de confidentialité des données ;
4. utiliser un des différents modes de collecte des mesures : *sur demande* (avec une requête spécifique), ou de manière *périodique*. Par exemple, le mode *DSME* de *IEEE Std 802.15.4-2015* rend possible une remontée périodique de l'état des liens radio (les messages sont nommés *periodic link status reports*) vers une entité centrale ;
5. utiliser une observation hybride : dans le but d'économiser de la bande passante, le ***SLA Observer*** peut utiliser une observation passive pour les activités normales, et réactive avec des trames dédiées en cas d'incidents ou pour des besoins applicatifs spécifiques.

La collecte de mesure active a un impact sur la consommation de ressources et d'énergie de l'architecture. Elle est donc à favoriser dans le cas de mesures nécessaires et qu'on ne peut pas collecter passivement (e.g. le niveau d'énergie critique d'un nœud intermédiaire).

Application au contexte de *6TiSCH* Dans le contexte de *6TiSCH*, la fonction d'observation est distribuée sur chaque nœud : chaque nœud transmet les mesures brutes vers

le ***Service Registry***, en utilisant un protocole d'échange de l'information d'observation, e.g. *CoAP* [**?**]. Les métriques de routage utilisées par *RPL* constituent également une information que l'on peut collecter (e.g. en créant des trames dédiées d'observation remontant le *DODAG*).

Les informations d'observation peuvent être intégrées à l'échéancier de communication : l'opérateur peut décider d'utiliser un nombre donné de cellules pour collecter les mesures.

4.4.5 Le *Service Registry* : une base de données pour les mesures brutes

Le ***Service Registry*** contient toutes les informations concernant les performances du réseau. Les mesures brutes produites par le ***SLA Observer*** ne peuvent pas être gardées localement dans chaque nœud puisque ceux-ci ont une mémoire limitée (ils peuvent seulement garder l'information courante). De plus, les mesures brutes doivent pouvoir être consultées par les autres entités de l'architecture (***SLA Manager***, ***SLA Admitter***) sans encombrer les ressources du réseau. Donc le ***Service Registry*** doit être centralisé (e.g. une base de données attachée au SI de l'opérateur).

L'opérateur doit garder en mémoire les mesure collectées pendant toute la durée de validité du *SLA*. Ainsi, si l'opérateur change d'algorithme pour une de ses entités fonctionnelles, de nouvelles décisions peuvent être prises en gardant en considération tout l'historique du réseau.

Étant donné que l'observation concerne un vaste nombre de nœuds pendant de longues périodes, les mesures doivent être identifiées et classées de manière univoque. L'observation périodique de *QoS* requiert la synchronisation globale des nœuds. Nous proposons le format suivant pour les métriques brutes :

- ***timestamp*** : l'instant de génération de la mesure ;
- ***measurer (id)*** : le nœud qui génère la mesure ;
- ***entity (ids)*** : les objets concernés par la mesure (e.g. lien) ;
- ***type*** : une unité abstraite (voir Section 4.4.4) ;
- ***mode*** : comment a été réalisée la mesure (voir Section 4.4.4) ;
- ***value*** : la mesure en elle-même.

La génération d'un estampillage, horodatage, ou ***timestamp*** est difficile car les mesures sont réalisées localement sur des nœuds distribués, disposant d'horloges autonomes susceptibles de se décaler [**?**]. Si les nœuds sont synchronisés (e.g. avec *IEEE Std 802.15.4-2015 mode TSCH*), les décalages d'horloges sont compensés : au prix d'un surcoût énergétique de maintien de la synchronisation, la précision des mesures est améliorée. Une observation passive réalisée seulement en sortie du réseau radio multi-saut, sans requérir un mécanisme de synchronisation, offrirait des mesures moins précises.

Si l'opérateur met à jour le ***SLA Observer***, de nouveaux types de mesures peuvent apparaître si nécessaire. Nous donnons ici un exemple plus détaillé du format d'échange des mesures brutes, dans le formalisme basé sur *XML* et explicité par le chapitre précédent :

```
<MeasureRequest ID="12478">
  <Timestamp>34567890</Timestamp>
  <Measurer>
    <RelayNode> 46 </RelayNode>
  </Measurer>
  <Measured>
    <RelayNode> 46 </RelayNode>
    <RelayNode> 47 </RelayNode>
  </Measured>
  <Type>
    <Layer> MAC </Layer>
    <Range> SingleHop </Range>
    <metric> Delay </metric>
```

```
14    </Type>
15    <Mode name ="Periodic">
16      <Period> 60 </Period>
17    </Mode>
18    <Value>0.37</Value>
19  </MeasureRequest>
```

Listing 4.1 – Format d'échange de mesures brutes (*XML*).

Cet exemple correspond à une mesure du délai effectuée par un nœud relais (le 46), sur son lien avec un voisin (le 47). Celle-ci est effectuée périodiquement (le champ `Period` a une valeur de 60 s). Nous verrons par la suite les règles plus précises de spécification que nous recommandons.

4.5 Synthèse du chapitre

L'utilisation de l'*IoT* se développe au sein des entreprises et des collectivités qui offrent des services aux usagers (de distribution d'eau, de gaz, *Smart Parking*, pollution, éclairage public, etc.). Des acteurs spécifiques, les opérateurs de réseaux de l'*IoT*, offriront de la connectivité radio, partagée entre plusieurs clients. La mise en place d'une topologie de relais et de chemins multi-sauts permet de répondre efficacement aux besoins croissants de connectivité.

Les opérateurs feront des économies sur l'énergie et le coût de déploiement en mutualisant une même architecture pour plusieurs clients, et en leur proposant des garanties de qualité de service. Le cadre des *SLA* leur permettra d'identifier les clauses contractuelles sur lesquelles s'engager avec chaque client. Les engagements sur les *SLA* permettent de contrôler des contrats spécifiques et rendent donc viable le multi-service avec garanties de *QoS*.

Dans ce chapitre, nous avons proposé une feuille de route pour la construction d'une architecture de gestion de *SLA* pour réseaux de l'*IoT* opérés. Nous distinguons cinq entités fonctionnelles [**?**] :

1. le ***SLA Observer***, qui collecte les mesures de performance du réseau ;
2. le ***Service Registry***, qui garde les mesures sous la forme de données structurées ;
3. le ***SLA Admitter***, qui analyse la possibilité de mise en œuvre des *SLA* ;
4. le ***SLA Manager***, qui analyse l'état du réseau, vérifie l'application des objectifs de service et construit les instructions de maintien de la *QoS* et d'installation de nouveaux flux ;
5. le ***SLA Enforcer***, qui adapte la configuration du réseau aux instructions qui lui sont données.

Nous avons montré comment ces entités peuvent garantir de la *QoS*, et ainsi, maintenir les niveaux de service attendus par les clients. Notre architecture de *SLA* définit les interactions entre ces entités et donne la description des fonctions qu'elles appliquent et mettent en œuvre. Nos exemples sont positionnés dans le contexte du standard robuste *IEEE Std 802.15.4-2015 mode TSCH*.

Cette infrastructure offre de la flexibilité : l'opérateur est capable de gérer la durée de vie des relais en équilibrant le trafic, en fonction des démarrages et des interruptions de *SLA*.

Il rend compte des *KPI* attendus par les clients et par lui-même. L'architecture permet aussi à l'opérateur d'adapter la configuration du réseau (qui dépend de la technologie

choisie) aux changements de l'environnement radio, et le cas échéant, d'alarmer les acteurs sur un état anormal du réseau.

L'architecture doit être adaptée à la pile protocolaire sous-jacente : nous envisageons sa mise en œuvre dans le cadre de *6TiSCH* et une validation par expérimentation. En fonction du niveau d'exigence des applications (e.g. contraintes faibles en délai et débit), l'opérateur peut adopter la même architecture de gestion de *SLA* avec des choix technologiques différents de *6TiSCH* (e.g. *LoRa* [**?**]). L'étude de l'admission, basée sur la prédiction de *QoS*, doit être approfondie. La prise en compte de réseaux hétérogènes dans la gestion des *SLA* est également une piste d'amélioration.

Nous nous attachons dans les chapitres suivants à montrer comment implémenter deux de ces entités :

1. au Chapitre 5, nous proposons un mécanisme de remontée d'information d'observation de collecte des mesures simples de livraison et de délai sur le réseau (***SLA Observer***). Une fois collectées et stockées dans le ***Service Registry***, ces mesures sont exploitées par l'opérateur pour vérifier le respect des *SLA* en cours et en négocier de nouveaux ;
2. au Chapitre 6, nous proposons un algorithme d'allocation de ressources (***SLA Manager*** et ***SLA Enforcer***).

En effet, les fonctions d'observation et d'allocation nous ont paru les plus essentielles à la gestion de réseaux opérés pour l'*IoT*.

Chapitre 5

Observation efficace des réseaux de l'*IoT* opérés

Nous avons vu dans les chapitres précédents comment l'explosion des trafics incite, plutôt que laisser des réseaux séparés interférer entre eux, à adopter une gestion commune à plusieurs applications de plusieurs clients. Il faut ainsi gérer des flux à qualité de service différente.

L'opérateur doit garantir des niveaux minimum de fiabilité et de délai de bout-en-bout, en considérant chaque flux indépendamment des autres. Dans ce chapitre, nous nous intéressons au ***SLA Observer*** décrit au Chapitre 4 : notre objectif est de fournir les outils pour observer et vérifier individuellement les exigences de chaque client.

Nous avons vu que dans les réseaux de l'*IoT* mutualisés, nous distinguons l'opérateur du réseau, et ses différents clients. L'opérateur achemine les données produites par les nœuds feuilles de chaque client via ses nœuds relais.

Dans un contexte d'utilisation de *IEEE Std 802.15.4-2015 mode TSCH*, nous proposons un mécanisme qui recueille des blocs d'information d'observation pour chaque client et chaque application, par *apposition* d'éléments d'information (les *Information Elements* (*IE*) sur les trames de données. Nous montrons que, en utilisant le *piggybacking*, nous réduisons le surcoût en ressources de communication nécessaires à l'observation, par rapport à une approche traditionnelle utilisant des ressources dédiées.

5.1 Problème de la collecte d'observation sur réseaux de l'*IoT* multi-applicatifs

L'observation de réseau (ou *Network Monitoring*) consiste à faire remonter des informations de contrôle, générées par les nœuds du réseau (sur le canal de communication principal, ou hors bande), à un ou plusieurs points centraux du réseau, pour les agréger et analyser. L'opérateur en tire les conclusions sur l'état et les performances du réseau, en fonction des objectifs de l'analyse (e.g. anticiper les pannes).

L'importance croissante de l'*IoT* dans le contexte de la *ville intelligente* rend l'environnement radio instable lorsque les topologies sont denses, c'est-à-dire quand de nombreux usagers utilisent des terminaux connectés [**?**]. On a vu comment un opérateur de réseau de l'*IoT* offre de la connectivité à plusieurs clients indépendants (e.g. une entreprise de gaz et une entreprise de stationnement intelligent (ou *Smart Parking* [**?**])).

Contexte Dans le contexte des réseaux de l'*IoT* opérés, l'opérateur doit prendre en considération la multiplicité des flux transitant sur son réseau : chaque application générant différents flux de trames de différentes sources, l'observation doit être spécifique à chacun.

La mise en place de *SLA* limite la charge de trafic attendue pour chaque flux, sans pour autant en determiner la réalisation exacte. L'observation ne peut pas supposer le trafic connu et déterministe. De plus, les nœuds intermédiaires de l'opérateur génèrent également du trafic (e.g. maintien de la synchronisation). L'observation en elle-même utilise des ressources de communication.

Dans une technologie *FTDMA*, on a vu que des blocs temps-fréquences, ou cellules, permettent de répartir les ressources :

1. ces blocs sont alloués pour une communication entre plusieurs nœuds et l'échéancier ainsi défini est répété périodiquement ;
2. l'allocation de ressources peut être modifiée.

Nous ne faisons pas d'hypothèse générale sur le routage des trames : le multi-chemin est possible, selon la solution choisie par l'opérateur. Les nœuds feuilles, qui génèrent le trafic, ne sont pas toujours configurables par l'opérateur car ils sont parfois mis en place et contrôlés par chaque client. L'observation doit prendre en considération ces incertitudes (e.g. perte d'une trame sur une route donnée, mauvaise configuration d'une feuille n'entraînant pas la responsabilité de l'opérateur).

Enfin, les nœuds sont contraints (alternat de l'interface radio, mémoire limitée, retransmissions limitées, énergie limitée). L'observation doit fonctionner dans ce contexte.

5.2 Objectifs de l'observation des réseaux de l'*IoT* opérés

Les chapitres 3 et 4 montrent comment l'opérateur met en place des *SLA* avec chaque client, c'est-à-dire des accords spécifiques qui garantissent des exigences minimales de délai et de fiabilité pour la collecte de données des applications [**?**].

Dans cet environnement, l'opérateur doit proposer aux clients des garanties de *QoS* par flux. Respecter ces contraintes implique une isolation des flux clients et un traitement différencié selon la qualité de service (*QoS*) exigée. Les clients s'attachent en général aux *KPI* suivants [**?**] :

1. le taux de livraison de bout-en-bout des trames générées en entrée du réseau. C'est le ratio, pour chaque application et chaque nœud feuille, des trames reçues en sortie du réseau de l'opérateur sur celles qui y sont entrés ;
2. le délai de bout-en-bout sur le réseau de l'opérateur, des mêmes trames. C'est le temps de transit des trames depuis le routeur d'entrée jusqu'aux passerelles.

En effet, ces seuls paramètres permettent de spécifier les besoins des clients. L'opérateur peut dimensionner les ressources allouées à chacun à partir de ces deux paramètres.

Pour l'*IoT* opéré, l'observation doit répondre aux objectifs suivants :

1. prouver aux clients le respect des clauses du *SLA* ;
2. permettre à l'opérateur de contrôler la performance de son réseau.

5.2.1 Observer la réalisation de chaque *SLA*

L'opérateur configure son réseau de manière à respecter les engagements pris, en alloant des ressources sur ses relais. Il traite les flux en isolation pour respecter des exigences différenciées en termes de *QoS* et de sécurité des données. Pour prouver que les clauses du

SLA sont bien respectées, il faut mesurer les performances obtenues pour chaque flux. Cela justifie la mise en place de mécanismes d'observation orientés *SLA* (mesure des paramètres de bout-en-bout).

Dans ce chapitre, nous proposons des mécanismes pour surveiller la performance de bout-en-bout des applications des clients. Ces mécanismes permettent à l'opérateur de prouver à tout moment que les exigences des clients sont remplies. Le défi consiste à mesurer efficacement les performances d'un grand nombre d'applications. En effet, chaque client exprime ses propres critères de Qualité de Service (*QoS*), pour chacun de ses nœuds feuilles et applications, et requiert que chaque feuille soit surveillée indépendamment. Par exemple, une entreprise d'éclairage exige que 95 % des trames soient correctement transmises pour chaque feuille.

La difficulté réside dans la satisfaction simultanée de l'ensemble des exigences. Les mécanismes que nous proposons permettent l'observation sur l'ensemble du réseau des flux entrants et ils sont configurables selon les besoins spécifiés (i.e. la fréquence d'observation de chaque application). Ils sont donc pertinents pour une architecture mutualisée multi-client multi-service.

5.2.2 Vérifier et contrôler l'état du réseau

L'opérateur utilise l'observation comme base de la gestion du réseau. L'observation doit répondre aux objectifs suivants :

1. anticiper des problèmes, e.g. l'épuisement de l'énergie d'un nœud ;
2. diagnostiquer des pannes et les localiser, (les identifier comme pannes permanentes ou intermittentes) ;
3. prévoir l'admission de nouveaux clients/flux sur le réseau (e.g. collecte d'informations de routage pour permettre un placement ultérieur de nœuds) ;
4. évaluer la possibilité d'un changement de topologie (e.g. installation de nouveaux relais) ;
5. optimiser les performances du réseau.

En proposant notre contribution, nous nous intéressons principalement à l'observation efficace des paramètres de *SLA*. Les futurs travaux devront se focaliser sur l'observation des performances, en particulier sur la prévision des évolutions du réseau et sur la capacité d'admission, parce qu'elle est indispensable à l'opérateur pour admettre de nouveaux clients et préserver la longévité du réseau.

5.2.3 Un ensemble de métriques à mesurer

Nous détaillons ici les variables ou *métriques* des réseaux de l'*IoT* que l'observation peut mesurer, de quelle manière la mesure est réalisée et dans quel objectif.

Les métriques physiques correspondent aux propriétés du médium dans l'environnement physique de chaque nœud.

La ***qualité du canal*** est définie par la mesure de l'énergie sur un canal fréquentiel donné. Elle permet de quantifier le niveau d'interférences externes sur un canal donné. Il s'agit de la proportion du temps pendant laquelle un canal est occupé par un signal supérieur en énergie à une certaine valeur seuil. En *FTDMA*, la mesure peut être réalisée lorsque sur une cellule donnée, le récepteur détecte de l'énergie sans pouvoir déchiffrer le signal. Mesurer la qualité du canal permet d'éviter son utilisation quand il est occupé par d'autres applications externes.

La mesure des ***interférences internes*** permet d'identifier les utilisations concurrentes du canal par la même technologie. Elles s'expriment sous la forme d'un graphe de conflit qui recense ces concurrences. Elles peuvent être évaluées de plusieurs manières :

1. en corrélant *a posteriori* les pertes subies simultanément dans un voisinage ;
2. en écoutant de manière passive le trafic perçu pendant une période de temps (vision locale du voisinage) ;
3. en utilisant la diffusion active de messages de découverte de voisinage.

Accompagnée d'une analyse centralisée, cette mesure favorise une meilleure allocation de ressources.

Les métriques de nœud correspondent aux paramètres spécifiques au fonctionnement de chaque nœud.

Pour les nœuds qui ne sont pas alimentés par une source externe, l'***énergie résiduelle*** est une métrique significative afin d'anticiper une perte de connectivité liée à l'épuisement de leur batterie. Suivant le matériel, elle peut être :

1. une mesure physique si le nœud est muni du capteur adéquat ;
2. une estimation logicielle à partir d'un compteur d'événements sur le nœud.

Elle est collectée sur dépassement de seuil(s) ou de manière régulière afin d'en estimer les variations.

En dehors du problème d'énergie, un nœud peut être ***défaillant***, c'est-à-dire qu'il ne communique pas avec son voisinage pendant un temps donné. Plusieurs raisons peuvent déclencher cet état (malveillance, intempérie, boucle de code, etc.). La mesure identifie le nœud et l'action à mener (e.g. remplacement, redémarrage logiciel). La mesure peut être effectuée par échange actif et local, déclenché en cas d'absence de trafic.

Les métriques de lien et d'accès au médium correspondent aux variables de la couche d'accès au médium (en particulier *IEEE Std 802.15.4-2015 mode TSCH*).

Le ***nombre de cellules allouées*** permet d'estimer la charge par nœud ou par lien. Ce nombre peut varier :

1. si les décisions d'allocation sont prises de manières distribuées (e.g. ajout de cellules à la demande) ;
2. en centralisé, si le nœud n'a pas correctement reçu les messages de configuration.

L'estimation de la charge permet de prendre des décisions d'admission. Sa mesure est inhérente au protocole utilisé sur chaque nœud (e.g. le protocole *6P* [**?**]). Elle peut être établie de manière globale pour un nœud ou un lien, ou différenciée par flux (c'est-à-dire par *track*).

En relation avec l'usage des cellules allouées, l'***occupation de la mémoire*** des nœuds par des messages reçus de nœuds voisins ou générés sur place permet d'évaluer :

1. le risque de congestion et de pertes par dépassement de mémoire ;
2. le délai subi localement par les trames au passage d'un nœud ;

Le plan de contrôle de chaque nœud maintient ces informations à jour. Il s'agit de la mise en place d'agrégateurs sur chacun des nœuds (moyennes à différentes échelles temporelles, fenêtres fixes et/ou glissantes). Ces mesures peuvent être collectées sur demande, d'une entité centrale ou d'un nœud voisin, afin de préparer un choix d'allocation.

L'estimation de la qualité des liens radio peut être liée à des métriques physiques. En effet, le module radio des nœuds peut intégrer des outils de mesure de ***puissance du***

signal reçu (ou *Received Signal Strength Indicator (RSSI)*), de ***rapport signal à bruit***, ou d'***indicateur de qualité de lien*** (ou *Link Quality Indicator (LQI)*) dont la réalisation dépend du matériel physique [**?**].

Nous avons vu au Chapitre 2 que la qualité des liens s'exprime en comptabilisant les ***pertes de trames par lien***. Le *PER* (ratio du nombre de trames non reçues sur le nombre de trames envoyées, sur un lien donné) peut s'exprimer globalement, par flux, ou de manière spécifique à chaque cellule (la division temporelle y est prise en compte, tandis que la division fréquentielle est *lissée* par le *Channel Hopping*). L'évaluation peut être réalisée en comparant les compteurs d'émission et de réception sur chaque nœud et en prenant en compte les retransmissions de trames pendant une période donnée. Ces informations servent aux choix de routage.

Les métriques de routage permettent à l'opérateur d'organiser les communications sur son réseau. Le choix des ***prochains sauts*** représente une liste ordonnée de voisins qui peut être construite :

1. à partir des métriques physiques, des métriques de nœud ou de lien. Par exemple, le meilleur prochain saut celui qui constitue le chemin avec les liens les moins chargés, la charge étant évaluée périodiquement ;
2. activement : par exemple on obtient par diffusion de messages et comptage des relais traversés le ***nombre de sauts*** (*hop count*) sur un chemin ;
3. à partir d'autres critères indépendants : par exemple selon le niveau de sécurité des liens, ou selon la stratégie de déploiement de l'opérateur.

Le ***rang*** de chaque voisin influe également sur la construction et le maintien de la topologie de routage, en y évitant les boucles.

Les métriques de bout-en-bout sont définies par flux. Elles correspondent aux exigences des *SLA* telles que décrites au Chapitre 3. Le ***délai de bout-en-bout*** est mesuré par comparaison des instants d'entrée et de sortie du réseau, pour chaque message.

Le ***taux de livraison de bout-en-bout*** est un *KPI* majeur pour chaque application. *Zhao et al.* proposent de l'inférer au niveau des passerelles en comparant les numéros de séquence des trames reçues avec celles générées [**?**]. Cependant, nous ne pouvons pas adopter cette approche pour évaluer le *KPI* : le taux de livraison ne peut être calculé que lorsqu'une trame est finalement reçue. Entre deux réceptions, aucune information ne peut être déduite, ce qui est problématique, en particulier lorsque le trafic n'est pas parfaitement périodique. Cette technique est imprécise si l'opérateur ne connaît pas le profil de génération de chaque flux.

Dans la prochaine section, nous détaillons les solutions d'observation existantes pour les réseaux de l'*IoT*, qui répondent chacune à certains des objectifs que nous avons décrits.

5.3 État de l'art sur l'observation dans les réseaux de l'*IoT*

L'observation de réseau a d'abord été une nécessité dans les réseaux filaires traditionnels. En effet, il faut gérer les équipements à distance, donc collecter les informations sur leur état et performances. Les différentes approches existantes se distinguent sur deux choix :

1. quelle pile protocolaire et quels mécanismes utiliser pour remonter les informations d'observation depuis les entités du réseau jusqu'aux points de collecte ;
2. quels formats utiliser pour quelles informations.

5.3.1 L'observation des réseaux radio multi-sauts

En télécommunications filaires, les ressources de communication sont moins chères (détection de collisions, énergie non limitante, bande passante plus large), donc une démarche active est possible. Dans *Simple Network Management Protocol (SNMP)* [**?**], référence pour l'observation des réseaux filaires, les informations stockées localement sur chaque équipement (dans les bases *MiB*) sont transmises au superviseur suivant un modèle d'échange actif (requête-réponse). Le même besoin d'observation existe pour les réseaux de capteurs sans fil. Pour bien contrôler l'architecture, il est nécessaire de connaître les défaillances des nœuds, l'épuisement de leurs ressources [**?**], les conditions radio dans lesquelles ils se trouvent, etc. Dans les réseaux sans fil, la bande passante est plus chère : les trames ne peuvent pas être dédiées au plan d'observation sans avoir un impact significatif sur le trafic des usagers. Par conséquent, le modèle actif de requête-réponse de *SNMP* [**?**] est lourd et ne convient pas.

La tâche d'observation est plus difficile sur le canal radio qui n'est pas fiable [**?**]. L'information à remonter est donc plus riche et plus difficile à collecter (risque de perte de trames d'observation). De plus, il faut distribuer les ressources de communication entre tous les capteurs. En effet, le nombre de capteurs partageant le même canal peut être important et chaque collision/retransmission est coûteuse pour des nœuds contraints en énergie, mémoire et ressources *CPU*.

Sur une technologie *FTDMA*, l'observation permet de contrôler le partage du canal radio. En effet, l'allocation de ressources requiert une information précise sur les métriques physiques pour dédier des ressources à chaque flux et limiter le risque de collisions entre eux.

Dans un cadre déterministe, le coût d'un mécanisme d'observation est évalué par la quantité de ressources de communication qu'il nécessite. En effet, l'accès au canal est coûteux dans les réseaux sans fil, et les coûts en énergie augmentent avec la quantité d'information à transmettre [**?**]. Ainsi, avec *6TiSCH*, la consommation d'énergie est prévisible [**?**]. Par conséquent, il faut réduire le nombre de messages d'observation et leur taille.

La précision de l'observation dépend de la quantité d'information remontée. Les mécanismes d'agrégation des données [**?**] permettent de réduire le coût en ressources de l'observation, mais au détriment de la précision : l'information est moyennée sur un ensemble de capteurs, ou sur une longue période. Un compromis consiste à choisir le niveau d'agrégation qui réduit les coûts d'observation tout en maintenant un niveau suffisant de détection des problèmes importants, au moindre coût [**?**].

Dans ce travail, nous ne pouvons pas seulement fournir des mesures moyennes, agrégées, d'observation : tous les résultats individuels sont nécessaires pour observer les indicateurs clés de performance (*KPI*) et prouver que ces derniers sont satisfaits.

Il existe deux modes principaux de collecte des informations d'observation [**?**]. Les nœuds adoptent une démarche :

1. active (des ressources sont dédiées à l'observation). L'observation est alors initiée par l'opérateur et spécifique (à un nœud, à un instant donné, pour une information donnée). Elle est plus coûteuse en temps (échange de requête-réponse) et en ressources ;
2. passive (on infère les informations à partir du trafic de données existant, il n'y a pas d'impact sur le réseau) [**?**]. Dans ce cas, les informations sont obtenues *a posteriori*, à partir d'éléments qui ne répondent pas nécessairement aux besoins spécifiques d'observation.

À l'instar d'*Octopus* [**?**] qui différencie 3 modes de fonctionnement de son outil d'obser-

vation (périodique, événementiel, ou sur demande), nous proposons une démarche hybride, qui n'exécute un échange actif que ponctuel, lorsque le *monitoring* passif n'est pas suffisant pour répondre à un des objectif de l'observation en cours.

5.3.2 L'observation dans les protocoles standards

L'observation des réseaux de capteurs est un travail en cours en standardisation [**?**, **?**]. Nous avons vu qu'à l'*IETF*, le groupe de travail *6TiSCH* [**?**] utilise l'expérience du monde *IP* classique pour la gestion du réseau et donc l'observation, active et passive.

Le modèle *Poller-Pollee* [**?**, **?**] permet d'exploiter ces deux approches. Il s'agit de distinguer deux rôles de nœuds :

1. les nœuds *pollee* n'agrègent pas leurs informations et communiquent uniquement jusqu'à un nœud *poller* qui leur est associé ;
2. les *poller* qui agrègent les informations de leurs *pollee* selon une stratégie choisie.

Par exemple, une stratégie passive est utilisée pour la communication des *pollee* aux *poller*, et une stratégie active des *poller* aux passerelles. En effet, cela permet au sous-ensemble des *poller* (choisis pour optimiser tel ou tel critère selon le but recherché) de ne pas saturer le réseau. Dans notre cas, les routeurs d'entrée de l'opérateur jouent ce rôle de *poller*. Les *pollee* sont les nœuds feuilles du client. L'agrégation au niveau des routeurs d'entrée se fait pour la collecte et sans perte d'information : on remonte les valeurs brutes.

Lahmadi et al. proposent d'utiliser les protocoles *IETF* pour relayer les informations d'observation (*6LoWPAN*, *RPL*) [**?**]. Ils utilisent un modèle *Poller-Pollee* et une encapsulation de l'observation dans tous les messages de remontée, afin de réduire le nombre de trames d'observation : un *poller* envoie directement ses requêtes à un ensemble de *pollee*. Le *poller* agrège l'information pour l'envoyer à la passerelle. Mais leur approche ne permet pas une observation exhaustive pour la validation des *SLA* (les *poller* sont choisis sur la base du routage et non sur la disposition des sources de trafic). Parce que les informations de livraison sont agrégées sur les *poller*, l'opérateur ne peut pas calculer le taux de livraison de bout-en-bout pour chaque flux. Dans notre cas, l'agrégation n'est pas envisageable.

L'observation au sein de *6TiSCH* Nous avons vu comment le groupe de travail *IETF 6TiSCH* permet de contrôler finement l'allocation de ressources, en offrant des garanties de *QoS* différenciée sur des flux de trafic de différentes applications/clients en isolation, et en même temps d'optimiser et fiabiliser l'utilisation du canal radio, en utilisant une technologie *FTDMA* sur *IP* [**?**]. *6TiSCH* permet à l'opérateur d'assigner à chaque flux un ensemble de cellules dédiées (un *track*).

La synchronisation des nœuds dans *6TiSCH* facilite l'observation puisque ceux-ci ont tous la notion du temps intervalle par intervalle : l'horodatage est possible grâce à l'*Absolute Slot Number* (*ASN*). Deux éléments de la pile protocolaire *6TiSCH* favorisent l'observation :

1. la sous-couche protocolaire *6top* [**?**] comprend des fonctions de *monitoring* des métriques de lien : elle permet la mise à jour de tables de voisinage, de compteurs de trames, de retransmissions, de valeurs de qualité de lien et de performance des cellules (*PER*, *RSSI*, *ETX*, *LQI*) [**?**] ;
2. *6TiSCH* permet à l'opérateur d'exécuter *CoAP* [**?**] sur ses relais comme protocole applicatif pour l'observation. *CoAP* permet de communiquer les métriques de nœud (e.g. l'énergie résiduelle) en utilisant les formats standard de gestion des dispositifs connectés, comme *COMI* [**?**].

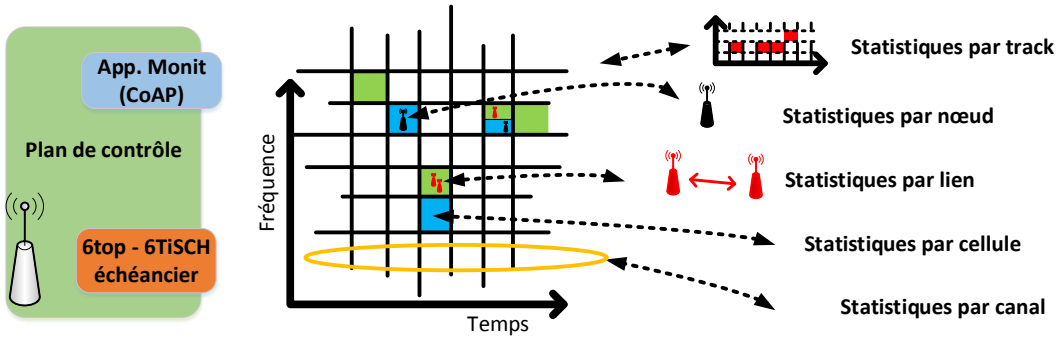


FIGURE 5.1 – Le *monitoring* au sein de *6TiSCH*.

La Fig. 5.1 montre comment les éléments de l'échéancier sont surveillés par *6top*. *6TiSCH* permet à l'opérateur de construire des statistiques de performance par nœud, lien, cellule, canal, ou *track*.

Pour répondre aux objectifs des opérateurs de réseaux de l'*IoT*, les standards doivent préciser les mécanismes de collecte des informations d'observation. En particulier, les informations d'observation des *SLA* doivent être collectées de bout-en-bout sur le réseau.

5.3.3 La collecte de l'information d'observation

La spécificité de l'observation des réseaux de l'*IoT* réside dans la difficulté de la collecte de l'information distribuée sur les nœuds.

La collecte par agrégation *Govindan et al.* proposent d'agréger l'information d'observation avant sa collecte de manière à économiser de la bande passante [**?**]. Les auteurs étudient la robustesse de la collecte face aux pertes de trames. Ils proposent trois modes de collecte (périodique, sur événement, sur demande) afin d'adapter l'observation aux besoins de précision. D'autres propositions limitent le surcoût de l'observation en contrôlant la diffusion des messages sur le réseau [**?**]. Le choix de collecte de l'information ne se base pas sur l'importance de l'observation mais sur une quantité maximale d'informations à remonter.

Dans le cas de l'opérateur, l'observation est sujette aux mêmes exigences de livraison que les flux applicatifs. Par conséquent, les ressources utilisées pour la collecte d'information d'observation doivent être garanties. Cela justifie la remontée d'information en utilisant des cellules *6TiSCH* dédiées, plus coûteuses que des cellules partagées non garanties. Une remontée en cellules partagées permet en revanche de collecter des informations lorsque les contraintes sont plus souples (e.g. collecte proactive de ***rang*** des voisins), sans les exigences des *KPI* des clients.

La collecte par *piggybacking* Le *piggybacking* est le fait de profiter de la transmission d'une trame de données pour lui apposer des en-têtes spécifiques pour le contrôle, entre autres pour l'observation. Dans *IEEE Std 802.15.4-2015 mode TSCH*, la taille maximale d'une trame est déterminée par la taille de la cellule, compte tenu de l'hypothèse que nous avons faite sur la possibilité de transmettre une trame et son acquittement sur chaque cellule. *IEEE Std 802.15.4-2015* formalise l'*apposition* dans les *IE*. Dans *6top*, le protocole 6P utilise des *IE* adéquats pour son fonctionnement. Nous pouvons donc exploiter les *IE* pour apposer les informations d'observation.

Le *piggybacking* permet la remontée d'information d'observation, en utilisant des ressources déjà allouées pour un flux donné, avec des garanties spécifiques de délai et de taux de livraison. L'observation est donc ainsi garantie. Elle évite ainsi de dédier des cellules à l'observation. Nous détaillons cette solution par la suite.

5.3.4 Inférer sur la base d'informations partielles

L'observation du réseau doit aussi permettre à l'opérateur de maintenir les performances de son réseau. Pour ce faire, de nombreuses techniques existent afin de diagnostiquer les problèmes de manière passive, c'est-à-dire sans impact sur le trafic applicatif [**?**].

Le diagnostic peut être construit localement [**?**] : les nœuds voisins échangent des observations afin de confirmer un diagnostic collectif à remonter jusqu'aux passerelles. Ces constructions locales permettent de réduire le taux de fausses alarmes (diagnostic erroné).

Lorsqu'au contraire, une défaillance est détectée de manière centralisée (e.g. interruption du trafic) un diagnostic orienté peut être effectué : l'origine du problème est localisée par construction de topologie sur des ensembles de nœuds potentiellement fautifs de plus en plus réduits [**?**].

Le modèle d'inférence peut se construire sur la base d'informations *apposées* de manière passive [**?**] : une analyse probabiliste permet alors de deviner l'origine du problème la plus vraisemblable. Cette approche passive peut être appliquée à l'inférence de liens fautifs à partir de données de bout-en-bout sur les chemins empruntés [**?**]. Ces informations permettent alors de reconstruire des routes viables [**?**].

Aujourd'hui, nous ne connaissons aucune solution qui réponde aux besoins des opérateurs et prévoie des mécanismes de bout-en-bout d'observation pour réseaux de l'*IoT* opérés. Nous proposons dans ce travail de relever ce défi.

5.4 Remontée d'informations d'observation des *SLA* par *piggybacking* efficace

Dans cette section, nous utilisons une solution exhaustive (remontée pour tous les capteurs et applications de chacun des clients) d'observation des paramètres de *SLA*. En effet, il faut différencier chaque application et chaque nœud car les *KPI* s'appliquent spécifiquement (e.g. une valeur de *PDR* attendue par nœud, individuellement, pour une application de télérelève).

Nous présentons un premier mécanisme minimal d'observation, permettant de suivre périodiquement les deux principaux paramètres de *SLA* : le taux de livraison des messages et leur délai. Pour ce faire, nous installons des compteurs de trames sur les relais de l'opérateur, application par application. Nous en comparons les valeurs aux nombres de trames reçues au passerelles.

Nous proposons d'intégrer ces mécanismes à *6TiSCH* sur les routeurs d'entrée du réseau de l'opérateur. Le routeur d'entrée est le premier équipement - appartenant à l'opérateur - qui relaie les trames à partir des nœuds feuilles (i.e. les dispositifs des clients, par exemple des capteurs de surveillance de la pollution). Les données sont acheminées via le réseau radio multi-saut opéré et transmises au système d'information de chaque client via une passerelle. Le client peut par exemple exiger que 95 % de ses trames soient livrées, en 15 minutes.

Nos contributions de cette section sont les suivantes :

1. nous proposons un mécanisme standard de mesure du délai de bout-en-bout par flux pour une architecture mutualisée d'opérateur reposant sur une technologie radio

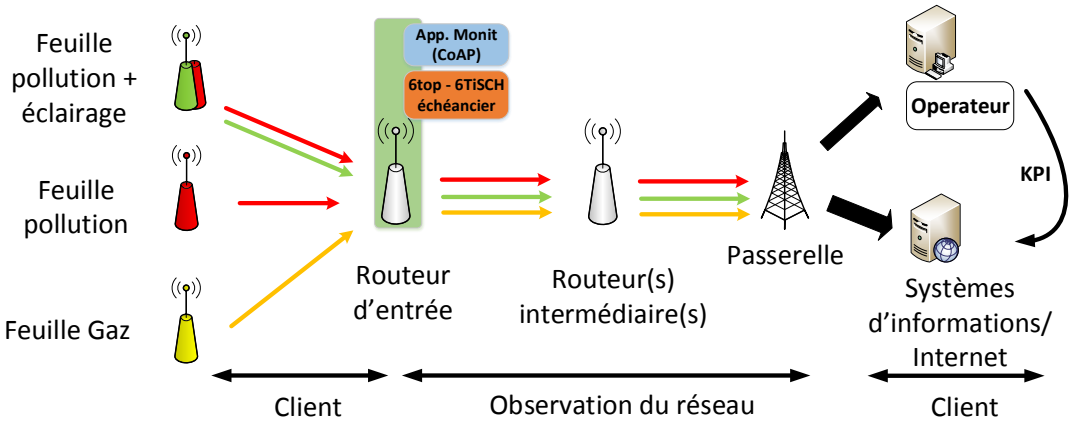


FIGURE 5.2 – L'observation des *SLA* sur les nœuds du réseau opéré.

FTDMA. Nous expliquons comment exploiter l'*ASN* et l'*apposition* pour mesurer le délai de bout-en-bout ;

2. nous proposons un mécanisme standard de mesure du taux de livraison de bout-en-bout par flux et sa transcription concrète dans les protocoles existants de la pile *6TiSCH* ;
3. nous quantifions le coût de notre proposition en termes de ressources de communication.

5.4.1 Cas d'utilisation et modèle

Nous avons vu qu'à l'échelle d'une ville, les flux de plusieurs clients empruntent le même réseau. Nous considérons principalement des trafics de remontée, puisqu'ils sont majoritaires par rapport aux flux descendants (e.g. la commande d'actionneurs). Une charge maximale de trafic est supposée connue par flux pour chacun des usagers. Les trafics sont en effet spécifiés dans les *SLA*. Cette hypothèse correspond à l'absence de trafic non garanti (e.g. génération de trafic non contractualisé par un client).

Nous nous plaçons dans un cas d'utilisation de la pile protocolaire *6TiSCH* pour l'observation :

1. l'observation au niveau applicatif avec *CoAP* [**?**] ;
2. les fonctions de gestion de *IEEE Std 802.15.4-2015 mode TSCH* avec *6top* [**?**].

Les nœuds du réseau sont synchronisés de manière à pouvoir utiliser l'échéancier *FTDMA*.

Chaque routeur d'entrée exécute une application d'observation, en utilisant *CoAP* (Fig. 5.2). Nous réutilisons les éléments d'information (*IE*) définis pour le fonctionnement de *6top* (e.g. l'*IE* permettant de transporter l'*ASN* est directement utilisable). Nous créons aussi des éléments d'information spécifiques pour apposer, sur les trames transitant sur le réseau (trames de données, trames de contrôle *RPL*, etc.), des informations d'observation des *SLA*.

Nous considérons que chaque nœud participe à une ou plusieurs applications (e.g. contrôle de l'éclairage, ou la surveillance de la pollution, Fig. 5.2). Chaque nœud doit se joindre au réseau, en choisissant un ou plusieurs routeurs d'entrée. Nous ne considérons que l'observation de la *QoS* sur le réseau de l'opérateur : les performances ne sont pas garanties sur le premier saut car parfois, le nœud feuille n'appartient pas à l'opérateur, mais au client lui-même. Une mauvaise installation des nœuds feuilles des clients, un changement

des conditions de propagation, pourraient causer des pertes de trames non imputables à l'opérateur. C'est donc bien au niveau des routeurs d'entrée que doit se faire la mesure du nombre de trames entrantes, par feuille et par application. Les routeurs d'entrée constituent le premier endroit où l'opérateur peut contrôler ce qui transite sur son réseau et exercer des stratégies de contrôle d'admission et de sécurité. Nous trouvons donc logique d'y intégrer aussi l'observation des trafics du réseau.

Nous considérons que chaque application génère des messages de taille uniforme, pour simplifier l'évaluation de performances et l'interprétation des résultats. En réalité un client peut négocier une taille maximale pour ses messages. Notre mécanisme d'observation doit s'adapter à la taille de chaque trame de données transmise.

La taille maximale des messages applicatifs d'une source de trafic est définie formellement dans chaque *SLA* afin que l'opérateur puisse en vérifier la conformité. Chaque message peut être fragmenté en un ou plusieurs paquets *IP* qui transitent sur le réseau de l'opérateur, de chaque nœud feuille aux passerelles (Fig. 5.3). L'opérateur doit donc faire correspondre le comptage des paquets *IP* à la taille des messages applicatifs afin de pouvoir lire les paramètres du *SLA*. Une application doit spécifier un port *UDP*, imposé par l'opérateur. Comme nous utilisons la pile *IP* standard, l'opérateur est alors en mesure d'identifier une application en inspectant les couples $< feuille\ IPv6_{@},\ UDP_{port} >$ dans chaque paquet.

Les *KPI* sont mesurés sur une période déterminée, qui dépend de chaque application. Par exemple, un taux de livraison pour un compteur de gaz peut être mesuré chaque jour, tandis qu'un système d'éclairage public nécessite une fiabilité minimale sur une période plus courte : 1 heure.

Ici, nous nous concentrons uniquement sur les communications montantes : les capteurs envoient leurs mesures à destination d'une passerelle, connectée à l'internet. Les *KPI* pour le sens descendant, rarement mis en œuvre dans l'*IoT* actuel, ne sont pas pris en compte dans ce travail.

Lors de la mise en place du réseau, nous allouons une adresse *IP* unique à chaque nœud feuille afin qu'elle lui serve d'identifiant unique (Fig. 5.3). Cela permet d'identifier la source d'un message sans ambiguïté. Dans ce cas, le multi-chemin est possible (e.g. un nœud feuille peut changer de routeur d'entrée) puisque les sources sont identifiées de bout-en-bout.

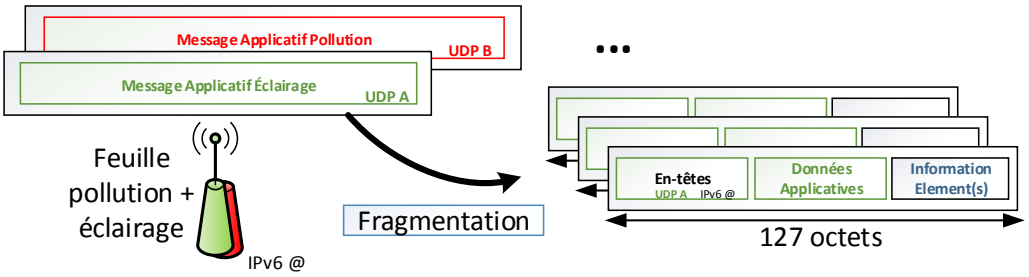


FIGURE 5.3 – La fragmentation des messages applicatifs.

La Fig. 5.3 montre comment un message d'une feuille supportant deux applications est fragmenté en plusieurs paquets *IP*. Les données du nœud feuille transitent depuis l'application du client vers une passerelle en passant la couche *IP* de chaque nœud intermédiaire.

Les *SLA* spécifient une période d'observation spécifique à chaque application. Cela permet de prouver régulièrement au client que les clauses sont respectées. L'opérateur peut augmenter la fréquence d'observation si les conditions l'obligent (dégradation de la qualité d'un lien, etc.). L'opérateur peut aussi obtenir une information d'observation sur demande, pour des besoins de gestion, ou en cas de non réception des informations périodiques.

Nous détaillons dans les sous-sections suivantes comment nous mesurons, collectons et calculons les *KPI* de délai et de taux de livraison de bout-en-bout, sur une bande passante limitée (réseau radio multi-saut) et avec des coûts énergétiques réduits.

5.4.2 Mesurer le délai de bout-en-bout

Le client s'intéresse au délai de réception de ses messages applicatifs. L'opérateur doit donc s'engager sur le temps de transit des trames correspondantes. Le délai que l'on peut appréhender unitairement (i.e. le plus petit paramètre mesurable de bout-en-bout) est le temps de parcours d'une trame d'un client sur le réseau de l'opérateur (entre le routeur d'entrée et le point de sortie).

Le délai de bout-en-bout doit se mesurer pour toutes les trames de chacun des nœuds feuilles, pour chaque application. En effet, la spécification des *SLA* [**?**] offre un large choix de paramètres complexes. Seule cette information d'observation complète (pour toutes les trames) permet de réaliser tous les calculs de paramètres envisagés.

Il faut donc comparer pour chaque message l'instant d'entrée à l'instant de sortie. Pour cela, un horodatage de chacune des trames sur le routeur d'entrée est nécessaire. La comparaison est rendue possible par le fait que les réseaux *FTDMA* sont synchronisés. Chaque routeur d'entrée possède la même vision précise du temps et donc l'horodatage peut être exploité.

Chaque routeur d'entrée appose lui-même un horodatage sur chaque trame de données qui le traverse. Si la trame est reçue à une passerelle, celle-ci peut alors stocker les deux valeurs (horodatage d'entrée et de sortie). Si la trame est perdue, le délai de bout-en-bout ne peut pas être calculé. Il faut que la précision de l'horodatage soit équivalente ou meilleure que l'exigence de précision sur le délai. Nous estampillons les événements à l'aide de l'*ASN* : chaque intervalle correspond à un incrément de l'*ASN* (sur 5 octets). Par conséquent, la résolution est de l'ordre de la durée de l'intervalle de temps (typiquement 10 ms). L'*ASN* prend des valeurs différentes pendant 350 ans. Nous considérons cela largement suffisant pour les contraintes des applications visées (temps de réaction de l'ordre de la dizaine de secondes, durée de vie de la solution de l'ordre de la dizaine d'année, e.g. pour la télérelève, l'éclairage public ou la remontée d'alarmes).

Le paramètre à calculer pour le délai peut être spécifié de différentes manières suivant le *SLA* : une fonction est appliquée aux valeurs unitaires collectées (maximum, minimum, moyenne, etc.). Suivant le *SLA*, la fonction de calcul du délai prendra comme paramètres un nombre donné de valeurs remontées (e.g. la moyenne des délais unitaires pour les flux d'une l'application donnée sur les 10 dernières trames reçues).

Nous proposons la solution simple suivante :

1. lorsque le routeur d'entrée reçoit une trame, il y insère un *IE* contenant l'*ASN* courant. Cet *IE* occupe 10 octets (5 pour le type et la taille de l'*IE* et 5 pour l'*ASN* lui-même) ;
2. lorsqu'une passerelle reçoit la trame, elle extrait l'*ASN* de la trame et calcule le délai de bout-en-bout en le comparant à l'*ASN* courant. Cette métrique est ensuite stockée dans le ***Service Registry*** de l'opérateur.

5.4.3 Mesurer le taux de livraison de bout-en-bout

L'idée est de calculer le ratio des messages applicatifs des clients, qui sont bien restitués pendant une période donnée : les trames routées qui ne se sont pas perdues sur le réseau de l'opérateur. Nous proposons un mécanisme d'observation périodique : la période de calcul est spécifiée dans les *SLA* pour chaque application.

L'observation périodique Nous pourrions comme pour le délai, encapsuler un numéro de séquence dans les trames de données. L'idée serait de détecter une perte de trame en repérant un décalage de numéro. Mais nous avons décidé de ne faire aucune hypothèse sur le profile de trafic entrant, ni sur sa périodicité, ni sur la route qu'il emprunte. Dans ce cas, outre la difficulté de la mise en place d'un numéro de séquence distribué par routeur d'entrée et flux :

1. nous ne pouvons pas calculer de ratio, le nombre de trames entrantes étant inconnu ;
2. lorsqu'aucune trame n'est reçue, il est impossible de distinguer le cas de la perte de toutes les trames du cas de l'absence d'envoi par la source pendant la période ;
3. si les trames empruntent plusieurs chemins distincts, leurs arrivées seront entrelacées au niveau des passerelles. Les numéros de séquences ne seront pas lus dans l'ordre de génération des trames mais dans l'ordre d'arrivée à une passerelle donnée. Le calcul du *PDR* est donc différé.

Il faut donc comparer le nombre de trames sortantes avec le nombre de trames entrantes.

Les nœuds étant synchronisés, les routeurs d'entrée peuvent compter, pour chaque période d'observation (différente pour chaque application client, chaque *SLA*), le nombre de messages entrants depuis les feuilles. Un vecteur de valeurs de ces compteurs (pour une application et pour tous les nœuds feuilles) est ainsi généré périodiquement.

Au niveau des passerelles, l'opérateur tient les mêmes compteurs de messages en sortie du réseau (par source et par application). Lorsque les compteurs d'entrée et de sortie sont reçus au niveau des passerelles, ou après un délai maximum, les paires de valeurs (compteurs d'entrée et de sortie pour une période donnée) sont stockées en base de données (le ***Service Registry*** au Chapitre 4).

La construction des vecteurs d'observation Chaque routeur d'entrée incrémente un ensemble de compteurs spécifiques, associés à un couple $< IPv6_{@}, UDP_{port} >$. Ainsi, l'opérateur est en mesure d'identifier les applications dans chaque trame en inspectant les couples $< feuille\ IPv6_{@}, UDP_{port} >$.

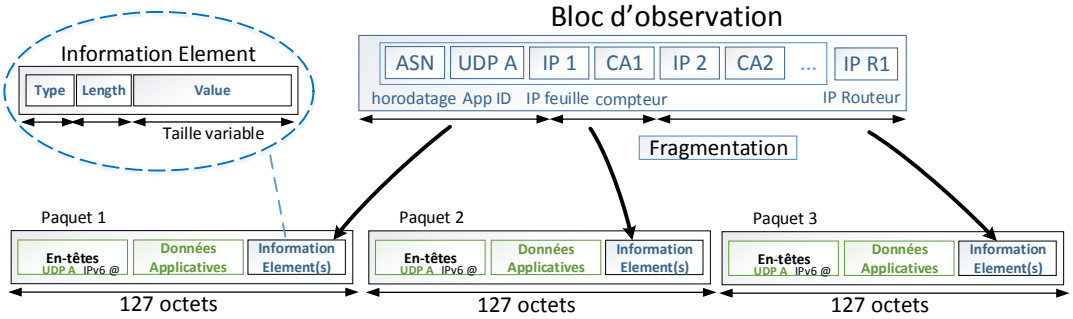


FIGURE 5.4 – Vecteur périodique de valeurs de compteurs pour une application A donnée, sur un routeur d'entrée donné.

Pour chaque application supportée par un ou plusieurs nœuds feuilles, le routeur d'entrée doit rendre périodiquement compte aux passerelles des valeurs de tous les compteurs. Comme chaque application peut spécifier une période d'observation différente, chacune d'entre elles est traitée indépendamment.

Ainsi, le routeur d'entrée crée un vecteur de tous les compteurs pour une application donnée, identifié par un port *UDP* (Fig. 5.4). Les nœuds feuilles peuvent communiquer

avec différents routeurs d'entrée au cours de la même période d'observation. Les trames de la même feuille A peuvent arriver sur plusieurs routeurs d'entrée différents, e.g. à cause d'un changement, d'un routage redondant, de la mise en fonctionnement d'un nouveau relais. La présence de trames de chacun des nœuds feuilles n'est pas connue *a priori*. Afin de rationaliser les informations au niveau de chaque passerelle, les vecteurs doivent donc spécifier l'adresse *IP* de chaque nœud.

Le routeur d'entrée construit alors un ensemble d'informations d'observation (nommé *bloc d'observation*, Fig. 5.4) qui contient :

1. le port *UDP* UDP_A ;
2. l'ASN correspondant au moment de génération du bloc ;
3. le vecteur de couples $< feuille\ IPv6_@,\ compteur >$.

Pour chaque application, le routeur d'entrée garde en mémoire dédiée le bloc d'observation correspondant.

La Fig. 5.4 montre la forme que doit prendre le vecteur de compteurs, avec l'adresse *IP* de chacun des nœuds sources, l'identifiant de l'application et l'ASN de fin de période d'observation. La taille du vecteur dépend du nombre de nœuds feuilles répertoriés sur chaque routeur d'entrée pour une application donnée :

$$\begin{aligned} &Taille(Vecteur(App)) = \\ &Taille(ASN) + Taille(UDP_{port}) + nb_{dev}(App) * (Taille(IP_@) + Taille(compteur)) \end{aligned} \quad (5.1)$$

Avec $Taille(V)$ représentant la taille de la variable V, et nb_{dev} le nombre de nœuds rattachés au routeur d'entrée pour cette application. Les autres variables sont donnés dans le Tab. 5.1.

L'échange entre nœuds de ces informations est réalisé en intégrant dans un ou plusieurs *IE* l'information d'observation périodique. Si les informations sont assez réduites (seulement quelques feuilles sont fixées à ce routeur d'entrée pour une application donnée, i.e. leurs trames transitent par lui), un *IE* suffit. Sinon, les informations seront fragmentées dans plusieurs *IE* (Fig. 5.4).

5.4.4 Collecter l'information d'observation

Un mécanisme sur demande Nous détaillons ici comment les informations d'observation, distribuées sur tous les routeurs d'entrée, peuvent être collectées à la demande : l'opérateur demande explicitement l'information de chacun des routeurs d'entrée. Le mécanisme ***sur demande*** concerne tout ou partie de l'information disponible sur chaque nœud (ensemble des vecteurs applicatifs).

Pour collecter cette information, nous utilisons le modèle *requête-réponse* via un échange de trames dédiées à l'observation. Dans le cas d'utilisation de *CoAP*, l'opérateur initie l'échange en envoyant un message *CoAP GET* pour obtenir le bloc d'observation, pour une application donnée, pour un intervalle de temps donné.

De la même manière, les routeurs d'entrée peuvent mettre en œuvre un comportement autonome, en utilisant le mécanisme de *CoAP Observe* [**?**]. Avec un premier *GET* l'opérateur configure chaque routeur d'entrée en indiquant une période donnée, de sorte que le routeur d'entrée crée de façon autonome un message périodique *CoAP* contenant une collection de blocs d'observation. Dans le cas où des informations d'observation sont manquantes après une période donnée (e.g. à cause d'une perte de trame), l'opérateur envoie un *GET* au routeur d'entrée correspondant.

Même si l'approche ***sur demande*** permet de diagnostiquer avec précision un problème quand il est détecté, elle n'est pas efficace car elle doit consacrer certaines cellules à la collecte d'information d'observation (échanges ponctuels de requêtes et réponses).

Un mécanisme périodique efficace en énergie Nous proposons également un mécanisme qui *appose* les blocs d'observation périodiques sur les trames de données. De nombreuses trames de données contiennent moins de 127 octets, nous les complétons pour y inclure des informations d'observation. Étant donné que chaque intervalle de temps a une durée fixe, nous améliorons ainsi l'utilisation de chaque cellule.

Dans l'exemple de la Fig. 5.4, à la fin de la période d'observation de l'application UDP_A, chaque routeur d'entrée génère un ou plusieurs *IE* qui contiennent le bloc d'observation correspondant (chaque bloc est fragmenté dans différents *IE* si nécessaire). Ensuite, ces *IE* sont *apposés* sur les trames de données transmises par le routeur d'entrée vers la passerelle. Chaque passerelle est alors en charge de ré-assembler les *IE* pour reconstruire le bloc d'observation.

La Fig. 5.5 précise que lorsqu'un message applicatif est fragmenté ou de petite taille, le routeur d'entrée peut *apposer* un *IE* sur le dernier fragment si l'espace restant y est suffisant (> 5 *octets*). Nous nommons ***conteneur*** l'espace disponible pour apposer des blocs sur chaque dernier fragment.

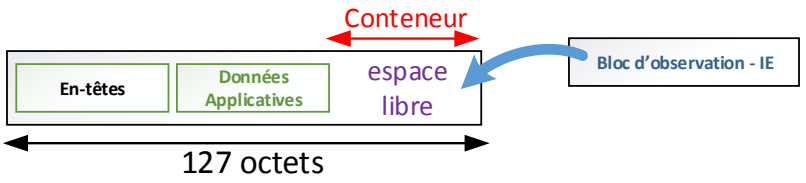


FIGURE 5.5 – L'*apposition* des blocs d'observation sur les messages de données.

La quantité de trames de données peut être insuffisante pour transmettre toutes les informations d'observation. Dans ce cas, nous proposons que le routeur d'entrée crée une trame dédiée à l'observation (nommée trame de ***monitoring-dédié***) s'il a encore des informations d'observation à transmettre après un intervalle de temps donné. La passerelle est en mesure de reconstituer l'ensemble des informations avec les *IE* déjà reçus.

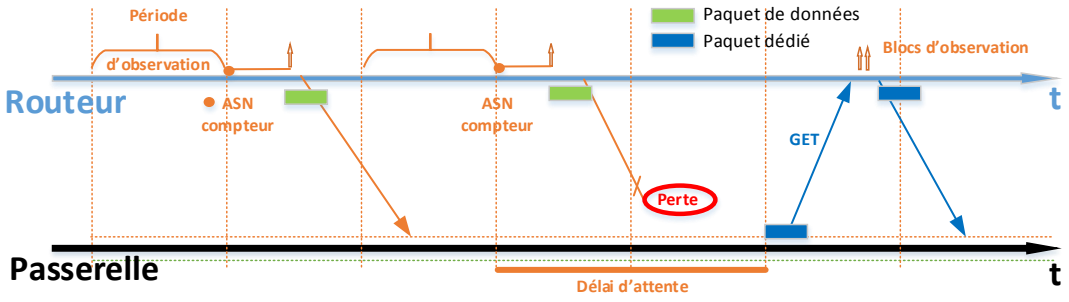


FIGURE 5.6 – La formation et l'échange des informations d'observation périodique.

La Fig. 5.6 montre la génération périodique et le *piggybacking* des valeurs de compteurs telle que décrite ci dessus. En cas de non réception d'une information d'observation après un délai donné, l'opérateur envoie une requête symbolisée par un *GET*, concernant la ressource manquante (nous utilisons le mécanisme ***sur demande***). Le routeur d'entrée concerné répond alors à la demande sur une trame dédiée. Le second bloc d'observation

dans la figure n'est pas reçu après un délai de deux fois la période d'observation : un *GET* est alors généré.

Suivant la période exigée d'observation, indiquée dans chaque *SLA*, l'opérateur sera en mesure de calculer les taux de livraison de toutes les données concernées.

Les routeurs d'entrée choisissent de manière autonome les conteneurs sur lesquels ils *apposent* les blocs d'observation. Les stratégies de *piggybacking* sont étudiées par la suite.

5.4.5 Observation : isolation et fragmentation

Chaque routeur d'entrée doit sélectionner le ***conteneur*** dans lequel apposer ses blocs d'observation. Si un routeur d'entrée sélectionne un petit conteneur, il peut gâcher des ressources : un autre conteneur plus grand peut être disponible plus tard, qui n'obligerait pas la fragmentation des informations. Inversement, si un routeur d'entrée attend trop longtemps sans trouver assez d'espace, une nouvelle trame de ***monitoring-dédié*** est générée, ce qui gaspille de la bande passante et de l'énergie.

Donc, nous évaluons deux stratégies d'apposition opposées :

1. le *piggybacking* par ***applications mélangées*** : aucune sélection n'est faite. Les relais sélectionnent le premier conteneur disponible dans n'importe quelle trame transmise par le routeur d'entrée ; le *piggybacking* de l'observation se réalise sur le premier fragment disponible sans distinguer une application d'une autre ;
2. le *piggybacking* par ***applications isolées*** : l'information d'observation de l'application UDP_A n'est apposée que sur les trames de données générées par l'application UDP_A. Nous préservons l'isolation des flux, c'est-à-dire que l'observation de l'application UDP_A n'a pas d'impact sur la performance de l'application UDP_B.

La seconde stratégie est plus contraignante, en particulier parce que les applications qui génèrent des trames avec une taille proche du maximum ne disposent pas assez d'espace pour apposer toutes les informations d'observation.

Limites du *piggybacking* périodique Le mécanisme de remontée périodique prépare ses informations sous la forme d'*IE* qu'on appose dans les trames à destination d'une passerelle. Ainsi, nous n'utilisons pas de nouveaux blocs temps-fréquences pour ce mécanisme (nous complètons l'utilisation de ceux des messages de données).

Le nombre de messages de données peut cependant être insuffisant pour pouvoir transmettre l'intégralité des informations de contrôle. Dans ce cas, les informations s'accumuleraient sur le routeur d'entrée, épuiseraient sa mémoire, ou se perdraient. De même, si la trame de données support se perd (collision, perte de connectivité), l'information est perdue.

Plusieurs solutions peuvent être apportées afin de réduire ces risques :

1. une stratégie de choix du message porteur : le *piggybacking* se fait alors seulement sur certains messages (plus rapides, plus fiables, ou moins confidentiels). Cela accroît la complexité du mécanisme au niveau du routeur d'entrée ;
2. choisir le *piggybacking* par ***applications isolées***. Cela limite la quantité des opportunités de collecte. En particulier pour des trafics applicatifs qui laissent peu de place au *piggybacking*. Et en cas de perte pour une application donnée, on perd aussi l'information sur cette perte (le mécanisme est moins fiable) ;
3. une stratégie de fragmentation des données applicatives : imposer une taille maximale de fragment pour les données transmises à partir du routeur d'entrée. Ainsi, nous réservons de fait un espace dédié au *piggybacking* d'observation sur chaque

trame de données. Imposer ainsi une taille maximale de fragment de données augmente la quantité de ressources nécessaires (le nombre de blocs temps-fréquences) à la transmission du trafic de données.

Nous proposons que le routeur d'entrée forme un message dédié à l'observation lorsque les *IE* en mémoire sont trop nombreux. Ce message est au format décrit pour l'observation ***sur demande***.

5.4.6 Analyse des informations collectées

Une fois les mesures d'observation collectée et stockées en base de données, il suffit de les analyser pour en extraire les conclusions sur les clauses des *SLA*. Les calculs sont exécutés par le ***SLA Manager*** décrit au Chapitre 4.

Construction du *PDR* Si l'observation périodique a correctement fonctionné (pas de pertes d'informations d'observation), le calcul du taux de livraison correspond simplement au ratio, sur cette période, du nombre de trames des clients reçues sur le nombre de trames entrées sur le réseau de l'opérateur, pour chaque source et chaque application.

Il faut agréger les données issues des différents routeurs d'entrée et des passerelles. En effet, le trafic d'une source pouvant entrer sur le réseau par différents routeurs d'entrée sur la même période (multi-chemin), il faut faire la somme des compteurs correspondants.

S'il y a perte d'information d'observation, le ***SLA Manager*** déclenche une action sur le réseau : il peut demander une mesure précise en préparant un message *CoAP GET* à destination des routeurs d'entrée dont les données manquent. Il peut changer la période d'observation d'une application.

Les résultats obtenus suite à ce *GET* doivent être analysés en fonction des exigences des clients. On peut par exemple faire le calcul du *PDR* sur une période plus longue incluant la période d'observation où les informations sont manquantes.

Construction du délai De la même manière, le délai de transit sur le réseau se mesure par différence entre l'*ASN* de sortie et l'*ASN* d'entrée. Nous appliquons une même fonction (e.g. la moyenne ou le maximum) sur un ensemble donné de valeurs de délai : chacune correspond à une trame reçue, pour chaque nœud et application.

5.5 Évaluation de performances

Dans cette section, nous évaluons par simulation le coût en ressources de communications des mécanismes standards d'observation de *SLA* proposés. Nous considérons des paramètres raisonnables pour les protocoles utilisés et pour un cadre d'applications urbaines telles que spécifiées dans le standard *ETSI* [**?**].

5.5.1 Paramètres et hypothèses

Nous quantifions en premier lieu l'impact sur les ressources de l'*apposition* de l'information d'observation par rapport à l'utilisation exclusive de trames de ***monitoring-dédié***. Nous avons obtenus ces résultats par simulation en utilisant un algorithme de *Monte-Carlo*, reproduisant le comportement d'un routeur d'entrée [**?**].

Nous calculons d'abord la quantité totale de trafic de données et d'observation, en octets et en nombre de trames et blocs. Nous en déduisons le nombre de conteneurs de chaque application, compte tenu de la fragmentation des trames de données. Ensuite, nous

TABLE 5.1 – Paramètres du mécanisme d'observation périodique sur un routeur d'entrée

Paramètres	Valeur	Couverture $\Gamma(App)$
Télérelève de compteurs d'eau	100 octets, 2/jour	4 feuilles
Application de stationnement intelligent	90 octets, 24/jour	8 feuilles
Application de surveillance de la pollution	10 octets, 24/jour	2 feuilles
En-tête de fragmentation (données)	5 octets	
En-tête de fragmentation (*IE*)	5 octets	
En-têtes de protocoles de la pile *6TiSCH*	62 octets	
Taille maximale de trame *6TiSCH*	127 octets	
Port *UDP* compressé	1 octet [1]	
adresse *IP* compressée	2 octets	
Absolute Slot Number (*ASN*)	5 octets	
Compteur d'observation	1 octet	
Applications par feuille	1 ou 2	
Feuilles par routeur d'entrée	15	
Fréquence des messages d'observation	1 à 24 messages par jour	

comparons les stratégies en choisissant les lois d'arrivées des messages de données. Par exemple, les messages sont triés par tailles décroissantes. Ce choix est défavorable pour la stratégie d'***applications mélangées***.

Nous regardons la quantité d'information qui transite sur un routeur d'entrée en une journée, en supposant que tous les nœuds feuilles qui lui sont rattachés sont correctment configurés et qu'il n'y a pas de pertes de trames à ce saut. Comparer la quantité d'octets nécessaire à l'observation avec celle des données donne, en première approche, une estimation du coût énergétique relatif du mécanisme étudié.

Les paramètres à partir desquels nous évaluons le mécanisme périodique sont résumés en Tab. 5.1.

Afin d'étudier les scénarios multi-applications hétérogènes dans le contexte des villes intelligentes, nous adoptons la configuration moyenne décrite dans le rapport technique de l'*ETSI* [**?**]. Plus précisément, notre étude se concentre sur trois applications typiques : télérelève de compteurs d'eau, stationnement intelligent et surveillance de la pollution. Nous choisissons des valeurs représentatives pour leurs modèles et paramètres de trafic, comme le résume le Tab. 5.1. De plus, nous supposons un déploiement homogène et uniforme sur une grande ville. Pour chaque application, seule une partie des nœuds feuilles supporte le trafic (voir la couverture, $\Gamma(App)$ dans le Tab. 5.1).

Nous supposons que la charge utile maximale d'une trame de données est de 65 octets (ajoutés à 62 octets d'en-têtes, nous atteignons la taille de trame maximale de 127 octets). Au-delà de 65 octets, les messages de données sont fragmentés. Nous supposons que le réseau met en œuvre la fragmentation et la compression d'en-têtes avec *6LoWPAN* [**?**]. Les en-têtes *UDP* et *IP* sont compressés (Tab. 5.1). En retirant de la charge utile maximale les en-têtes de fragmentation (5 octets), chaque fragment dispose de 60 octets de charge utile.

La taille des messages applicatifs considérés varie de 10 octets à 100 octets. Nous varions la fréquence de génération des messages entre un par heure et un par jour [**?**]. Nous supposons que la répartition des nœuds feuilles est homogène sur l'espace de la ville considérée. Dans ces conditions, chaque routeur d'entrée reçoit le trafic d'environ 1 feuille pour les applications les moins denses, jusqu'à celui d'environ 8 feuilles pour le *Smart*

1. Cela correspond à la quatrième option de compression de ports *UDP* présentée en [**?**], réservant 256 possibilités en cumulant les 4 bits destinés au *Source Port* aux 4 bits du *Destination Port*.

Parking (pour un total de 15 feuilles).

La fréquence d'observation est la principale variable de l'évaluation. Une solution d'observation pertinente ne devrait pas générer plus d'informations que l'application elle-même. Par conséquent, nous exprimons la fréquence d'observation comme un pourcentage de la fréquence de génération des messages de données. Nous varions la fréquence d'observation entre 0 % et 120 %, pour observer son impact sur le surcoût de communication. Dans ces conditions, la fréquence d'observation est choisie ici pour balayer les valeurs allant de 1 remontée par jour à 24 remontées par jour.

Par exemple, 8 nœuds feuilles contribuent à l'application de stationnement et génèrent un message par heure. Une fréquence d'observation de 50% (de la fréquence de données) signifie qu'un bloc d'observation est généré toutes les 15 min :

$$50\% * 8 * 1\ messages/heure = 4\ messages/heure \tag{5.2}$$

Le *piggybacking* des *IE* d'observation se fait sur le dernier fragment de données, si la charge utile disponible sur celui-ci est suffisante (>5 o). Si le bloc d'observation est plus grand que la taille disponible dans un fragment de données, il est lui-même fragmenté dans plusieurs *IE* et apposé sur plusieurs fragments de données (avec un en-tête de fragmentation de 5 octets pour l'information d'observation).

Lorsque la quantité d'information d'observation et le surcoût protocolaire lié à sa fragmentation dépasse la quantité disponible pour le *piggybacking*, nous supposons que chaque routeur d'entrée génère des trames dédiées permettant d'écouler ce qui n'a pas pu être remonté par le *piggybacking*. Les messages dédiés d'observation sont supposés similaires (charge utile, fragmentation) aux autres trames de données.

Deux procédures sont évaluées sur le routeur d'entrée :

1. conteneurs ***exclusifs*** : une trame contient au plus **un** bloc d'observation. Si l'*IE* ne remplit pas tout le conteneur, une partie de la bande passante est gaspillée. Cependant, cette solution est très simple à mettre en œuvre ;
2. conteneurs ***partagés*** : Quand une trame doit être transmise, le routeur d'entrée remplit tout l'espace restant avec toutes les informations d'observation qu'il a gardées en mémoire jusque là. En particulier, une trame peut transporter plusieurs *IE*. Ainsi, toutes les trames ont une charge utilisée proche du maximum de 127 octets, en-têtes compris.

Le choix de la procédure de conteneurs est orthogonal au choix de la stratégie d'apposition (applications mélangées ou l'isolation d'application, voir la section 5.4.5). De plus, l'opérateur doit choisir une des deux procédures pour les trames de ***monitoring-dédié***.

Avec la procédure de conteneurs ***exclusifs***, le coût en ressources de communication de l'observation de chacune des applications est indépendant et le coût total en est la somme. Avec la procédure de conteneurs ***partagés***, le coût total dépend de l'efficacité du *piggybacking*. En effet, selon les tailles disponibles et les arrivées de données à remonter, le surcoût lié aux en-têtes de fragmentation augmentent si les conteneurs sont de petites tailles, ainsi que la quantité de messages dédiés.

L'évaluation du mécanisme se fait en variant la fréquence d'observation de chaque application. Pour chaque application, le calcul de la taille des blocs périodiques d'observation s'effectue selon l'Eq. (5.1). L'ensemble de ces expressions pour chaque application permet de calculer la quantité de blocs d'observation que chaque routeur d'entrée doit faire remonter. Nous calculons le nombre de conteneurs utilisés et le surcoût d'observation correspondant (y compris les en-têtes et la fragmentation potentielle des blocs) en fonction de chaque procédure (e.g. dans la procédure de conteneurs ***partagés***, les blocs sont fragmentés pour utiliser au maximum les conteneurs).

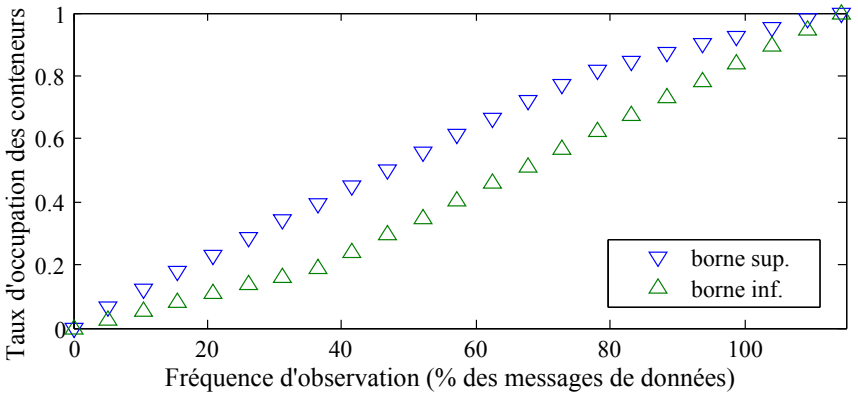
5.5.2 Taux d'occupation des conteneurs

La Fig. 5.7 illustre le ratio de conteneurs remplis d'information d'observation apposée. Cette métrique mesure la saturation du mécanisme d'observation.

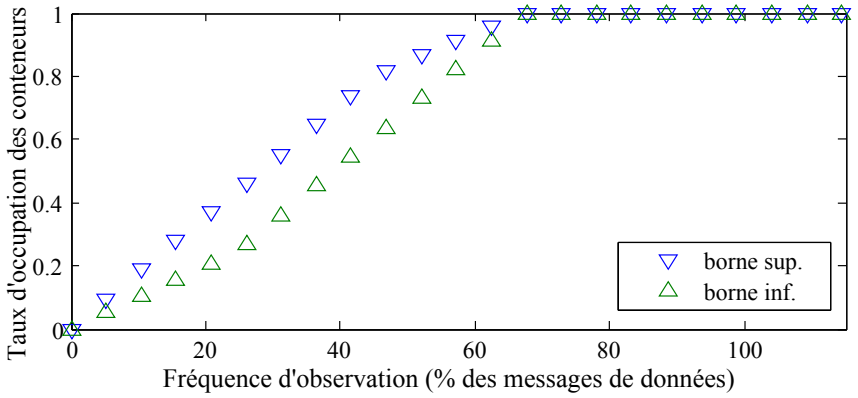
Cependant, l'efficacité de l'observation dépend également de la distribution de la taille des différents conteneurs et de l'ordre dans lequel ils apparaissent sur le routeur d'entrée. Cet ordre étant inconnu, nous essayons de l'encadrer par les pires et meilleurs cas suivants :

1. meilleur cas (borne inférieure) : nous supposons que le routeur d'entrée transmet toujours les trames avec les conteneurs les plus grands en premier (ordre décroissant des tailles de conteneurs). Ceci constitue un scénario optimiste : la fragmentation est réduite au minimum puisque les conteneurs les plus importants sont utilisés en premier lieu ;
2. pire cas (borne supérieure), nous supposons qu'il transmet toujours les trames avec les plus petits conteneurs en premier. Cela représente un scénario pessimiste : il maximise la surcharge créée par la fragmentation.

En fait, notre solution d'observation obtiendra des résultats entre ces deux bornes. Nous voyons que les bornes sont suffisamment serrées (Fig. 5.7) pour estimer avec précision la performance nous obtiendrions si les tailles de trames sont choisies au hasard.



(a) Conteneurs partagés (plusieurs blocs d'observation par conteneur)



(b) Conteneurs exclusifs (un bloc d'observation par conteneur)

FIGURE 5.7 – Taux de conteneurs utilisés (métrique de saturation).

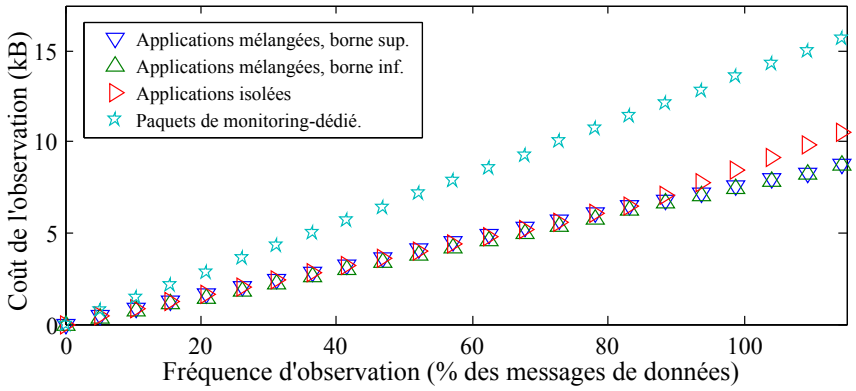
La Fig. 5.7 montre le premier résultat de cette évaluation, effectuée sous *Matlab*.

Quand nous limitons le *piggybacking* des *IE* d'observation à un seul par trame de données (procédure de conteneurs exclusifs, Fig. 5.7b), tous les conteneurs sont utilisés

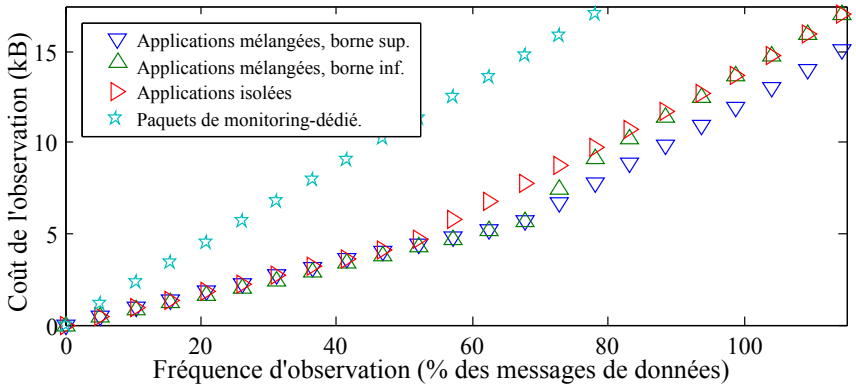
lorsque la fréquence d'observation est supérieure à 65 % de la fréquence de données. Dans ce cas, des trames de ***monitoring-dédié*** doivent être créées, ce qui influe négativement sur la consommation d'énergie et la bande passante.

Au contraire, **quand on autorise une trame de données à supporter plusieurs *IE* d'observation dans le même conteneur** (procédure de conteneurs partagés), **nous réduisons significativement le surcoût de communication** (Fig. 5.7a). Il n'est pas nécessaire de générer une nouvelle trame de ***monitoring-dédié*** si la fréquence d'observation ne dépasse pas 115%.

Nous avons pu observer en faisant varier la taille des messages applicatifs (de -10 octets à +50 octets par rapport à la taille indiquée en Tab. 5.1, pour toutes les applications à la fois) que celle-ci a un impact important sur les performances et le coût du mécanisme, en modifiant le nombre et la taille des conteneurs. La taille maximale des messages qui est signée dans les *SLA* aura donc également un impact sur le coût de l'observation associée : il est nécessaire de quantifier ce coût (e.g. avec notre méthode).



(a) Conteneurs partagés (plusieurs blocs d'observation par conteneur)



(b) Conteneurs exclusifs (un bloc d'observation par conteneur)

FIGURE 5.8 – Surcoût de la solution d'observation – nombre d'octets d'information.

5.5.3 Surcoût des mécanismes d'observation

Nous étudions le coût supplémentaire en ressources lié au processus d'observation, c'est-à-dire le nombre d'octets dédiés au mécanisme d'observation (Fig. 5.8). En particulier, nous comparons la solution d'*apposition* (et donc de réutilisation de l'espace disponible sur les trames) et la solution dédiée (des trames de ***monitoring-dédié*** sont créées et utilisent des ressources dédiées). Nous supposons que pendant chaque jour, un routeur d'entrée transmet

50 Ko de trafic de données pour trois clients (Tab. 5.1).

Dans le cas où nous n'utilisons pas le mécanisme d'apposition (étoiles en Fig. 5.8), de nouvelles trames de ***monitoring-dédié*** sont créées dès qu'un bloc d'observation doit être transmis vers une passerelle. Ceci constitue la pire solution, car de nouvelles opportunités de transmission (intervalles de temps) doivent leur être dédiées à chaque observation. Toutes les autres solutions apposent des informations d'observation sur les trames de données et sont beaucoup plus efficaces pour réduire la bande passante et la consommation d'énergie. **Les solutions d'apposition permettent de réduire efficacement le surcoût d'observation**.

Lorsque nous comparons les conteneurs exclusifs et partagés (Fig. 5.8a et 5.8b), nous vérifions qu'**autoriser un conteneur à apposer plusieurs blocs d'observation réduit considérablement le surcoût**. En effet, la procédure de conteneurs exclusifs commence à saturer lorsque la fréquence est supérieure à 60% ou 70% : de nouvelles trames de ***monitoring-dédié*** doivent être générées, ce qui augmente le surcoût.

Nous notons également que si les informations d'observation d'une application sont autorisées à utiliser le trafic de données d'une autre application (graphe des *applications mélangées*), cela réduit le surcoût par rapport à la stricte isolation du trafic de différentes applications (*applications isolées*). Cependant, le gain est mineur. Par conséquent, nous préférons la méthode d'isolation d'application, parce qu'**elle favorise une séparation stricte du trafic entre les différents clients / applications sans impact significatif sur la performance du système**.

5.5.4 Conclusions de l'évaluation

Dans ce travail, nous avons adopté le point de vue d'un opérateur de réseau de l'*IoT* : il faut observer avec précision le réseau pour prouver que les exigences des clients sont respectées. Nous proposons des mécanismes pour mesurer le délai et le taux de livraison de bout-en-bout. Parce que les ressources de communication constituent une contrainte majeure dans les réseaux sans fil multi-sauts, nous évaluons l'efficacité de différentes techniques d'apposition. En particulier, nous comparons plusieurs stratégies, mélangeant plusieurs informations d'observation dans la même trame de données, ou au contraire garantissant l'isolation de trafic. Nos mécanismes d'apposition sont efficaces pour réduire l'utilisation de la bande passante. Les stratégies de mise en mémoire des informations avant leur remontée opportuniste réduisent le coût global de l'observation.

Le mécanisme de *piggybacking*, avec les variantes que nous avons évaluées, répond aux besoins d'une observation orientée *SLA* pour les réseaux urbains tels que présentés dans le rapport *ETSI* [**?**]. Il prend en compte des besoins d'observation raisonnables (la fréquence d'observation est ajustable selon les *SLA*). Pour une distribution homogène de trafic, il fonctionne avec un coût acceptable en termes de ressources utilisées sur chaque routeur d'entrée, rapportée à la quantité de données y transitant. En revanche, l'efficacité du *piggybacking* se rapproche de celle d'une remontée en trames dédiées lorsque les conditions de trafic ne lui sont pas favorables (espace disponible réduit).

5.6 Perspectives

Dans les réseaux de l'*IoT*, l'observation de réseau permet à l'opérateur de vérifier le respect de *SLA* et de maintenir le bon fonctionnement du réseau.

Ce chapitre vise à proposer des mécanismes d'observation de *KPI* pour vérifier les *SLA*. Notre contribution appelle à être évaluée de manière approfondie :

1. sur sa robustesse : quel impact ont les pertes de trame sur la collecte des informations ? Le mécanisme passe-t-il à l'échelle ?
2. sur l'impact d'arrivées de trames de taille variable sur les performances des mécanismes.

Nous envisageons la validation de l'observation orientée *SLA* par des expérimentations (sur *OpenWSN*, sur *FIT/IoT-Lab*), puis l'intégration des mécanismes en standardisation.

Ensuite, le maintien des performances du réseau met en perspective la possibilité de détecter les défaillances de nœuds et les pertes d'énergie [**?**]. Pour établir un diagnostic de défaillance localisée du réseau, l'opérateur doit savoir quel chemin a été suivi par chaque trame [**?**]. De la même manière, les liens non fiables doivent être détectés pour être écartés de la topologie de routage [**?**].

L'observation des paramètres de bout-en-bout du réseau permet à l'opérateur de remplir sa base de données de mesures, le ***Service Registry***. Il peut ainsi analyser le comportement du réseau et anticiper. Le prochain chapitre traite de l'allocation de ressources, sur la base d'estimation des *PER* des liens.

Chapitre 6

Allocation déterministe pour l'*IoT* opéré

Nous avons vu comment la collecte d'information pour l'Internet des Objets (*IoT*), par le biais de réseaux radio multi-sauts est sujette au problème de la fiabilité : les liens pâtissent des interférences et subissent des pertes de trames.

Dans le chapitre précédent, nous avons proposé une solution de remontée d'information d'observation réduisant l'impact de l'observation sur la consommation de ressources. Ici nous proposons plusieurs algorithmes permettant d'allouer des ressources *FTDMA* (e.g. pour des réseaux utilisant *6TiSCH*) tout en tenant compte des contraintes de fiabilité et de délai attendues par les différentes applications.

Nous mettons en place un choix de routage multi-saut par flux permettant un équilibrage de charge et ainsi la préservation de la durée de vie du réseau. Nous proposons deux mécanismes d'allocation (uniforme vs saut-par-saut), permettant des retransmissions. Ils permettent de respecter pour chaque flux applicatif une contrainte de taux de livraison de bout-en-bout. En réduisant le nombre de cellules nécessaires aux retransmissions et en les répartissant sur les relais, nous rendons notre allocation fiable et efficace en énergie. Enfin, nous proposons *KPI-Aware Scheduling Algorithm* (*KAUSA*), un algorithme d'ordonnancement facilitant le respect d'exigences de délai et de livraison, en respectant la contrainte de mémoire des nœuds.

Afin de valider notre approche, nous intégrons nos mécanismes de prise en compte des retransmissions dans l'algorithme qui fait référence dans l'état de l'art, *Traffic-Aware Scheduling Algorithm (TASA)*. Nous montrons au moyen de simulations les gains de nos approches en termes de fiabilité et leurs coûts en termes de quantité de ressources allouées.

6.1 Introduction

Nous avons vu comment distinguer le trafic des clients en différents flux. De cette manière, l'opérateur fournit des garanties de qualité de service (*QoS*) multi-flux. Nous supposons que l'opérateur alloue globalement les ressources pour chaque client. L'ordonnancement centralisé permet une meilleure utilisation de la capacité du réseau. Les *SLA*, définis de manière centralisée, précisent à la fois la quantité maximale de trafic généré et les deux indicateurs clés de performance (*KPI*) : le délai de bout-en-bout et le taux de livraison (*PDR*).

6.1.1 Contexte : l'ordonnancement des ressources *FTDMA*

Avec l'*IoT*, la densité des nœuds radio et le trafic de données augmentent dans les villes, en accroissant l'occupation du canal. Les communications radio entre les nœuds sont soumises à des collisions, des interférences, qui ont un impact négatif sur le taux d'erreur (*PER*) [**?**]. Pour garantir une fiabilité minimum à différents flux clients, l'opérateur doit maîtriser le *PDR* de bout-en-bout sur chaque route empruntée jusqu'à une passerelle. Celui-ci est estimé à partir du *PER* de chaque saut.

Le groupe de travail *IETF 6TiSCH* fournit une technologie *FTDMA* [**?**] où les transmissions de trames sont planifiées en temps et en fréquence dans des cellules. Le choix des cellules, sous la forme d'un échéancier temps-fréquence, constitue l'ordonnancement que l'opérateur met en œuvre. La cellule est la ressource minimale que nous allouons, correspondant à la transmission d'une trame et de l'acquittement correspondant. Les cellules permettent d'éviter les collisions en dédiant leur usage dans l'échéancier *6TiSCH*.

Ensuite, *6TiSCH* permet d'associer à chaque cellule un *track*. Chaque *track* correspond à un ensemble de cellules dédiées à un flux. Les trames d'un flux donné sont transmises sur les cellules marquées avec l'identifiant de *track* (le *track ID*) leur correspondant. Par conséquent, avec *6TiSCH* l'opérateur est en mesure d'allouer des ressources à chaque flux en isolation, et de quantifier la capacité du réseau restant disponible.

Le *Traffic-Aware Scheduling Algorithm (TASA)* [**?**] constitue un des principaux travaux pour calculer centralement un échéancier *FTDMA*. Les échéanciers sont compacts et peuvent être construits à partir d'un graphe de routage (e.g. fourni par *RPL*) [**?**].

TASA se base sur le trafic (supposé connu) généré localement sur les nœuds. C'est une solution centralisée pertinente pour l'ordonnancement dans les réseaux *6TiSCH* [**?**].

6.1.2 Énoncé du problème et propositions

En utilisant un arbre de routage (pour chaque nœud, un seul prochain saut *parent* est considéré, e.g. le parent préféré choisi par *RPL*), *TASA* fournit des échéanciers optimaux en termes de consommation globale de ressources [**?**]. Ainsi, l'approche de *TASA* restreint l'ensemble des routes utilisées. La charge n'est pas correctement répartie sur les nœuds, l'occupation de la mémoire de certains nœuds augmente. Le réseau devient moins efficace.

En outre, *TASA* ne considère pas de longs messages fragmentés afin d'être transmis sur différents blocs temps-fréquences. Dans ce cas, le délai de bout-en-bout des messages est important, car les fragments sont ordonnancés de manière indépendante. Selon la charge de trafic sur les nœuds intermédiaires, chaque fragment sera gardé en mémoire un temps différent.

TASA ne prend pas en compte :

— la robustesse aux pertes de trames : le taux de livraison des trames (*PDR*) ne fait pas partie des paramètres de l'algorithme ;

— la capacité d'adaptation à une augmentation du trafic : l'algorithme alloue seulement le nombre de cellules nécessaires selon la prévision de trafic ;

— la fragmentation des messages longs : le succès de la livraison dépend de celle de tous ses fragments.

Les opérateurs de réseau doivent tenir compte des exigences de trafic, de la topologie radio et de la qualité des liens pour organiser les communications. Dans ce chapitre, nous améliorons la proposition de *TASA* en allouant des ressources supplémentaires pour les retransmissions de trames, afin d'augmenter la probabilité de succès d'une livraison. Les retransmissions permettent de satisfaire une contrainte de *PDR* donnée, mais augmentent le délai et la charge de trafic et influent sur la consommation d'énergie.

De plus, nous proposons de satisfaire plusieurs contraintes de fiabilité (une pour chaque flux). Cela concorde avec la notion de *track* dans *6TiSCH* : chaque trame présente en mémoire d'un nœud sera transmise à son prochain saut en choisissant les cellules de transmission en fonction de leur identifiant (*track ID*). De cette façon, l'isolation de flux est respectée et les trames sont (re)transmises en utilisant les cellules allouées.

Nous proposons d'abord un mécanisme de *sur-dimensionnement* uniforme sur un chemin : le nombre de cellules ajoutées pour les retransmissions est constant pour tous les liens du chemin considéré. Ce mécanisme permet d'atteindre un *PDR* donné, mais avec une surcharge importante.

Ensuite, nous fournissons un mécanisme de *sur-dimensionnement* saut-par-saut qui choisit flux par flux le nombre de blocs temps-fréquences ajoutés à l'échéancier pour les transmissions et retransmissions. Nous adaptons le nombre de cellules ajoutées pour les retransmissions au *PER* de chaque lien et à la charge déjà allouée sur les nœuds. Nous intégrons les deux mécanismes à *TASA* et comparons les résultats par simulation.

Notre contribution est multiple :

1. nous fournissons un premier mécanisme pour calculer le nombre minimal de cellules, réparties uniformément sur chaque lien d'un chemin, qui permette suffisamment de retransmissions pour respecter les taux de livraison attendus ;
2. nous fournissons un second mécanisme qui calcule le nombre minimal de cellules, distribuées saut-par-saut, qui permette suffisamment de retransmissions pour respecter les taux de livraison attendus ;
3. nous mettons en œuvre ces deux mécanismes de sur-dimensionnement sur un échéancier *TASA* pour améliorer ses performances ;
4. nous proposons une métrique permettant d'attribuer une route à chaque flux, en fonction de la charge de trafic et des contraintes de fiabilité. En particulier, nous adoptons une approche multi-chemin : différents flux de la même source peuvent utiliser des chemins différents ;
5. nous décrivons notre proposition (*KPI-Aware Scheduling Algorithm* (*KAUSA*)) d'algorithme d'ordonnancement à contraintes de *KPI*, prenant en compte la fragmentation des messages. Nous fournissons une technique de retour en arrière (ou *backtracking*) qui améliore ses performances ;
6. nous démontrons par simulation la performance de notre approche en termes de satisfaction de *SLA* et de coût en nombres d'allocations.

Le chapitre est organisé comme suit : nous détaillons l'état de l'art en termes d'algorithmes d'ordonnancement (prochaine Section 6.1.3) puis nous décrivons notre modèle (Section 6.2). Ensuite, nous détaillons nos contributions en Section 6.3 et 6.4 ; nous évaluons enfin les performances de nos solutions (Section 6.5) et concluons (Section 6.6).

6.1.3 État de l'art : l'ordonnancement *FTDMA* à contraintes de *QoS*

Afin de satisfaire les *SLA*, l'opérateur doit gérer et ordonnancer le trafic client [**?**]. L'échéancier temps-fréquence doit prendre en compte les *KPI* par flux tout en maximisant la durée de vie du réseau. L'ordonnancement peut être réalisé de manière centralisée ou distribuée sur les nœuds du réseau.

Les algorithmes décentralisés ou distribués sont réactifs (e.g. en cas de mobilité, ou de défaillance de nœud) et permettent de limiter le surcoût de contrôle en favorisant des décisions locales d'ordonnancement. Dans les algorithmes *D-SAR* [**?**] et *CFDS* [**?**], les nœuds sources demandent des ressources de communication avec qualité de service à chaque fois

qu'ils ont du trafic à transmettre. En fonction de la bande passante et de la qualité de service requise par chaque nœud source, l'allocation est construite le long du chemin vers la destination. Les exigences de qualité de service sont supposées dynamiques, variables pour chaque nouvelle génération de trafic. Ni *D-SAR* ni *CFDS* ne fournissent des garanties stables de *QoS* pour un flux donné, parce qu'ils sont conçus pour des réservations temporaires.

Phung et al. proposent un ordonnanceur multi-canal, distribué, fonctionnant par apprentissage et par adaptation aux changements sur le réseau [**?**]. Étant local à chaque nœud, l'apprentissage autonome ne fournit pas de garanties de bout-en-bout, nécessaires pour satisfaire les *SLA*.

TASA alloue de manière centralisée des ressources pour une charge de trafic donnée sur chaque nœud [**?**]. Sur la base d'un graphe de conflits à 2 sauts, *TASA* planifie les ressources en traitant en premier les sous-parties les plus chargées de l'arbre de routage. La priorité est donnée aux nœuds qui ont le plus grand nombre de trames à transmettre. Dans ce cas, la taille de l'échéancier construit par *TASA* est minimale [**?**]. Les échéanciers compacts permettent d'optimiser à la fois la consommation d'énergie et le délai. *TASA* prévoit des transmissions simultanées pour des paires de nœuds distinctes, sur des canaux orthogonaux. Cependant, *TASA* ne considère pas les pertes de trames.

DETAS [**?**] propose une solution décentralisée d'ordonnancement, à partir des mêmes hypothèses que pour *TASA*. L'algorithme alterne sur chaque relais les cellules d'émission et de réception. Lorsqu'un nœud reçoit une trame, il la transmet dans la cellule suivante. Ainsi, si une trame est perdue, le reste de la bande passante est gaspillée le long du chemin. *DETAS* ne considère ni le respect de *KPI* par flux, ni l'équilibrage de charge.

Les déploiements industriels ont également de fortes contraintes sur leur trafic [**?**]. *Pöttner et al.* [**?**] proposent une méthode d'ordonnancement pour collecter des données dans une raffinerie de pétrole en considérant des délais de livraison critiques (e.g. délai attendu de 3 s). Ils subdivisent le réseau en petits ensembles de nœuds (pas plus de 24 nœuds dans la pratique). L'algorithme fournit à la fois un échéancier et les puissances de transmission nécessaires, pour une contrainte de fiabilité donnée. Cette approche ne favorise pas le maintien de la durée de vie du réseau (pas d'équilibrage de charge, différentes puissances d'émission). De plus, les auteurs ne considèrent pas le cas de plusieurs flux concurrents avec différentes valeurs de *KPI*.

Dobslaw et al. proposent *SchedEx* [**?**], une extension d'ordonnanceur qui modifie un échéancier afin de garantir une fiabilité de bout-en-bout minimale sur l'ensemble du réseau. Les auteurs calculent le nombre de retransmissions nécessaires pour toutes les trames, à chaque lien de l'arbre de routage. Ce nombre attendu de retransmissions est défini en fonction de la charge d'un lien radio et de sa fiabilité. En d'autres termes, *Schedex* ne garantit pas l'isolation de flux avec les exigences de *PDR* différenciées.

Yang et al. [**?**] construisent un échéancier optimal pour des flux contraints par le délai : de nouvelles cellules sont insérées dans l'échéancier, si la fiabilité de bout-en-bout est insuffisante et jusqu'à ce que la contrainte de délai ne soit plus remplie. *Yigit et al.* [**?**] ont étudié l'impact du routage sur l'ordonnancement : utiliser des liens non fiables augmente le nombre de cellules requises pour atteindre une fiabilité minimale.

Dans ce chapitre, nous proposons de prendre en compte à la fois la fragmentation et les pertes de trames. Notre algorithme d'ordonnancement garantit une fiabilité de bout-en-bout donnée pour chaque flux, en allouant un nombre précis de cellules pour chaque fragment et ses retransmissions. Notre solution, *KAUSA*, offre des garanties par flux et favorise l'équilibrage de charge sur le réseau tout en allouant efficacement les cellules.

6.2 Modèle de réseau multi-saut opéré

Afin de construire des échéanciers *FTDMA* fiables, nous modélisons le comportement du réseau tant pour l'ordonnanceur que pour les deux algorithmes de sur-dimensionnement.

Dans cette section, nous précisons les caractéristiques du réseau et les paramètres considérés. Le Tab. 6.1 résume les paramètres et les notations que nous utilisons dans notre modèle.

6.2.1 Modèle de nœud et de topologie

Nous modélisons le réseau de l'*IoT* comme un ensemble de nœuds sans fil échangeant des messages en multi-saut. Nous supposons que la topologie est connue et statique entre deux ordonnancements. Chaque nœud a une valeur de rang (e.g. calculée par *RPL*) qui représente une distance abstraite à une passerelle.

Chaque nœud dispose d'une interface radio *half-duplex*. Il peut stocker un nombre limité de trames dans sa mémoire. Nous considérons 3 types distincts de nœuds, tels que définis au Chapitre 4 :

1. **les nœuds feuilles :** les nœuds capteurs qui génèrent seulement le trafic client ;
2. **les relais :** les routeurs d'entrée et les routeurs intermédiaires qui transmettent le trafic client ;
3. **les passerelles :** les récepteurs finaux qui collectent le trafic client.

Chaque nœud feuille exécute une ou plusieurs applications qui génèrent des messages de taille constante (qui peuvent être fragmentés en plusieurs trames). Nous considérons uniquement le trafic de collecte, convergeant vers les passerelles. Pour chaque application, un nœud feuille crée un *flux* de messages dirigés vers une des passerelles.

6.2.2 Modèle de propagation

Comme nous l'avons décrit au Chapitre 2, nous utilisons le taux d'erreur (*PER*) pour caractériser la qualité des liens. Le *PER* représente le ratio de transmissions de trames non reçues sur un lien donné. Nous supposons qu'un mécanisme d'observation fournit des mises à jour régulières des informations de qualité des liens. Ces mises à jour permettent d'exécuter l'algorithme d'ordonnancement afin d'adapter l'échéancier aux changements (nous considérons que le réseau est suffisamment stable). Nous considérons que le *PER* est invariant dans le temps à l'échelle de l'ordonnancement et qu'il ne dépend pas du *Channel Offset* choisi puisque le saut de fréquences est utilisé.

Les interférences sont hétérogènes : en fonction de la concentration des nœuds et la densité du trafic, le *PER* évolue [**?**]. Nous modélisons un environnement hétérogène utilisant le modèle de propagation classique (*simple pathloss*) [**?**]. Le bruit (*noise*) et le masquage (*shadowing*) sont modélisées par des variables aléatoires de loi normale.

6.2.3 Modèle de communications et d'allocation de ressources

Les communications suivent un échéancier *FTDMA*. Une allocation correspond au choix d'un bloc temps-fréquence ou cellule, pour la transmission d'un fragment entre deux nœuds voisins. L'échéancier est périodique : les cellules choisies sont répétées dans le temps, jusqu'à une nouvelle modification de l'échéancier.

En fonction de sa taille, chaque message applicatif est fragmenté en une ou plusieurs trames qui transitent séparément, chacune sur une seule cellule. Par conséquent, chaque

message applicatif est constitué d'un ensemble de fragments, leur quantité dépendant de la taille du message.

Nous isolons chaque flux : chaque cellule est allouée pour un flux unique. L'ensemble des cellules affectées à un flux constitue un *track* identifié par son identifiant (le *track ID*).

Nous considérons un algorithme d'ordonnancement centralisé qui prend comme paramètres d'entrée deux éléments d'information topologique :

1. un graphe de routage acyclique. Pour chaque flux, nous construisons un chemin sans boucle depuis le nœud feuille vers une passerelle. Dans le cas de *TASA*, le graphe de routage peut être calculé en utilisant les informations de *RPL* [**?**]. Le modèle est également applicable à un algorithme de routage multi-chemin ;
2. le *Packet Error Rate* (*PER*) de chaque lien. Un ensemble de liens interfère s'il y a une perte d'information lorsque les transmissions ont lieu dans le même intervalle de temps et sur le même canal. Nous devons donc construire le graphe de conflits, c'est-à-dire les informations sur l'ensemble des liens interférents.

   Dans le cas où aucune autre information n'est donnée par l'observation sur les interférences, nous construisons le graphe de conflits en considérant que les nœuds à plus de trois sauts ne sont pas interférents. En effet, nous supposons que les interférences sur un même canal sont limitées à quelques sauts (e.g. 1 ou 2) [**?**].

Notre algorithme calcule le nombre de cellules d'un *track*, à chaque lien, pour un flux donné. L'allocation finale pour un *track* est la somme des allocations pour chaque message du flux. L'ordonnanceur alloue suffisamment de cellules pour la transmission des fragments de chaque message de bout-en-bout. L'ordonnanceur alloue des cellules supplémentaires afin de permettre des retransmissions saut-par-saut [**?**].

L'opérateur fournit des garanties uniquement sur les flux qui respectent leur spécification dans le *SLA* [**?**]. Nous considérons un trafic hétérogène, mélangeant plusieurs applications [**?**]. Nous faisons l'hypothèse que le trafic de chaque application est limité par une quantité périodique de fragments. Nous réalisons donc l'allocation de ressources pour des trafics périodiques. Pour chaque flux, le nœud source génère un certain nombre de messages de taille fixe à chaque période. Le nœud feuille garde en mémoire les fragments correspondants jusqu'à leur première cellule de transmission. Tous les fragments doivent parvenir à une passerelle en moins de temps que la taille en intervalles de l'échéancier.

6.2.4 Définition du trafic

Chaque nœud est capable de détecter la perte d'un fragment (s'il ne reçoit pas d'acquittement). Dans ce cas, il retransmet le fragment au prochain saut sur la prochaine cellule disponible associée au même *track ID*.

Le nombre de retransmissions saut-par-saut est limité :

— pour chaque fragment, par saut : $n_{rtx}^{max(frag)}(f)$, (e.g. 5 fragments) : contrainte du protocole *MAC* ;

— pour chaque message, par saut : $n_{rtx}^{max}(f)$, (e.g. 15 fragments) : choix d'ordonnancement. Nous réduisons ainsi le nombre d'allocations pour un *track* donné, et nous évitons la préemption de toutes les ressources.

Par exemple, nous allouons pour un message de 4 fragments au maximum 19 cellules (4 cellules de transmission, 15 de retransmission), et chaque fragment pourra en utiliser jusqu'à 6 (1 transmission et 5 retransmissions) s'il en reste parmi les 19.

1. Cette contrainte est dérivée de la valeur de $PDR_{msg}^{sla}(f)$ contractualisée dans le *SLA*.

TABLE 6.1 – Paramètres du modèle de réseau

Variable	Explication
$f \in F$	Un flux f parmi l'ensemble des flux F
$path(f)$	Chemin associé au flux f
$l_1, l_2, \ldots, l_{gw}$	Ensemble des liens d'un chemin vers une passerelle
$n_{msg}(f)$	Nombre de messages par *slotframe* pour un flux f
$n_{frag}(f)$	Nombre de fragments par message
$per(l)$	Packet Error Rate (PER) sur un lien l donné
$n_{rtx}^{max}(f)$	Nombre maximal de retransmissions par saut et message
$n_{rtx}^{max(frag)}(f)$	Nombre maximal de retransmissions par saut et fragment
$PDR_{msg}^{sla}(f)$	PDR de bout-en-bout contractualisé pour les messages de f
$PDR_{msg}(f)$	PDR de bout-en-bout pour les messages de f
$PDR_{msg}^{hop}(f,l)$	PDR des messages de f au lien l
$pdr_{frag}^{sla}(f)$	PDR de bout-en-bout, objectif attendu pour les fragments de f [1]
$pdr_{frag}(f)$	PDR de bout-en-bout pour les fragments de f, sans retransmissions
$pdr_{frag}^{rtx}(f,k)$	PDR de bout-en-bout pour les fragments de f, max. k transmissions
$DELAY_{msg}^{sla}(f)$	Contrainte : délai de bout-en-bout attendu pour les messages de f
$n_{cell}(A)$	Nombre total de cellules allouées sur un nœud A
$n_{cell}(l)$	Nombre total de cellules allouées sur un lien l
$load(l)$	Charge d'allocation sur un lien l donné à une étape de l'algorithme
$alloc_f$	Liste des nombres d'allocations pour un message de f sur $path(f)$
$alloc_f^{l_i}$	Nombre d'allocations du *i-ème* lien de $path(f)$: élément de $alloc_f$
$track_f$	Nombre de cellules constituant le *track* de f le long de $path(f)$
$track_f^{l_i}$	Nombre de cellules allouées pour f sur le *i-ème* lien de $path(f)$
$n_{rtx}^{unif}(f)$	Nombre de cellules de retransmissions pour un message de f, allouées par le mécanisme uniforme
$Sol(f)$	Ensemble des solutions de l'algorithme saut-par-saut

Prise en compte des retransmissions La première approche de calcul d'opportunités de retransmissions, est uniforme le long du chemin d'un flux. À chaque saut, un nombre identique de cellules est ajouté afin de permettre aux nœuds de retransmettre un fragment lorsqu'une première transmission a échoué. Nous modélisons ainsi les retransmissions uniformes comme des *duplicata* d'un fragment, générés au niveau du nœud source de chaque flux. Ils transitent tout le long du chemin exactement comme les fragments. Avec cette hypothèse, l'ordonnanceur couvre le cas où les fragments sont retransmis un nombre constant de fois à chaque nœud.

La seconde approche considère des allocations de cellules saut-par-saut. Le nombre d'allocations, variable à chaque saut, est calculé en tenant compte des opportunités de retransmission nécessaires afin de respecter un PDR en termes de probabilités pour chaque message.

Par exemple, l'algorithme donne 7 cellules pour un message de 4 fragments, sur un saut donné :

— un message x utilisera 5 des 7 cellules : 1 transmission pour les 3 premiers fragments et 2 cellules pour le dernier ;

— un autre message y utilisera 6 cellules pour le premier fragment et 1 pour le deuxième : la livraison du message échoue puisque tous les fragment ne sont pas transmis ;

— pour l'ordonnancement, l'algorithme considère l'allocation pour 4 fragments avec au total 3 retransmissions, pour chaque message.

6.2.5 Expression de la contrainte de fiabilité

Chaque application a sa propre contrainte de fiabilité, exprimée sous la forme d'un taux de livraison attendu de bout-en-bout, notée $PDR^{sla}_{msg}(f)$. Pour chaque flux f, le *PDR* est le rapport entre le nombre de messages reçus au niveau de la passerelle, et le nombre de messages envoyés par la source. Un échéancier satisfait la contrainte de fiabilité pour un flux donné si le *PDR* minimum est atteint lorsque calculé sur une fenêtre de temps fixe (e.g. une minute/une heure/un jour), comme convenu dans le *SLA*.

Nous considérons qu'un message applicatif est perdu quand au moins un de ses fragments est perdu. En faisant l'hypothèse que les pertes de fragments ne sont pas corrélées, on déduit aisément du *PDR* applicatif le *PDR* de bout-en-bout attendu par fragment :

$$\forall f \in F,\ \left(pdr^{sla}_{frag}(f)\right)^{n_{frag}(f)} = PDR^{sla}_{msg}(f) \tag{6.1}$$

Dans l'Eq. (6.1), F est l'ensemble des flux, $n_{frag}(f)$ le nombre de fragments des messages du flux f, $pdr^{sla}_{frag}(f)$ le *PDR* attendu par fragment, et $PDR^{sla}_{msg}(f)$ le *PDR* applicatif attendu pour le flux f (cf. Tab. 6.1).

6.3 Prise en compte des retransmissions : méthodes de sur-dimensionnement

6.3.1 Une première approche : calcul de sur-dimensionnement uniforme

Dans cette section, nous fournissons une méthode pour calculer le nombre minimal de cellules de retransmission $n^{unif}_{rtx}(f)$ qui, ajouté uniformément à l'échéancier sur chaque saut le long d'un chemin, permet de vérifier une contrainte de *PDR*.

Expression du *PDR* par fragment avec des retransmissions Étant donné que le *PDR* d'un lien se déduit aisément du *PER* ($pdr(l)+per(l) = 1$) [**?**], nous pouvons facilement calculer $pdr_{frag}(f)$, le *PDR* de bout-en-bout d'un fragment sans retransmission pour un flux f :

$$pdr_{frag}(f) = \prod_{l \in path(f)} (1 - per(l)) \tag{6.2}$$

Dans l'Eq. (6.2), $path(f)$ est l'ensemble des liens qui relient la source du flux à une passerelle. Sans opportunités de retransmission, l'échéancier ne garantit pas que $pdr_{frag}(f) \geq pdr^{sla}_{frag}(f)$ et donc que le *PDR* est satisfait pour le flux f.

L'Eq. (6.3) donne $pdr^{rtx}_{frag}(f,k)$, le *PDR* de bout-en-bout d'un fragment avec k opportunités de transmission ($k-1$ cellules de retransmission) pour un flux f.

$$pdr^{rtx}_{frag}(f,k) = 1 - (1 - pdr_{frag}(f))^k \tag{6.3}$$

L'Eq. (6.3) exprime le succès d'au moins une livraison de bout-en-bout, sur k tentatives indépendantes.

Satisfaction de la contrainte de fiabilité avec retransmissions Afin de calculer le nombre minimum de cellules de retransmission nécessaires pour un message, le long d'un chemin, nous supposons que les pertes de fragments y sont uniformément distribuées. Nous répartissons donc uniformément les opportunités de transmissions entre les fragments. Nous divisons $n_{rtx}^{unif}(f)$ par $n_{frag}(f)$ comme suit :

$$n_{rtx}^{unif}(f) = q \cdot n_{frag}(f) + r, \quad 0 \leq r < n_{frag}(f) \quad \text{avec} \quad n_{rtx}^{unif}(f) \leq n_{rtx}^{max}(f) \tag{6.4}$$

Dans l'Eq. (6.4), q et r sont le quotient et le reste de la division euclidienne de $n_{rtx}^{unif}(f)$ par $n_{frag}(f)$. De cette manière :

— $n_{frag}(f) - r$ fragments obtiennent chacun 1 cellule de transmission et q cellules de retransmissions le long du chemin (Eq. (6.3)). Autrement dit, chaque fragment peut être transmis $q + 1$ fois ;

— r fragments obtiennent chacun 1 cellule de transmission et $q + 1$ cellules de retransmissions le long du chemin (Eq. (6.3)). Autrement dit, chaque fragment peut être transmis $q + 2$ fois.

On a ainsi distribué le nombre entier de retransmissions sur les fragments. La division euclidienne permet de répartir le nombre exact de retransmissions à 1 près (certains fragments sont retransmis q fois, d'autres $q + 1$ fois).

Le succès de la transmission d'un message correspond aux succès des transmissions de tous ses fragments. À partir de l'expression de la contrainte de *PDR* par fragment (Eq. (6.1)) et des équations précédentes (Eq. (6.2) et Eq. (6.3)), nous calculons le *PDR* de bout-en-bout pour les messages de chaque flux f, $PDR_{msg}(f)$:

$$PDR_{msg}(f) = \left(pdr_{frag}^{rtx}(f, q+1)\right)^{n_{frag}(f)-r} \left(pdr_{frag}^{rtx}(f, q+2)\right)^{r} \tag{6.5}$$

Et en développant avec l'Eq. (6.3) :

$$PDR_{msg}(f) = \left(1 - (1 - pdr_{frag}(f))^{q+1}\right)^{n_{frag}(f)-r} \left(1 - (1 - pdr_{frag}(f))^{q+2}\right)^{r} \tag{6.6}$$

Dans l'Eq. (6.6), $pdr_{frag}(f)$ est le *PDR* de bout-en-bout par fragment sans retransmission (Eq. (6.2)). Dans l'Eq. (6.5), $pdr_{frag}^{rtx}(f, k)$ est le *PDR* de bout-en-bout d'un fragment avec k transmissions (Eq. (6.3)).

Notre objectif est de trouver le $n_{rtx}^{unif}(f)$ minimal qui permette de satisfaire la contrainte de fiabilité :

$$PDR_{msg}(f) \geq PDR_{msg}^{sla}(f) \tag{6.7}$$

Le produit des *PDR* de bout-en-bout de chaque fragment, avec deux nombres différents de retransmissions (q ou $q+1$) rend le calcul direct complexe. L'algorithme commence donc avec 0 cellule de retransmission, $n_{rtx}^{unif}(f) = 0$. Nous calculons $PDR_{msg}(f)$ (Eq. (6.6) et Eq. (6.4)) en incrémentant $n_{rtx}^{unif}(f)$ jusqu'à ce que la contrainte de fiabilité de l'Eq. (6.7) soit respectée, ou que nous atteignions le nombre maximal de retransmissions $n_{rtx}^{max}(f)$. Dans ce cas, la contrainte de *PDR* n'est pas satisfaite, nous fixons $n_{rtx}^{unif}(f)$ à la valeur maximale et nous continuons avec le flux suivant. L'algorithme garde en mémoire les valeurs de chaque calcul intermédiaire pour l'utiliser pour d'autres flux.

6.3.2 Une seconde approche : calcul de sur-dimensionnement saut-par-saut

Nous présentons maintenant une méthode pour calculer pour chaque flux le nombre minimal saut-par-saut d'opportunités de retransmission pour chaque message. Notre approche permet de réduire le nombre total d'allocations de cellules sur le chemin, tout en satisfaisant l'exigence de *PDR* de bout-en-bout pour chaque flux.

Expression du *PDR* de bout-en-bout par message avec retransmissions Notre contrainte de fiabilité par flux reste la même que dans l'Eq. 6.7, mais dans la suite nous exprimons $PDR_{msg}(f)$ avec des cellules de retransmission saut-par-saut.

Comme précédemment, nous considérons chaque saut indépendant et aussi chaque transmission de fragment indépendante. Nous avons maintenant différents nombres de cellules de retransmission par saut le long du chemin, de sorte que l'Eq. (6.3), donnant le *PDR* d'un fragment de bout-en-bout ne convient plus.

Notre objectif consiste à équilibrer la charge sur les liens. Notre approche permet de réduire le nombre total d'allocations de cellules et minimise la charge maximale d'allocation sur le chemin.

Nous traitons l'objectif d'équilibrage de charge flux par flux. En effet, nous supposons que les messages du même flux ont une taille constante. Une allocation message par message augmenterait alors inutilement la complexité et le temps d'exécution de l'algorithme.

Pour chaque saut donné, le succès de la transmission d'un message dépend du succès des transmissions de tous ses fragments. Au moins un succès doit avoir lieu pour chaque fragment du message. Un fragment transmis avec succès n'est pas retransmis. Autrement dit, au moins $n_{frag}(f)$ succès sont nécessaires parmi $alloc_f^l$ tentatives (voir le Tab. 6.1 pour les notations). Cette condition est exprimée par une fonction de distribution cumulative partielle de la formule du binôme :

$$\forall f \in F, \forall l \in path(f), PDR_{msg}^{hop}(f,l) = \sum_{k=0}^{n-n_{frag}(f)} \binom{n}{k} p^k (1-p)^{n-k}$$
$$\text{avec } n = alloc_f^l,\ p = per(l) \quad (6.8)$$

Dans l'Eq. (6.8), $PDR_{msg}^{hop}(f,l)$ est le *PDR* saut-par-saut pour un message d'un flux donné f sur le lien l.

En supposant chaque saut indépendant, notre contrainte peut s'exprimer comme le produit le long du chemin de chaque expression par saut :

$$\forall f \in F, \quad PDR_{msg}(f) = \prod_{l \in path(f)} PDR_{msg}^{hop}(f,l) \quad (6.9)$$

Satisfaire la contrainte de fiabilité avec un minimum de retransmissions saut-par-saut L'algorithme doit trouver l'ensemble le mieux équilibré de cellules de retransmission permettant de respecter la contrainte de fiabilité (Eq. (6.7)).

Nous notons $alloc_f$ la liste des nombres d'allocations le long du chemin, pour un flux f :

$$alloc_f = \left\{ alloc_f^l, l \in path(f) \right\} \quad (6.10)$$

Pour chaque lien du chemin, le nombre de cellules du *track* de f correspond à la somme des nombres d'allocations de chaque message :

$$track_f = n_{msg}(f) \cdot alloc_f \qquad \text{et} \qquad \forall l \in path(f), track_f^l = n_{msg}(f) \cdot alloc_f^l \quad (6.11)$$

Le résultat de l'algorithme, $alloc_f$, répond au problème si tous les nombres d'allocations $alloc_f^{l_i}$ se situent entre le nombre de fragments et le maximum de transmissions :

$$\forall l \in path(f),\ n_{frag}(f) \leq alloc_f^l \leq n_{frag}(f) + n_{rtx}^{max}(f) \quad (6.12)$$

Nous notons $Sol(f)$ l'ensemble des allocations vérifiant l'Eq. (6.7) et l'Eq. (6.12). $Sol(f)$ est l'ensemble des solutions de l'approche de dimensionnement saut-par-saut.

Algorithme 1 : Calcul des nombres d'allocations pour un flux donné f

Données : $f \in F$, $path(f)$, $n_{msg}(f)$, $n_{frag}(f)$, $n_{rtx}^{max}(f)$ $\forall l \in path(f), n_{cell}(l)$, $PDR_{msg}^{sla}(f)$
Résultat : $\forall l \in path(f), load(l)$, $alloc_f \in Sol(f)$ vérifiant l'Eq. (6.14) et l'Eq. (6.7)

1 $treated \leftarrow \emptyset$;
2 $\forall l \in path(f), alloc_f^l \leftarrow n_{frag}(f) + n_{rtx}^{max}(f)$;
3 $\forall l \in path(f), load(l) \leftarrow \left(n_{cell}(l) + n_{msg}(f) \cdot alloc_f^l\right)$;
4 calculer $PDR_{msg}(f)$ en appliquant l'Eq. (6.9);
5 **si** $PDR_{msg}(f) \geq PDR_{msg}^{sla}(f)$ **alors**
6 **tant que** $\exists l \in path(f), l \notin treated$ **faire**
7 **répéter**
8 choisir un $l_{max}(f) \in path(f), l_{max}(f) \notin treated$ tel que $load(l_{max}(f)) = \max_{l \in path(f), l \notin treated}(load(l))$;
9 $alloc_f^{l_{max}(f)} \leftarrow alloc_f^{l_{max}} - 1$;
10 calculer $PDR_{msg}(f)$ en appliquant l'Eq. (6.9);
11 calculer $load(l_{max}(f))$ en appliquant l'Eq. (6.13);
12 **jusqu'à** $PDR_{msg}(f) < PDR_{msg}^{sla}(f)$ **ou** $alloc_f^{l_{max}(f)} < n_{frag}(f)$;
13 $alloc_f^{l_{max}(f)} \leftarrow alloc_f^{l_{max}} + 1$;
14 calculer $PDR_{msg}(f)$ en appliquant l'Eq. (6.9);
15 calculer $load(l_{max}(f))$ en appliquant l'Eq. (6.13);
16 $treated \leftarrow treated \cup \{l_{max}(f)\}$;
17 **fin**
18 $\forall l \in path(f), n_{cell}(l) \leftarrow load(l)$
19 **fin**

En faisant l'hypothèse que le nombre de retransmissions par fragment pour chaque message est limité à un maximum (Eq. (6.12)), notre objectif est de distribuer les allocations sur les liens à l'échelle du réseau. La charge d'allocation est la somme des cellules déjà allouées à un lien donné et des cellules allouées pour le flux f :

$$\forall l \in path(f),\ load(l) = \left(n_{cell}(l) + n_{msg}(f) \cdot alloc_f^l\right) \tag{6.13}$$

Afin de réduire globalement les différences entre les nombres d'allocations, nous cherchons à minimiser la charge maximale sur les liens de chaque chemin :

$$\min_{alloc_f \in Sol(f)} \left(\max_{l \in path(f)} load(l) \right) \tag{6.14}$$

Pour ce faire, nous proposons un algorithme glouton dont la série d'opérations est similaire à celle de l'algorithme *Reverse-Delete* dédié à la recherche d'arbre couvrant de poids minimal [**?**]. Notre algorithme donne une solution flux par flux au problème de l'Eq. (6.14).

Algorithme glouton inverse de sur-dimensionnement En premier lieu, pour chaque flux f, l'algorithme de calcul des nombres d'allocations (Alg. 1) initialise le nombre de cellules à allouer, pour chaque message et à chaque saut, à la valeur maximale $n_{frag}(f) + n_{rtx}^{max}(f)$ (ligne 2). Si le *PDR* minimum attendu n'est pas satisfait à cette étape (Eq. (6.7)), il n'existe pas de solutions. Dans ce cas, le flux n'est pas alloué (cf. condition ligne 5) et l'algorithme continue avec le suivant.

Sinon, l'algorithme répète les trois étapes suivantes :

1. Nous trouvons le lien le plus chargé du chemin, parmi les liens qui ne sont pas encore traités (ligne 8) ;
2. Nous décrémentons le nombre d'allocations de ce lien et lui actualisons la charge (ligne 9) ;

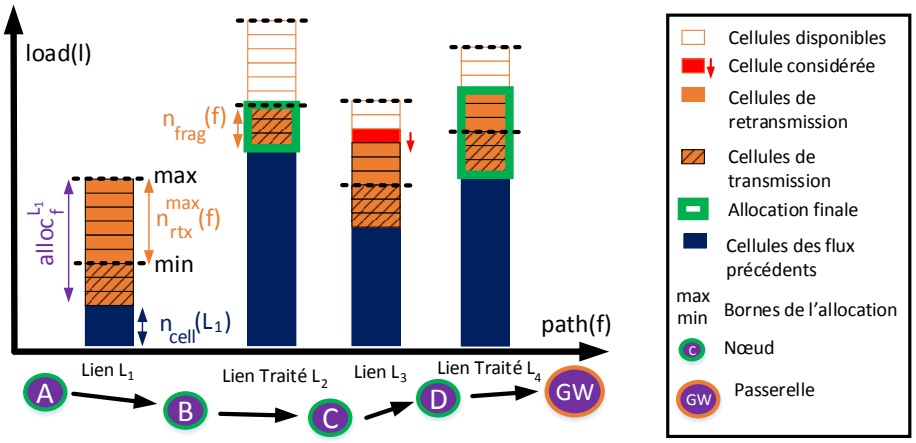


FIGURE 6.1 – Exemple d'une itération de l'algorithme de sur-dimensionnement par saut pour un flux d'un message

3. Nous vérifions que la contrainte de *PDR* de bout-en-bout (Eq. (6.7)) est encore satisfaite et que la transmission de tous les fragments est encore possible (Eq. (6.12)) (ligne 12).

Si cette condition n'est plus valide, nous remettons à jour les variables à la dernière valeur valide (ligne 13) et nous marquons le lien considéré comme traité (ligne 16).

L'algorithme réduit de manière itérative la charge du lien le plus chargé (étape 1). Ce lien peut être différent à chaque itération, jusqu'à ce que la contrainte de *PDR* ne soit plus valide : nous ne pouvons plus réduire la charge de ce lien.

Lorsque les nombres d'allocations ont été successivement calculés pour tous les liens du chemin, nous mettons à jour le nombre de cellules allouées pour chaque nœud avec les nouvelles valeurs (ligne 18) et l'algorithme continue avec le flux suivant.

La Fig. 6.1 illustre une itération de l'algorithme, pour un chemin constitué de quatre liens et pour un flux comportant un seul message (trois fragments). Les valeurs des composantes de $alloc_f$ varient entre le nombre de fragments d'un message (3 dans cet exemple) et le nombre maximal de transmissions ($6 + 3 = 9$ dans la figure). Les nombres d'allocations pour les liens L_2 et L_4 ont été fixés par l'algorithme à une itération précédente.

Pour le flux considéré, l'algorithme n'a pas prévu de cellule de retransmission pour le lien L_2 car c'est le plus chargé, à cause des allocations des précédents flux ($n_{cell}(L_2)$ est le plus important des 3 liens). Ainsi l'algorithme, en essayant de réduire la charge maximale, a d'abord atteint la valeur minimale pour ce lien en décrémentant ses opportunités de retransmission. L'approche favorise l'équilibrage de charge (nous reportons le surcoût de la fiabilité sur les autres liens).

Le lien L_1 n'a pas encore été traité. Son nombre d'allocations est maximal. Puisqu'il est le lien non traité le plus chargé, L_3 est décrémenté d'une cellule.

Nous calculons ainsi pour chaque flux l'ensemble des nombres d'allocations avec une valeur minimale de charge maximale. Nous avons démontré l'optimalité de cette approche en termes de distribution des charges pour un flux sur un chemin donné dans le rapport de recherche [**?**].

La variable $PDR_{msg}(f)$ (Eq. (6.9)) diminue avec chaque décrément du nombre de cellules (étape 2 de l'algorithme). Mais l'impact d'un décrément sur le *PDR* dépend du *PER* de chaque lien. Un algorithme visant à minimiser le nombre total de cellules choisirait de diminuer en priorité la charge des liens les plus important pour l'Eq. (6.7) (i.e. choisir les liens par ordre décroissant de *PER*).

6.3.3 Allocation des retransmissions : extensions de l'algorithme *TASA*

Dans cette section, nous expliquons le fonctionnement de *Traffic-Aware Scheduling Algorithm, (TASA)* [?] [?], puis nous montrons comment mettre en œuvre les deux stratégies de sur-dimensionnement (uniforme et saut-par-saut) dans *TASA*.

Fonctionnement de *Traffic-Aware Scheduling Algorithm (TASA)* L'algorithme considère que chaque nœud organise une structure de *file d'attente* des trames (nommées *paquets*) (premier entré, premier sorti, *FIFO*) nommée *queue* dans l'algorithme. Un certain nombre de *paquets* est présent dans la *file* de chaque nœud source au début de l'échéancier (intervalle 0). A chaque allocation, un *paquet* transite depuis la *file* du nœud émetteur jusqu'à la *file* du récepteur.

TASA alloue autant de cellules dans un échéancier que le trafic attendu pendant une période de temps, et répète l'échéancier résultant jusqu'à la prochaine allocation. L'approche de *TASA* alloue les cellules intervalle après intervalle, de manière consécutive, ce qui donne un échéancier optimal en termes de taille active.

TASA prend comme paramètres d'entrée :

1. un arbre de routage enraciné à chaque passerelle ;
2. un graphe de conflit.

TASA applique de manière itérative une technique de couplage issue de la théorie des graphes :

1. À chaque intervalle k, l'algorithme sélectionne récursivement les liens qui obtiendront une allocation de cellule. Le processus de sélection parcourt les arbres formés depuis chaque passerelle vers les feuilles et choisit les liens en fonction de la charge et en respectant la contrainte *half-duplex* sur les nœuds (Alg. 2). L'ensemble des liens sélectionnés peut être utilisé au même intervalle de temps (aucun nœud n'est à la fois émetteur et récepteur à un intervalle de temps k donné). L'ensemble de liens ainsi construit est appelé *Duplex Conflict-Free Link* ($DCFL(k)$).
2. Ensuite, *TASA* applique une fonction de coloriage à chaque intervalle k. L'algorithme attribue un canal de fréquences c à chaque sous-ensemble de liens de $DCFL(k)$ qui n'interfèrent pas entre eux (les sous-ensembles sont déterminés à partir du graphe de conflit).
3. Les *files* sont mises à jour : la cellule (k, c) est allouée au plus ancien *paquet* présent dans la *file*.

TASA construit des échéanciers sans prendre en compte la fragmentation de longs messages. En effet, chaque fragment d'un message serait transmis indépendamment des autres, sans les attendre à chaque relais comme dans les mécanismes de *route-over*.

Dans l'Algorithme 2, n est le nombre de boucles récursives. La charge du sous-arbre à l'étape 2 représente la somme des tailles de *file* des éléments de l'arbre enraciné au nœud considéré. Le pas de récursivité (étape 4) permet de respecter la contrainte *half-duplex*. La condition d'arrêt du processus récursif correspond au fait que les nœuds feuilles n'ont aucun descendant.

Mise en œuvre du sur-dimensionnement uniforme dans *TASA* *TASA* ne prend pas en compte les retransmissions. Nous proposons de mettre en œuvre l'approche uniforme en incluant les *duplicata* des fragments de bout-en-bout. Nous exécutons l'algorithme de *TASA* sans aucune modification.

Algorithme 2 : Algorithme récursif de sélection d'ensembles *Duplex Conflict-Free Link (DCFL)* dans *TASA*.

Données : N : ensemble des nœuds $i \in N$ considérés;
$GW \subseteq N$: ensemble des passerelles;
$\forall i \in N, arbre(i)$: ensemble des descendants de chaque nœud i, avec i;
$\forall i \in N, fils(i)$: ensemble des fils de chaque nœud i;
$\forall i \in N \mid i \notin GW, parent(i)$: parent de chaque nœud i (hors passerelles);
$\forall i \in N, Q(i)$: nombre de *paquets* dans la *file* de chaque nœud i;
Résultat : $DCFL(k)$: ensemble des liens *Duplex Conflict-Free Link* pour un intervalle de temps k donné

Initialisation :
Ensemble S de départ : l'ensemble des passerelles :
  $\forall i \in GW, S \leftarrow S \cup \{i\}$;
n : nombre de boucles récursives ;
  $n \leftarrow 0$;
  $DCFL(k) \leftarrow \emptyset$;
tant que $taille(S) > 0$ **faire**
  Étape 1) Recherche de l'ensemble D des plus proches descendants des nœuds de S ayant une *file* non vide :
    $\forall i \in D, \exists j \in S \mid i \in arbre(j)$;
    $\forall i \in D, Q(i) > 0$;
    $\forall i \in D, Q(parent(i)) = 0$;
  Étape 2) Sélection de chaque descendant d_{max} correspondant à la charge de sous-arbre maximale :
    P : ensemble des parents des éléments de D ;
    $\forall i \in P, \exists d_{max}(i) \in fils(i) \cap D \mid \sum_{k \in arbre(d_{max}(i))} Q(k) = \max_{j \in fils(i) \cap D} \sum_{k \in arbre(j)} Q(k)$;
  Étape 3) Ajout de chaque lien entre les descendants choisis et leurs parents dans $DCFL(k)$:
    $\forall i \in P, DCFL(k) \leftarrow DCFL(k) \cup \{[d_{max}(i) \rightarrow i]\}$;
  Étape 4) Poursuite récursive de la sélection à l'étape $n+1$:
    $S \leftarrow \emptyset$;
    Nouvel ensemble S de départ :
      - les autres fils des éléments de P, s'il y en a :
        $\forall i \in P, \forall j \in fils(i) \mid j \neq d_{max}(i), S \leftarrow S \cup \{j\}$;
      - les fils de chaque descendant choisi, s'il y en a
        $\forall i \in P, \forall j \in fils(d_{max}(i)), S \leftarrow S \cup \{j\}$;
    $n \leftarrow n + 1$;
fin

Nous changeons en revanche la définition de trafic. À l'intervalle 0, l'algorithme d'ordonnancement considère que chaque source a dans sa *file* :

1. les fragments de chaque message applicatif ;
2. les *duplicata* de bout-en-bout des fragments de chaque message (Eq. (6.5)).

De cette façon, *TASA* alloue des cellules pour un nombre constant de retransmissions tout le long du chemin. L'échéancier résultant fournit suffisamment de cellules à chaque saut pour la satisfaction du *PDR*.

Lorsque un nœud détecte la perte d'un *paquet* (il ne reçoit pas d'acquittement), il tente une retransmission sur la prochaine cellule ayant le même *track ID*. Le même comportement est répété jusqu'à atteindre le nombre maximum de retransmissions. Le fragment est ensuite supprimé et enlevé de la mémoire du nœud.

Mise en œuvre du sur-dimensionnement saut-par-saut dans *TASA* Afin d'intégrer les cellules de retransmission saut-par-saut et flux par flux dans *TASA*, nous modifions légèrement l'algorithme original. Nous distinguons les cellules de transmission et de retransmission pour chaque flux :

1. lorsque nous sommons le trafic d'un sous-arbre, nous prenons en compte le nombre nécessaire de cellules de retransmission que nous avons calculé pour chaque flux (Eq. (6.12), $alloc_f^l - n_{frag}(f)$) en même temps que la taille des *files* ;
2. le processus de sélection considère que les nœuds ayant encore des cellules de retransmission à allouer ont une *file* non vide ;
3. l'ordre d'allocation des cellules change. La cellule (k, c) est allouée message par message :
   (a) aux plus anciens fragments dans la *file* ;
   (b) aux retransmissions correspondantes ;
4. les *files* sont actualisées par message, après la dernière cellule de retransmission. Tous les fragments ont alors été transmis au récepteur.

Ces modifications permettent un mécanisme de *route-over*, car elles fournissent des cellules consécutives qui donnent à chaque nœud la possibilité de rassembler tous les fragments de chaque message dans sa *file* avant de les relayer.

6.4 *KAUSA* : allocation de ressources multi-flux à contraintes de *SLA*

Dans cette section, nous présentons *KPI-Aware Scheduling Algorithm* (*KAUSA*). L'algorithme inclut l'approche précédente pour la satisfaction des *PDR* par flux. *KAUSA* permet en plus de prendre en compte des contraintes de délai de bout-en-bout par message. En effet, les approches présentées dans les sections précédentes ne permettent pas de choisir les cellules de telle sorte que les transmissions des fragments d'un même message aient lieu les unes proches des autres.

6.4.1 Vision globale de *KAUSA*

Nous considérons un algorithme d'ordonnancement centralisé qui alloue des cellules *FTDMA* à un ensemble de flux. *KAUSA* prend comme paramètres d'entrée :

1. les paramètres *SLA* de chaque flux : le profil de trafic et les *KPI* attendus ;
2. les informations de topologie : le *PER* de chaque lien et le rang de chaque nœud ;
3. les interférences : la distance minimale (nombre de sauts) à partir de laquelle l'opérateur juge les interférences négligeables.

La Fig. 6.2 montre les étapes de l'allocation. *KAUSA* construit d'abord un chemin pour chaque flux (étape 1). Les chemins sont choisis afin d'équilibrer la charge sur le réseau tout en satisfaisant une fiabilité minimale exigée dans le *SLA*. Ensuite, *KAUSA* calcule le nombre de cellules nécessaires à chaque saut pour satisfaire le *PDR*, d'après le *PER* de chaque lien.

Les nombres d'allocations résultant doivent permettre de respecter le délai dans le cas où les cellules sont allouées consécutivement (le délai devient alors le nombre d'intervalles correspondant).

Si le chemin choisi ne permet pas de respecter la contrainte de *PDR* ou de délai, nous appliquons une technique de retour en arrière (*backtracking*) sur le choix des liens (Sec. 6.4.6). De nouveaux chemins sont considérés en éliminant récursivement les liens des choix possibles. Nous évitons à la fois les parties les moins fiables et les plus chargées du réseau. La boucle récursive s'arrête lorsque la source n'a plus aucun lien possible vers un prochain saut. Le retour en arrière au niveau des liens favorise le respect des *KPI* au détriment de l'équilibrage de charge.

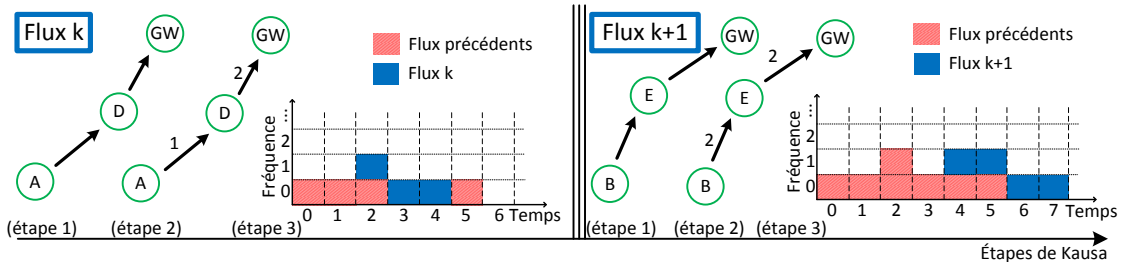


FIGURE 6.2 – Vision d'ensemble de *KAUSA*, par étape, pour deux flux k et $k+1$.

L'ensemble des cellules associées à un message à un saut donné est nommé *plage*. À chaque saut, une *plage* contient suffisamment de cellules pour les transmissions des fragments du message, ainsi que suffisamment de cellules de retransmissions. *KAUSA* alloue les cellules de chaque plage séquentiellement dans l'échéancier *FTDMA* (étape 3).

Dans la Fig. 6.2, *KAUSA* alloue le flux $k+1$ en tenant compte des cellules déjà allouées pour les messages du flux k et des flux précédents.

Nous évitons les dépassements d'espace mémoire en contrôlant la quantité de fragments gardés sur chaque nœud, en considérant le scénario le moins favorable de retransmissions. *KAUSA* anticipe ou retarde en conséquence les allocations pour un message donné.

Si aucune allocation n'est trouvée pour un message donné, nous appliquons une technique de retour en arrière sur les allocations des précédents flux (Sec. 6.4.7). Nous retirons successivement les allocations des flux précédents, cherchons de nouveaux chemins et essayons d'autres allocations. Le retour en arrière au niveau des flux s'arrête quand il atteint le premier flux alloué.

6.4.2 Ordonner les flux

Nous classons les flux par ordre décroissant de *charge attendue*, de sorte que *KAUSA* traite en premier les plus demandeurs en termes d'allocations. Nous calculons la métrique de *charge attendue* directement à partir des paramètres *SLA* (Tab. 6.1). Dans l'Eq. (6.15), celle-ci correspond au nombre de fragments attendus par *slotframe*, pour un flux f donné :

$$n_{msg}(f) \cdot n_{frag}(f) \cdot PDR^{sla}_{msg}(f) \tag{6.15}$$

Dans le cas où deux flux ont des valeurs similaires pour cette métrique (e.g. moins de 1 % de différence), nous considérons le délai attendu $DELAY^{sla}_{msg}(f)$ comme deuxième critère de classement. À cette étape, nous choisissons de donner la priorité aux contraintes de charge et de fiabilité sur celles de délai. Cette priorité est compensée par la prise en compte du délai dans l'allocation et par le mécanisme de *backtracking* en cas d'echec d'allocation.

Lorsque plusieurs flux ont les mêmes *KPI* (généralement pour une application multi-source, e.g. la télérelève), ils ont la même métrique de *charge attendue* et de délai. Nous utilisons le rang de la source pour les départager (et si la source est identique, nous utilisons le *track ID*). Nous donnons à la source la plus éloignée en termes de rang la plus haute priorité, car la route peut être plus longue, et donc la charge finale plus élevée pour le flux considéré.

6.4.3 Construction des chemins

Nous proposons un mécanisme de routage multi-chemin sous contraintes de *KPI* et d'équilibrage de charge. Notre objectif est de minimiser l'impact de chaque flux sur le

réseau en termes de charge d'allocation. Nous construisons le chemin pour un flux, en tenant compte des allocations antérieures. Nous devons trouver des prochains sauts valides pour chaque relais le long du chemin, en termes de charge de trafic et de fiabilité.

Nous mettons à jour la topologie de routage, à partir des passerelles, en utilisant un algorithme à vecteur de distance (e.g. *Bellman-Ford*). Nous choisissons pour chaque nœud le prochain saut formant la meilleure route vers une passerelle, en termes de charge maximale de nœud, de charge cumulée sur la route et de nombre attendu de transmissions, (ou *Expected Transmission Count (ETX)*). Nous considérons ces trois critères dans l'ordre lexicographique :

$$< \max_{i\in[1;gw]} n_{cell}(A_i), \sum_{i\in[1;gw]} n_{cell}(A_i), \sum_{i\in[1;gw]} ETX(i) > \tag{6.16}$$

Dans l'expression (6.16), les entiers $1, 2, \ldots, gw$ sont les indices des liens $l_i \in [l_1, l_{gw}]$ constituant le chemin depuis le nœud considéré (A_1) jusqu'à une passerelle. Nous calculons $ETX(i)$, pour chaque chaque lien l_i, par le rapport $1/(1 - per(l_i))$. A_i est l'émetteur du lien l_i. La somme des ETX nous permet d'estimer le nombre nécessaire d'allocations de cellules pour un fragment sur un chemin donné.

Le prochain saut est choisi parmi les relais voisins de rang inférieur afin d'éviter les boucles de routage. Nous vérifions que le chemin obtenu est suffisamment fiable si on considère des transmissions indépendantes de fragment (cf. Eq. 6.1). *KAUSA* ne considère pas les chemins qui exigent, pour un lien donné et pour un fragment, plus de retransmissions que le maximum $n_{rtx}^{max(frag)}(f)$. L'Eq. (6.17) exprime la satisfaction de la contrainte de fiabilité en utilisant le nombre maximum de retransmissions pour un fragment, saut-par-saut :

$$\prod_{l\in path(f)} \left(1 - per(l)^{n_{rtx}^{max(frag)}(f)}\right) \geq pdr_{frag}^{sla}(f) \tag{6.17}$$

Si l'Eq. (6.17) n'est pas vérifiée, le lien choisi n'est pas assez fiable, le chemin n'est alors pas valide et l'algorithme doit revenir en arrière en écartant les liens trop peu fiables.

6.4.4 Allocation des messages

L'allocation se fait message après message, pour chaque flux ordonné. Nous associons chaque message à une *plage* pour chaque saut du chemin (Fig 6.3). Chaque plage est un ensemble de cellules dédiées au message sur un lien donné.

Les plages sont allouées séquentiellement. Afin d'éviter d'ordonnancer la transmission avant la réception d'un fragment, les plages du même message sont contiguës, mais ne se chevauchent pas. Par ailleurs, un ensemble de plages est valide s'il respecte la contrainte de délai : l'écart entre le premier intervalle de la première plage et le dernier intervalle de la dernière plage doit être inférieur au délai (en nombre d'intervalles).

La Fig. 6.3 représente la séquence d'allocation pour un message donné, sur le chemin d'un nœud feuille A vers une passerelle GW. Les flèches numérotées indiquent l'ordre d'allocation. La séquence commence avec la *plage de départ* choisie sur le lien le plus chargé dans l'échéancier : e.g. à l'étape 1, la plage du Lien 3 est la plage de départ.

1. **Allocation des cellules de la plage de départ :** il faut d'abord positionner les cellules de la plage de départ. Nous essayons itérativement toutes les *cellules de départ* possibles en parcourant tout l'échéancier. Nous choisissons la plage de départ, qui minimise l'occupation de canaux sur l'ensemble de ses cellules (voir en (a), Fig. 6.4). Nous arrêtons le parcours s'il se trouve une plage de départ avec une occupation nulle.

2. **Allocation des cellules des plages précédentes :** nous allouons les cellules des plages correspondant aux précédents sauts, de la dernière cellule à la première cellule et en remontant le chemin vers la source. Le lien considéré est donc le Lien 2, étape 2 et 3, Fig. 6.3. Nous choisissons de minimiser l'attente des fragments à chaque nœud en allouant les cellules les plus proches de la *cellule de départ*. Nous répétons le même processus (étape 4 et 5) jusqu'à atteindre le premier saut AB.
3. **Allocation des cellules des plages suivantes :** les plages de cellules des sauts suivants sont allouées consécutivement, à la suite de la dernière cellule de la *plage de départ* (étape 6). Si les conditions d'allocation des cellules ne peuvent pas être vérifiées ou si la contrainte de délai ne peut pas être respectée, l'allocation n'est pas valide.
4. **Modification de la plage de départ :** si l'allocation échoue à un saut donné, ou si le délai n'est pas respecté, l'allocation recommence pour le même message avec la deuxième meilleure *plage de départ* en termes d'occupation fréquentielle. Nous reprenons la recherche si celle-ci n'était pas achevée.

Si nous ne trouvons aucune plage de départ correcte, l'allocation du message échoue et l'algorithme doit revenir en arrière au niveau des flux.

La Fig. 6.4 montre un exemple d'allocation pour un message sur un chemin à 3 sauts $ABCD$. Une fois fait le choix de la plage de départ (étape (a)) les cellules 3, 4 et 5 sont allouées consécutivement pour le lien le plus chargé (Lien 2, étape (b)). Du fait du *Channel Hopping*, les canaux de fréquences utilisés diffèrent. Puis, la plage précédente est allouée pour le lien 1, sur les cellules 2, 0, puis 9 (étape (c)). En effet, le caractère cyclique des échéanciers permet de considérer les cellules 9 et 0 consécutives (ici la taille de l'échéancier est très petite, 10 intervalles, pour simplifier l'exemple). La cellule 1 est *sautée*. Enfin, à l'étape (d), la plage du dernier saut (Lien 3) est allouée à la suite de la première, cellules 6, 7 et 8.

La Fig. 6.5 montre un exemple (la légende reste inchangée) de *saut* de cellule (a), et de *décalage* de plages (b). Dans le cas 1 (partie supérieure), la cellule 5 est *sautée* pour le lien GH. Dans le cas 2 (partie inférieure), un problème d'espace mémoire est détecté à la cellule 1. Les plages des liens EF et FG sont *décalées* vers la gauche pour tenter de résoudre le problème.

6.4.5 Conditions d'allocation des cellules

Au sein d'une plage, nous allouons consécutivement les cellules, soit sur les intervalles de temps précédents, vers la source, soit sur les intervalles suivants, vers la destination

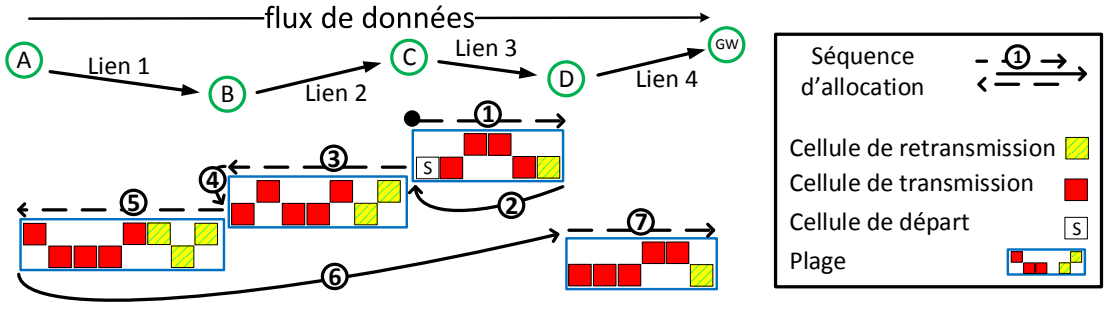


FIGURE 6.3 – Vision d'ensemble de la séquence d'allocation des plages dans *KAUSA*.

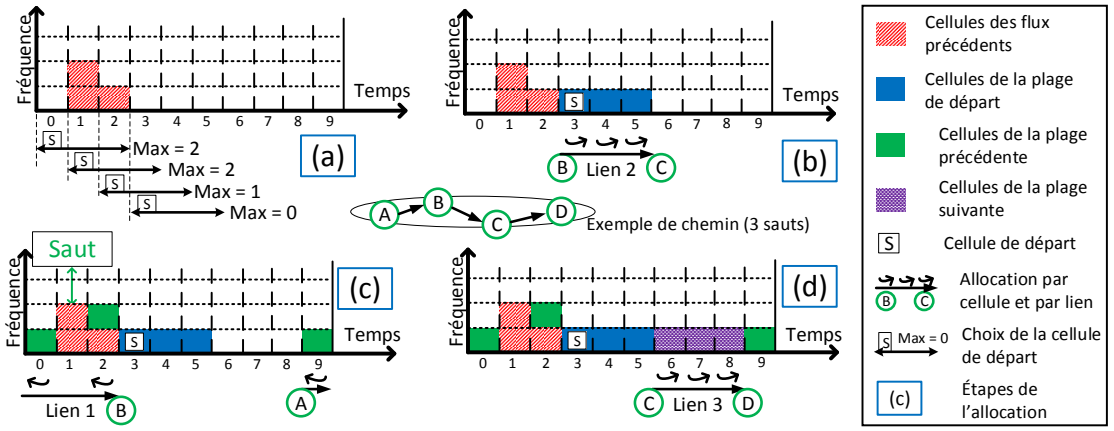


FIGURE 6.4 – Exemple d'allocation, par étapes, pour un message de 3 fragments sans retransmission.

(étape 5 ou 7 sur la Fig. 6.3).

Les cellules d'une plage doivent respecter les conditions suivantes :

condition de *half-duplex* : l'émetteur et le récepteur ne sont pas déjà en train d'utiliser leur interface radio ;

condition de *capacité* : il y a un canal disponible dans le voisinage ;

condition de *mémoire en réception* : le récepteur a suffisamment de mémoire disponible dans le cas où toutes les transmissions sont réussies dans les premières cellules de la plage. Ce cas est le pire pour le récepteur : sa mémoire est occupée plus tôt ;

condition de *mémoire en émission* : l'émetteur a suffisamment de mémoire disponible pour les fragments non encore transmis, dans le cas où les transmissions ne sont réussies qu'à la fin de la plage. Parmi les cas qui valident les conditions du *SLA*, celui-ci est le pire pour l'émetteur : sa mémoire est libérée plus tard.

Si une de ces conditions n'est pas respectée, nous *sautons* une cellule (Fig. 6.5) et nous essayons avec les suivantes. Les conditions de mémoire doivent être vérifiées pour toute la plage, y compris pour les cellules *sautées*, parce qu'aucun fragment n'y est transmis, donc aucun espace mémoire n'y est libéré.

Pendant l'allocation d'une plage donnée, si les conditions de mémoire ne sont plus valables, nous devons récursivement *décaler* les plages vers les intervalles de temps précédents, vers la source, jusqu'à ce que le problème soit résolu (Fig. 6.5). Pour chaque plage, nous déplaçons les cellules allouées et nous vérifions à chaque cellule les conditions. Si la longueur de l'ensemble de plages dépasse la taille de l'échéancier en *décalant* les plages, l'allocation n'est pas valable et nous devons modifier la *plage de départ*.

6.4.6 Technique de *backtracking* au niveau des liens

KAUSA construit un chemin alternatif pour un flux donné (étape 1 dans la Fig. 6.2) :

1. lorsque l'algorithme de routage ne trouve aucun chemin convenable pour un flux donné ;
2. lorsque le calcul du nombre de cellules échoue pour un flux donné.

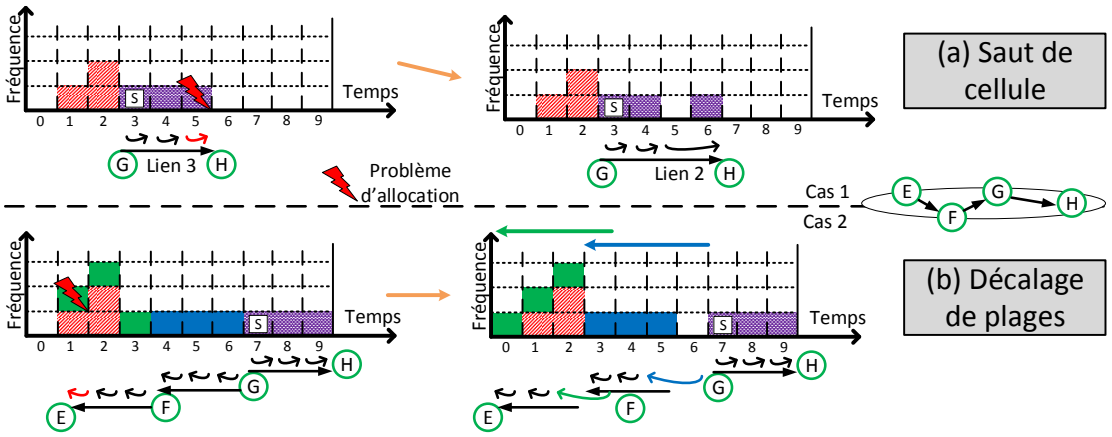


FIGURE 6.5 – Saut de cellule et décalage de plages.

Un des liens du chemin est alors *écarté* pour le flux. Ensuite, un nouveau chemin est construit selon la métrique de routage (Eq. 6.16) mais en évitant les liens *écartés*.

Si l'appel au mécanisme de *backtracking* est lié à une impossibilité de construire un chemin ou à un non-respect de la contrainte de délai, nous *écartons* le lien ayant le pire *PER*, afin de construire un chemin plus fiable.

Dans tous les autres cas, nous *écartons* pour cette étape le lien qui somme la charge d'allocation la plus élevée pour ses nœuds voisins, afin de construire un chemin moins chargé. Si aucun chemin valide n'est trouvé à cette étape, nous éliminons définitivement de la boucle de *backtracking* le lien avec le pire *PER* et nous replaçons parmi les possibilités les liens temporairement *écartés*.

Nous faisons des appels de *backtracking* récursifs jusqu'à ce que nous trouvions un chemin valide satisfaisant le délai (succès) ou que tous les voisins de la source soient *écartés* (échec). Lorsque le *backtracking* au niveau des liens échoue, notre algorithme met de côté le flux concerné et reprend avec le suivant. Ainsi, le *backtracking* au niveau des liens ne couvre pas tous les chemins possibles, mais élimine progressivement les liens ayant la plus mauvaise qualité.

6.4.7 Le *backtracking* au niveau des flux

Un *backtracking* au niveau des flux est initié lorsqu'une allocation échoue pour un message donné (l'étape 3 échoue sur la Fig. 6.2). Nous sauvegardons dans un premier temps l'allocation courante.

Pour le flux concerné, nous essayons d'abord le *backtracking* au niveau des liens. Si celui-ci échoue sur le flux, nous revenons sur les liens du flux précédent (e.g. dans la Fig. 6.2, si le *backtracking* échoue sur les liens du flux $k+1$, nous revenons à l'étape 1, pour le flux k). Lorsque le *backtracking* au niveau des flux atteint le premier flux alloué ($k = 0$) et échoue, il n'y a pas de solution avec les hypothèses que nous avons faites. Nous reprenons *KAUSA* avec l'allocation précédemment sauvegardée et continuons à partir du flux suivant.

6.5 Évaluation des performances

6.5.1 Scénario d'évaluation

Nous exécutons un simulateur de réseau, basé sur la méthode de *Monte-Carlo*, composé de plusieurs modules développés en utilisant des scripts *Python* [**?**]. Nous utilisons le modèle de propagation radio développé en Section 6.2.2 sur un ensemble donné de topologies. Les simulations soulignent l'impact des variations de certains paramètres, en maintenant tous les autres à une valeur par défaut. Le Tab. 6.2 résume les paramètres et les valeurs par défaut que nous utilisons dans l'évaluation des performances.

Nous définissons deux applications différentes (e.g. une application de télérelève de compteurs de gaz, app. 1, et une application de *Smart Parking*, app. 2). Nous choisissons une contrainte forte de *PDR* (app. 1) et une contrainte forte de délai (app. 2). Chaque nœud feuille génère un flux, pour la moitié d'entre eux associé à la première application, pour l'autre moitié à la seconde application.

Pour les deux applications, les plages de variations des contraintes de délai et de *PDR* sont réalisées en proportion des deux valeurs par défaut (Tab. 6.2). Les variations des contraintes sont exprimées en pourcentage d'incrément dans les figures. Par exemple, la contrainte de *PDR* varie pour l'app. 2 par incréments successifs entre 0.80 et 0.98.

Les nœuds sont placés dans un rectangle de 400×200 mètres. Les relais sont positionnés selon un maillage triangulaire (tous les 70 mètres environ). Les nœuds feuilles sont uniformément répartis sur le rectangle. Les deux passerelles sont placées aux positions (100,100) et (300,100).

Nous estimons le *PER* pour chaque lien selon le modèle de propagation *pathloss* [**?**], en prenant les valeurs présentées dans le Chapitre 2, Fig. 2.3. Afin de rendre possible la simulation à l'échelle choisie, deux nœuds sont considérés comme voisins si leurs deux *PER* sont inférieurs à un seuil donné : 95%.

Sur la base des valeurs de *PER*, nous construisons un arbre de routage enraciné à chaque passerelle, selon la métrique d'*ETX*. Nous construisons le graphe de conflits sur la base des voisinages à 2 sauts.

Nous comparons d'abord l'algorithme original de *TASA* avec notre mise en œuvre avec les extensions que nous avons proposées. Ensuite, nous comparons *KAUSA* avec l'extension de *TASA* avec des retransmissions saut-par-saut. Nous exécutons, sur 16 topologies générées aléatoirement, les algorithmes en faisant varier les paramètres suivants :

1. **intensité du trafic** : nous faisons varier le nombre de messages pour chaque flux ;
2. **taille maximale de l'échéancier** : au delà d'une certaine taille, les échéanciers sont tronqués, une portion seulement est appliquée ;
3. ***PDR* attendu** : nous rendons plus difficile la satisfaction de la contrainte de *PDR* ;
4. **délai attendu** : nous relâchons la contrainte de délai.

Nous considérons les critères de performance suivants :

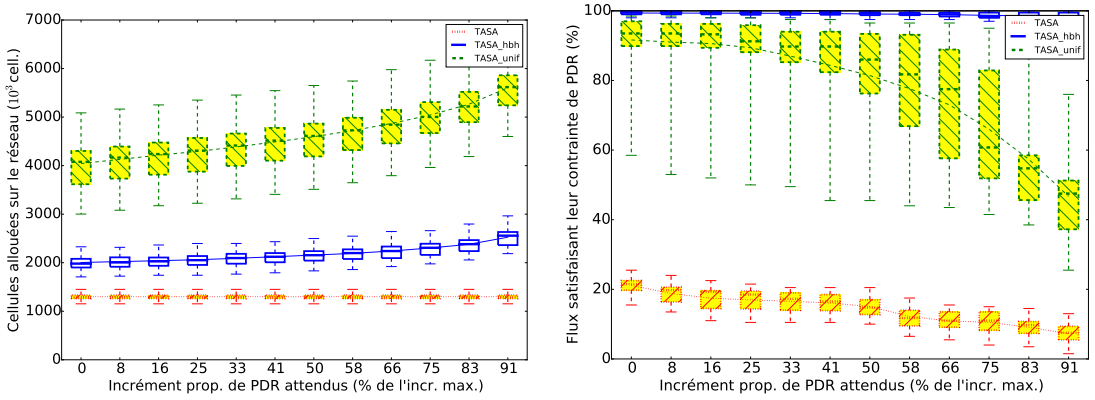
1. **satisfaction de la contrainte de *PDR*** pour chaque flux : nous évaluons la fiabilité des échéanciers créés avec le pourcentage des flux garantissant leur contrainte ;
2. **respect des conditions du *SLA*** : nous évaluons la satisfaction à la fois du délai et de la contrainte *PDR* pour chaque flux ;
3. nombre **maximum d'allocations par nœud** : ce critère détaille la capacité du réseau à supporter plus de trafic (présence de goulets d'étranglement) ;
4. **charge d'allocation globale** du réseau : nous évaluons l'efficacité des solutions en termes d'usage des ressources.

TABLE 6.2 – Paramètres de l'évaluation

Paramètre	valeur par défaut	plage de variation
Taille max. d'échéancier (intervalles)	1000	100 à 1200
Nombre de canaux de fréquences	16	
$n_{rtx}^{max}(f)$, nombre maximum de retransmissions	16	
Nombre de feuilles, distribution uniforme	200	
Nombre de relais, côté de la maille	24	$70m$
Nombre de passerelles	2	
$n_{msg}(f)$, nombre de messages par *slotframe*	1	1 à 12
$n_{frag}(f)$, taille de message (fragments), app. 1	3	
$n_{frag}(f)$, taille de message (fragments), app. 2	2	
PDR attendu, app. 1	0.97	0.97 à 0.997
PDR attendu, app. 2	0.80	0.80 à 0.98
Dimensions du rectangle	$400m \times 200m$	
$DELAY_{msg}^{sla}(f)$, délai attendu, app. 1	90	18 à 266
$DELAY_{msg}^{sla}(f)$, délai attendu, app. 2	60	12 à 177
Occupation maximale de mémoire (*KAUSA*)	20 fragments	

6.5.2 Résultats

Les figures suivantes mettent en évidence les différences entre *TASA*, *TASA* avec chacune des deux extensions proposées (retransmissions), et *KAUSA*. Nous comparons d'abord les trois variantes de *TASA* entre elles, puis la meilleure à *KAUSA*. Les courbes montrent les résultats obtenus en simulation en représentant le minimum, le premier quartile, la médiane, le troisième quartile et les valeurs maximales obtenues pour les 16 topologies générées aléatoirement. La ligne continue reliant les valeurs discrètes représente les valeurs moyennes.



(a) Charge d'allocation du réseau

(b) Satisfaction du *PDR*

FIGURE 6.6 – Influence de la contrainte de *PDR*.

Critique du mécanisme uniforme La Fig. 6.6a montre que le mécanisme uniforme de sur-dimensionnement est coûteux en termes d'usage des ressources du réseau : l'impact du mécanisme sur le nombre de cellules allouées sur le réseau est significatif.

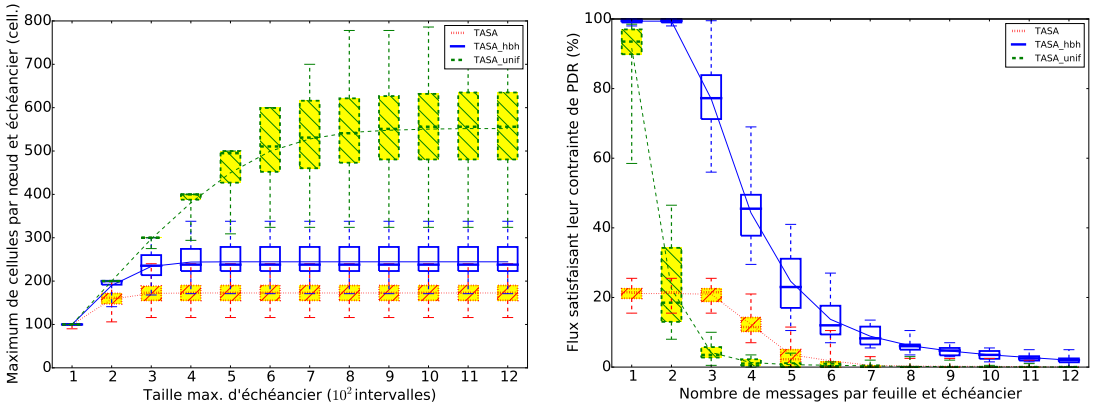
L'étude des performances du mécanisme uniforme est plus détaillée dans le rapport de

recherche [?]. On y voit que le mécanisme d'origine (*TASA*) ne permet pas de satisfaire la contrainte de fiabilité sur les liens radio sujets à pertes. Le mécanisme de prise en compte de retransmissions permet la satisfaction des contraintes de *PDR*, quand le réseau n'est pas en limite de capacité (Fig. 6.6b).

Le mécanisme uniforme est moins efficace que le mécanisme saut-par-saut parce que ses hypothèses sont plus exigeantes : en considérant les transmissions des fragments indépendantes de bout-en-bout, il surestime le nombre de cellules nécessaires et entraîne la saturation (Fig. 6.6b).

Le nombre maximum de cellules allouées par nœud, en transmission ou en réception, est constant pour *TASA* : *TASA* ne considère pas la contrainte de *PDR*. Le rapport montre aussi que le mécanisme uniforme cesse de fonctionner correctement pour des trafics intenses ou des tailles maximales d'échéanciers courtes car il sature les ressources (relais, charge globale) pour certains flux au détriment de tous les autres. Le *PDR* est alors dégradé.

L'algorithme saut-par-saut est plus performant : le *PDR* se dégrade moins rapidement. Avec le mécanisme saut-par-saut, l'échéancier reste efficace (la charge maximale par nœud est toujours inférieure à deux fois celle de l'algorithme original, Fig. 6.7a).



(a) Maximum de charge par nœud

(b) Satisfaction du *PDR*

FIGURE 6.7 – Influence de la taille de l'échéancier et de l'intensité de trafic.

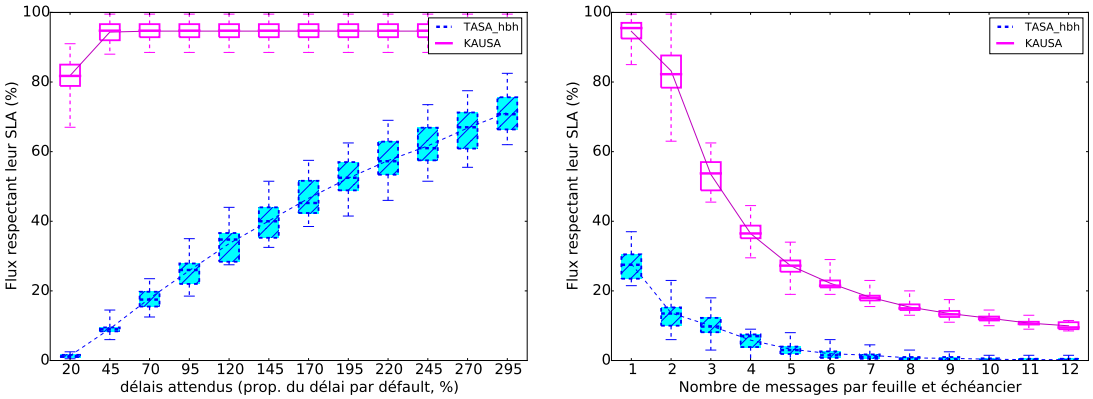
Gains du mécanisme saut-par-saut Par rapport à l'algorithme original, *TASA* [?], notre version améliorée avec retransmissions saut-par-saut, nommée $TASA_{hbh}$ construit des échéanciers fiables tout en distribuant la charge sur le réseau.

Dans notre simulation, environ 40 flux (20%) satisfont leur contrainte de *PDR* sans installer de retransmissions (Fig. 6.7b). Dans la Fig. 6.6b, cette proportion diminue de 20% à 10% lorsque la contrainte augmente. Au contraire, l'algorithme avec retransmissions permet de satisfaire la contrainte de *PDR* pour les messages de plus de 95% des flux.

Nous montrons en [?] que le *PDR* est dégradé lorsque la taille maximale de l'échéancier est trop courte pour contenir toutes les cellules nécessaires. L'échéancier doit en effet être plus long pour contenir toutes les opportunités de retransmissions. Dans la Fig. 6.7a, le mécanisme uniforme sature l'échéancier pour des tailles de moins de 700 intervalles. Avec le mécanisme saut-par-saut, la charge maximale sur un nœud donné est généralement inférieure à 300 cellules : il reste des ressources disponibles, le mécanisme ne sature pas le réseau.

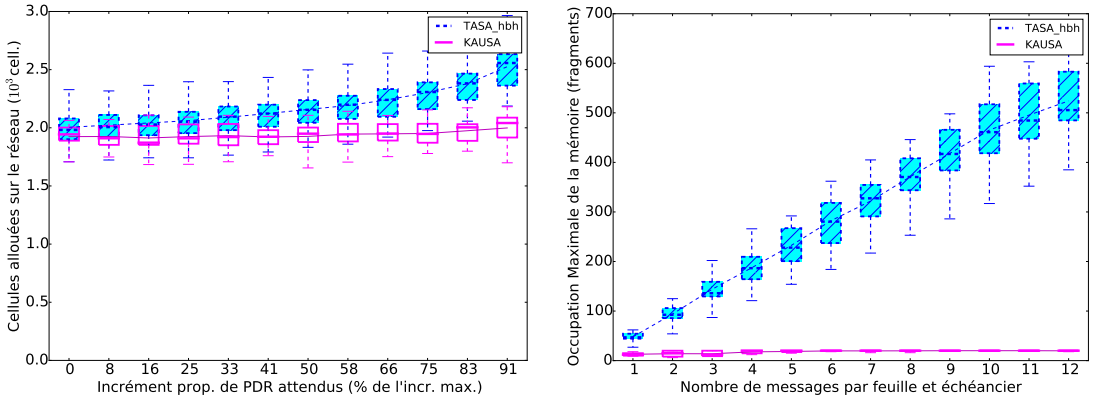
Lorsque l'intensité du trafic augmente, le mécanisme saut-par-saut maintient le *PDR* pour plus de flux que les versions originale et uniforme (Fig. 6.7b). Le mécanisme uni-

forme maintient même moins de flux que *TASA*, au delà de 3 messages par *slotframe*, car les retransmissions pour les premiers flux traités préemptent les ressources et saturent l'échéancier, au détriment des flux suivants.



(a) Contrainte de délai. (b) Augmentation du trafic.

FIGURE 6.8 – Performances de *KAUSA* en termes de validation de *SLA*.



(a) Charge d'allocation. (b) Occupation max. de la mémoire

FIGURE 6.9 – Charge d'allocation et occupation de mémoire de *KAUSA*.

Évaluation de *KAUSA* La Fig. 6.8 montre que *KAUSA* est plus performant que *TASA* avec retransmissions, en termes de satisfaction de *SLA*. Avec des exigences fortes (e.g. quand la moitié des flux attendent un délai de 30 intervalles, soit 50% du délai par défaut), environ 90% des flux valident leurs *KPI* (*PDR* et délai) dans le cas de *KAUSA*, (Fig. 6.8a).

La contrainte de délai est le principal facteur limitant de l'extension saut-par-saut de *TASA* : sa performance augmente presque linéairement avec l'assouplissement de la contrainte (Fig. 6.8a). *KAUSA* utilise généralement moins d'allocations, car l'algorithme trouve des chemins qui facilitent un meilleur équilibre entre leur longueur et la qualité des liens.

L'étude présentée en [**?**] et la Fig. 6.9a montrent que même lorsque l'intensité du trafic atteint la limite de capacité, (les deux puits sont saturés lorsque chaque nœud feuille génère plus de 3 messages par *slotframe*), *KAUSA* permet à certains flux d'être traités (50% avec 3 messages).

Lorsque les contraintes de *PDR* augmentent, les performances de *KAUSA* restent stables : *KAUSA* trouve des solutions spécifiques en fonction de la topologie (choix des liens et des cellules), (Fig. 6.9a). Le processus de *backtracking* prive certains flux de leur allocation dans *KAUSA*, ce qui réduit l'augmentation de la charge d'allocation, alors que *TASA* sert indistinctement tout le trafic, rendant impossible la satisfaction des contraintes.

Enfin, la Fig. 6.9b montre qu'avec un trafic faible, l'occupation de la mémoire des nœuds dans *KAUSA* reste environ à 10 fragments (la moitié du maximum), par rapport à environ 40 fragments pour *TASA* avec retransmissions. Avec un trafic élevé *KAUSA* préserve la mémoire des nœuds, tandis que ce paramètre n'est pas pris en compte dans *TASA*.

6.6 Synthèse du chapitre

Dans un réseau où plusieurs applications ont différentes contraintes de *PDR* et délai, l'opérateur doit garantir une *QoS* différenciée en termes de fiabilité et de délai de livraison. L'allocation des ressources pour les réseaux de l'*IoT* est ainsi difficile à réaliser.

Dans ce chapitre, nous proposons une solution qui alloue de manière centralisée des ressources *FTDMA* sous contraintes de *QoS* différenciées par application. Notre algorithme, *KPI-Aware Scheduling Algorithm* (*KAUSA*), équilibre la charge de trafic, prolongeant la durée de vie du réseau, tout en satisfaisant les contraintes de livraison et de délai pour chaque flux. Un algorithme de backtracking contrôlé permet d'atteindre une solution satisfaisante dans un délai raisonnable.

Nous proposons deux mécanismes de sur-dimensionnement qui s'adaptent aux *PDR* attendus, en prenant comme paramètre la qualité de chaque lien sur le chemin choisi. Nous étendons l'ordonnanceur *TASA* en allouant des ressources pour les retransmissions. Avec *KAUSA*, nous proposons une allocation par plages consécutives de cellules *FTDMA* qui permet de respecter les contraintes de délai.

Nous modélisons le réseau que nous ciblons, où les nœuds ont des capacités mémoire limitées, et où les liens ont des qualités différentes, afin de choisir les meilleures allocations possibles sous hypothèses réalistes. Dans ce contexte, *KAUSA* fonctionne mieux que *TASA* en termes de satisfaction de *SLA*. Nous assurons la qualité de service par flux sans que la mémoire des nœuds ne soit saturée.

Nous démontrons l'intérêt de tous ces mécanismes par le biais de simulations. Nous mettons ainsi en œuvre *KAUSA* et nous le comparons avec l'algorithme *TASA* dans sa version originale et étendue avec des opportunités de retransmissions. Nos résultats montrent que nous améliorons l'ordonnancement en permettant de garantir des échéanciers fiables et respectant les délais sur des liens sujets à des pertes de trames, tout en limitant le coût en termes d'allocations.

Perspectives Dans les travaux futurs, nous souhaitons étudier les améliorations possibles de *KAUSA* en tenant compte des différentes stratégies de gestion de files d'attente sur chaque relais. Nous évaluerons l'impact de la prise en compte des contraintes de délai en priorité. Nous envisageons une adaptation de *KAUSA* pour des trafics dynamiques et non plus périodiques. Enfin, la prise en compte distribuée des contraintes de *SLA* permettrait une allocation locale moins coûteuse en termes de mise en place de l'échéancier.

Chapitre 7

Conclusion et travaux futurs

7.1 Rappel du contexte de diversification de l'*IoT*

Nous avons positionné notre recherche dans un contexte en pleine évolution, celui de l'*IoT*. Jusqu'à présent, la collecte d'information par radio différenciait les applications et leurs types de terminaux, avec différentes technologies de collecte et méthodes d'analyse des données. Nous observons une convergence des réseaux de l'*IoT*, favorisée par la virtualisation des réseaux et basée sur l'interopération de différentes technologies. Les efforts de standardisation de la *5G* visent à intégrer différentes technologies de réseaux maillés à celles des solutions cellulaires et mobiles [**?**]. De plus, l'*IoT* dépasse le champ des réseaux de capteurs sans fil en permettant une exploitation à distance du réseau. L'analyse des données est réalisée sur serveurs distants, et de même le plan de contrôle du réseau peut être extériorisé : la mise en œuvre des *SDN* sur les réseaux de l'*IoT* en facilite la convergence [**?**]. Celle-ci est encore davantage favorisée par l'usage de la pile protocolaire commune *IP* [**?**].

La densification des réseaux liée à l'essor des capteurs (e.g. dans les terminaux mobiles) et des nouveaux objets (e.g. les maisons et bâtiments connectés) rend la convergence difficile du fait de la radio. En effet, la concurrence à l'accès au médium radio croît avec la densité de nœuds, en particulier lorsque les communications sont réalisées sur des bandes de fréquences sans licences. L'usage non organisé des ressources de communication augmente l'impact des interférences, le risque de collisions, et réduit donc leur efficacité et la fiabilité.

Deux types de déploiements sont défendus pour répondre à ce problème, les infrastructures mono-sauts et multi-sauts. Le premier type de déploiements mono-sauts est préféré aujourd'hui parce que le coût est réduit (pas de relais, dispositifs moins complexes, protocoles plus simples), mais impose des contraintes (e.g. limiter la taille des messages) et présente des désavantages (e.g. bas débit, coût important de diffusion des trames de contrôle et des retransmissions, maintien difficile de fiabilité face à des interférences localisées).

Le deuxième type, multi-saut, correspond aux efforts de recherche sur les réseaux de capteurs sans fil. La mise en œuvre est plus chère car davantage de relais sont nécessaires pour obtenir la même couverture, et davantage d'intelligence est requise sur chaque nœud (e.g. application du routage). Le temps de configuration du réseau est plus grand. Mais le débit est plus important sur des liens de plus courte portée. Les choix de routage et le contrôle des communications doivent permettre le maintien de garanties de délai, de fiabilité et la gestion de la capacité du réseau.

Afin de répondre à l'augmentation de la densité de trafic et aux exigences applicatives, nous mettons en avant une approche opérée sur une infrastructure partagée à large couverture, permettant de mutualiser les relais et de rentabiliser les coûts de déploiement.

L'opérateur peut alors gérer les besoins des applications, en tirant profit de la capacité globale sur les relais pour traiter efficacement l'ensemble des flux.

7.2 Rappel des contributions : l'architecture opérateur

L'opérateur, pour assurer la gestion des communications radio multi-sauts, doit :

1. offrir à des clients un service de communication et en préciser les caractéristiques ;
2. configurer le réseau en fonction des exigences des clients ;
3. surveiller la bonne réalisation du service aux clients ;
4. prévoir l'arrivée de nouveaux clients et la consommation de ressources (gestion de la capacité).

En effet, afin de mutualiser son infrastructure, l'opérateur doit maintenir différents niveaux de *QoS* pour les flux de plusieurs clients. Les prévisions de *QoS* pour l'admission de nouveaux flux sont complexes pour les réseaux de capteurs multi-services. Étant donnée la variabilité des paramètres, elles représentent toujours un problème ouvert [**?**, **?**].

Nous avons défini une solution permettant de contractualiser les performances du réseau opéré en termes de *QoS*. L'opérateur peut ainsi répondre aux quatre obligations listées ci-dessus.

7.2.1 Spécification des *Service Level Agreements (SLA)* pour l'*IoT*

L'expression des besoins des clients de l'*IoT* est le premier problème que nous résolvons dans cette thèse. Nous proposons d'utiliser des contrats de Qualité de Service (*QoS*), nommés *SLA* pour spécifier les besoins de clients et les engagements de l'opérateur dans le cas de l'*IoT*. Dans le Chapitre 3, nous proposons une structure de *SLA* adaptée, comprenant les métriques et les outils correspondant aux réseaux radio multi-sauts multi-services. Pour des flux attendus et contractualisés, l'opérateur peut ainsi prévoir l'usage des ressources et estimer la capacité de son infrastructure. Nous donnons la possibilité de définir les engagements concernant la remontée des informations d'observation du réseau.

7.2.2 Architecture de gestion des réseaux de l'*IoT* avec *SLA*

Ensuite, nous devons mettre en application les clauses des *SLA* sur les nœuds déployés. Il s'agit de distribuer les informations de configuration aux nœuds afin de réserver les ressources et maintenir les conditions contractuelles. L'architecture proposée au Chapitre 4 permet d'orchestrer les opérations de gestion nécessaires. Elle articule un ensemble de fonctionnalités d'observation, d'analyse, de configuration et s'interface avec les *SLA*. En particulier, elle traduit les exigences applicatives en expressions sur les paramètres de réseau. Nous montrons que ces paramètres dépendent parfois de la technologie utilisée.

7.2.3 Observation des *KPI* sur le réseau de l'opérateur

Nous montrons la nécessité de la remontée d'informations d'observation (ou *monitoring*), pour pallier les défaillances du réseau, améliorer les configurations et vérifier le respect des *SLA*. Dans le Chapitre 5, nous proposons différentes stratégies de remontée utilisant le *piggybacking*, c'est-à-dire l'apposition d'éléments d'information de contrôle sur les trames de données transitant sur le réseau. En particulier, nous évaluons le coût de l'isolation de trafic, en le comparant à celui d'un mécanisme de remontée en trames dédiées. Nous discutons l'impact sur les ressources de telles stratégies. Nous montrons donc que le *piggybacking* permet la vérification des *SLA* de manière efficace.

Nous montrons comment nos stratégies s'appliquent directement à la vérification des contraintes de bout-en-bout qui constituent les *SLA*. L'analyse des données remontées permet à l'opérateur de démontrer à ses clients que les exigences de *QoS* sont satisfaites.

7.2.4 Allocation de ressources pour la réalisation des *KPI*

La dernière contribution présentée dans le document concerne la configuration du réseau multi-saut de l'opérateur. Nous montrons au Chapitre 6 comment l'allocation de ressources permet de différencier flux par flux la *QoS* en respectant leurs contraintes. Nous nous basons sur une technologie *FTDMA*, qui permet d'allouer des blocs temps-fréquences de manière dédiée, pour la transmission d'un fragment et de son acquittement. Le mécanisme de *Channel Hopping* nous permet de réduire l'impact des interférences. L'ordonnancement *FTDMA* évite les allocations concurrentes dans un voisinage de taille donnée, afin de réduire encore le nombre de collisions. Nous proposons de rendre plus fiable l'utilisation des liens radio en allouant saut-par-saut des ressources supplémentaires dédiées aux retransmissions de trames en cas de pertes.

Notre proposition équilibre globalement la charge des nœuds par la métrique de routage choisie. Nous détaillons *KAUSA*, notre proposition d'ordonnanceur qui alloue de manière centralisée des plages de blocs temps-fréquences. Nous dépassons l'état de l'art représenté par *TASA* [**?**] car *KAUSA* permet de différencier les flux applicatifs selon le délai attendu de bout-en-bout et le taux de livraison attendu des messages. L'algorithme équilibre la charge flux par flux entre tous les nœuds afin d'augmenter la longévité des nœuds relais. Il limite également le temps d'attente en mémoire des trames avant leur relais au prochain saut. Les échéanciers temps-fréquences ainsi construits prolongent donc la durée de vie du réseau.

Évaluation des performances Nous avons étudié la mise en œuvre de l'observation, et de *KAUSA* dans le cadre proposé par le groupe de travail *IETF 6TiSCH*. Nous avons développé nos évaluations informatiques sur la base de modèles existants de propagation, de comportement des nœuds et d'arrivée de trafic. Nous proposons des scénarios réalistes pour le positionnement des nœuds et les contraintes applicatives. Les résultats des simulations montrent la pertinence de nos approches en termes d'utilisation des ressources :

1. nous évaluons le surcoût du *piggybacking* par le nombre d'octets transmis ;
2. nous évaluons le coût de l'allocation de ressources par le nombre de cellules allouées.

Les ressources utilisées influencent la consommation d'énergie des nœuds, et la durée de vie du réseau global. En effet, la consommation d'énergie des nœuds est liée à la quantité d'informations transmises et reçues par chaque nœud. Sous l'hypothèse de la connaissance de la topologie du réseau et de la quantité de trafic attendu, nous montrons dans quelle mesure l'opérateur peut allouer et contrôler les ressources, et donc en prévoir l'usage, l'énergie nécessaire et ainsi gérer la capacité de son réseau.

7.3 Perspectives : vers la convergence des réseaux de l'*IoT* opérés

Dans cette section, nous détaillons dans un premier temps les travaux immédiats qui nous semblent pertinents pour compléter nos contributions. Puis nous évoquons les perspectives à long terme que nous envisageons après nos travaux.

7.3.1 Perspectives à court terme

Il faut tout d'abord mettre en œuvre les *SLA* que nous proposons au sein des relations entre un opérateur et ses clients. Nous proposons d'intégrer ceux-ci comme annexe aux réponses aux appels d'offres, en spécifiant ainsi les exigences traitées. Pour ce faire,

l'opérateur doit adopter une stratégie d'admission de nouveaux clients permettant de rentabiliser son déploiement. L'analyse de capacité du réseau doit donc être approfondie afin de gérer l'architecture mutualisée de manière pérenne. Ainsi, le fonctionnement du ***SLA Admitter*** doit être précisé.

Il faut vérifier que les mécanismes que nous proposons permettent le passage à l'échelle d'un réseau opéré multi-service. En particulier, la robustesse du *piggybacking* aux pertes de trames doit être étudiée. Nous prévoyons d'approfondir les simulations sur ce point. Nous envisageons une validation expérimentale pour vérifier que des scénarios réalistes en intérieur ou en extérieur permettent l'observation par ce mécanisme. L'expérimentation sur *FIT/IoT-Lab* est rendue difficile car le positionnement statique et rapproché des nœuds empêche la reproduction de conditions réalistes de propagation. De plus, jusqu'en septembre 2016, l'observation *via* la plate-forme des comportements des nœuds (remontée passive hors bande d'informations sur le port série) ne peut pas facilement être généralisée à l'échelle d'une centaine de nœuds. Nous poursuivons nos efforts dans ce sens. L'utilisation de mallettes déplaçables et d'outils d'observation génériques de réseaux de l'*IoT* dans *SensOrLab* [**?**] devrait également permettre une meilleure validation des mécanismes. La mise en œuvre des mécanismes dans les standards, en code source ouvert, sur ces plates-formes est à poursuivre. La question de la standardisation de nos travaux se pose.

Nous devons confronter nos modèles d'interférences, de trafic et d'exigences applicatives à des cas réels, par validation expérimentale. En particulier, l'allocation de ressources doit être testée sur des équipements réels afin de vérifier le respect des exigences et l'efficacité du partage des ressources. La configuration des nœuds (e.g. mise en place des échéanciers temps-fréquences), responsabilité du ***SLA Enforcer*** dans l'architecture proposée, doit être approfondie. En effet, une réponse dynamique aux évolutions des interférences et du trafic est nécessaire. Par exemple, il faut étudier l'adaptation locale de l'allocation de ressources, qui permet des communications fiables et ajustées au niveau de chaque nœud, sans induire de surcoût de contrôle important. Nous pouvons réutiliser les travaux de *6TiSCH* allant dans ce sens dans le cas d'un ordonnancement distribué [**?**].

Enfin, l'évaluation de la consommation d'énergie des nœuds doit être améliorée. Nous devons préciser le lien analytique entre le nombre de cellules temps-fréquences allouées et la consommation des nœuds. En particulier, il faut étudier la consommation dans le voisinage d'une émission ou d'une réception. Les modèles théoriques de consommation peuvent être enrichis par des mesures réelles (e.g. meilleure intégration de la mesure de consommation dans l'observation sur les plates-formes de validation). Il faut préciser le lien entre la consommation d'énergie et la durée de vie du réseau, en prenant en compte les modèles de prévision de fonctionnement des piles et la présence éventuelle de sources d'énergie.

7.3.2 Perspectives à long terme

Nous avons choisi d'étudier la mise en place de réseaux opérés pour l'*IoT* pour une infrastructure multi-saut. Nous nous sommes en effet intéressés au départ à l'évolution des standards, notamment la prise en compte dans le routage de multiples contraintes avec *RPL* : plusieurs instances de routage permettent de prendre en compte plusieurs services, mais sans garanties de *QoS*. En particulier, dans le cas de contraintes antagonistes, les choix distribués de routes ne répondent pas au problème global. Puis, nous nous sommes penchés sur l'allocation déterministe de ressources en suivant les travaux de *6TiSCH*. L'utilisation d'échéanciers temps-fréquences permet en effet de maintenir, en termes de probabilité, des niveaux de fiabilité et de délai le long de chemins multi-sauts.

Entre temps, les acteurs des réseaux de l'*IoT* ont favorisé les déploiements mono-sauts

de longue portée, plus rapides du fait du plus faible coût en termes d'équipements, mais plus contraints (e.g. taille des messages réduite) et moins fiables : garantir la *QoS* sur des liens de longue portée coûte cher en termes de ressources, et les opérateurs préfèrent obtenir une plus large couverture rapidement, plutôt qu'un meilleur taux de livraison et une plus grande capacité. Ainsi, une approche de réseaux opérés mono-sauts rend nécessaire l'étude de mécanismes de maintien de fiabilité sur des technologies de plus longue portée ou de plus grande capacité. Par exemple, un ordonnancement du type de *6TiSCH* sur *IEEE Std 802.15.4g*, couche physique spécifiée en particulier pour la télérelève, permettrait de mettre en œuvre la *QoS* avec un meilleur débit et une couverture plus large et plus dense. Nous devons proposer l'intégration des mécanismes étudiés dans le cadre des efforts de standardisation, afin de faire émerger des solutions déterministes, en lien avec *DetNet* [**?**]. En particulier, il faut évaluer le coût en ressources des mécanismes de *QoS*, selon les niveaux d'interférences et les débits imposés par la densité des nœuds et contraints par la réglementation.

L'intégration de l'*IoT* dans les standards de la cinquième génération de réseaux mobiles (*5G*) ouvre de nouvelles perspectives pour les opérateurs. En effet, l'augmentation des débits possibles, la multiplication des terminaux compatibles et les mécanismes permettant l'interopérabilité enrichissent le champ d'application de l'approche multi-service. De plus, l'usage de bandes de fréquences attribuées à chaque opérateur réduit le risque d'interférences externes. La mise en place de *SLA* dans ce contexte mobile, et la prise en compte de débits hétérogènes dans la spécification des exigences sont à étudier. En particulier, les *SLA* doivent être traduits et adaptés pour les différentes contraintes applicatives (e.g. identification des sources de trafic, format des messages de configuration, aggrégation de données). Nous devons vérifier que les entités fonctionnelles de l'architecture de *SLA* peuvent s'intégrer dans le contexte de la *5G*. En particulier, l'analyse des informations collectées en dehors du réseau (*cloud computing*) comporte des défis (passage à l'echelle, extraction de données pertinentes), que ce soit pour les données applicatives ou pour les informations d'observation du réseau. De plus, la mobilité rend difficile le maintien des garanties de *QoS*. Nous devons spécifier les clauses des *SLA* afin de rendre possible l'allocation des ressources (e.g. taux de livraison pour un terminal mobile, temps de rétablissement du service après déplacement).

Enfin, à l'échelle d'une ville, l'infrastructure doit être dimensionnée pour le transit de l'ensemble du trafic de contrôle comme de données. Nous pouvons tirer parti des deux approches mono-saut et multi-saut pour former des réseaux hybrides, et améliorer la collecte d'information en utilisant la séparation des plans dans *SDN* :

1. l'échange en longue portée des informations du plan de contrôle permettrait l'observation rapide du réseau (e.g. la gestion des pannes) et la configuration réactive des nœuds de l'opérateur ;
2. l'échange en multi-saut des données utiles permettrait le respect de garanties de *QoS* tout en maintenant la capacité et la couverture du réseau.

Cette séparation permettrait de dissocier les exigences des applications du fonctionnement courant du réseau. La réalisation technique d'une telle solution est à étudier en termes matériels (e.g. relais multi-modes) et logiciels, afin de permettre la gestion parallèle des plans de contrôle et données sur chaque nœud. Nous devons étudier la répartition des fonctions de supervision du réseau, ou bien localisées sur chaque nœud ou bien centralisées en dehors du réseau sans fil. En effet, un compromis entre la capacité des nœuds et le surcoût de communication pour le contrôle doit être trouvé.

Publications

Conférences internationales

[1] Guillaume Gaillard, Dominique Barthel, Fabrice Theoleyre, and Fabrice Valois. Service Level Agreement Architecture for Wireless Sensor Networks : a WSN Operator's Point of View. In *Network Operations and Management Symposium (NOMS), Krakow, Poland*. IEEE/IFIP, May 2014.

[2] Guillaume Gaillard, Dominique Barthel, Fabrice Theoleyre, and Fabrice Valois. Monitoring KPIs in synchronized FTDMA multi-hop wireless networks. In *2016 Wireless Days (WD)*, pages 1–6, March 2016.

[3] Guillaume Gaillard, Dominique Barthel, Fabrice Theoleyre, and Fabrice Valois. *Kausa : KPI-aware Scheduling Algorithm for Multi-flow in Multi-hop IoT Networks*, pages 47–61. Springer International Publishing, Cham, 2016.

[4] Guillaume Gaillard, Dominique Barthel, Fabrice Theoleyre, and Fabrice Valois. High-Reliability Scheduling in Deterministic Wireless Multi-hop Networks. This work has been accepted and presented at IEEE PIMRC 2016 Conference and is eligible for publication in IEEE Xplore®, July 2016.

Normalisation, standardisation

[5] Jonathan Munoz, Emmanuel Riou, Guillaume Gaillard, and Dominique Barthel. Example packets for 6tisch configuration. Internet-Draft draft-munoz-6tisch-examples-00, IETF Secretariat, February 2016. `http://www.ietf.org/internet-drafts/draft-munoz-6tisch-examples-00.txt`.

Rapports de recherche

[6] Guillaume Gaillard, Dominique Barthel, Fabrice Theoleyre, and Fabrice Valois. SLA Specification for IoT Operation - The WSN-SLA Framework. Research Report RR-8567, INRIA, July 2014.

[7] Guillaume Gaillard, Dominique Barthel, Fabrice Theoleyre, and Fabrice Valois. Enabling Flow-level Reliability on FTDMA Schedules with efficient Hop-by-hop Overprovisioning. Research Report RR-8866, INRIA Grenoble Rhône-Alpes, 2016.

FOLIO ADMINISTRATIF

DE L'UNIVERSITE DE LYON OPEREE AU SEIN DE L'INSA LYON

DATE de SOUTENANCE : 19/12/2016

ux de l'Internet des Objets à l'aide de contrats de qualité de service (Service Level Agreements)

Numéro d'ordre : 2016LYSEI152

ths (ED512)

nettre à un opérateur de déployer une infrastructure de réseau radio pour plusieurs applications Objets (IoT). Nous étudions la mutualisation d'une architecture pour différents flux de trafic afin de t du réseau en partageant la capacité des nœuds et une large couverture. Nous devons alors garantir oS) différenciée pour les flux de chaque application.
écifier des contrats de QoS nommés Service Level Agreements (SLA) dans le domaine de l'IoT. Ceux-urs clés de performance (KPI) de délai de transit et de taux de livraison pour le trafic provenant ués géographiquement. Dans un second temps, nous détaillons les fonctionnalités nécessaires à la sur le réseau opéré, sous la forme d'une architecture de gestion de SLA. Nous envisageons flux, l'analyse des performances courantes et la configuration des relais de l'opérateur.
nologie robuste, multi-saut, IEEE Std 802.15.4-2015 mode TSCH, nous proposons un mécanisme permettant de vérifier les différents KPI. Nous utilisons les trames de données existantes comme e réduire le surcoût en termes de ressources de communication. Nous comparons différentes g afin de trouver un compromis entre la performance et l'efficacité de l'observation. Puis nous orithme d'allocation de ressources sous contraintes de QoS multi-flux. Nous dédions des ressources es saut-par-saut pour chaque message. KAUSA prend en compte les interférences, la fiabilité des ttendue afin d'améliorer la répartition des ressources allouées et ainsi prolonger la durée de vie du es gains et la validité de nos contributions par simulation, sur la base de scénarios réalistes de trafic et

Service, Réseaux de Capteurs Sans Fil, Multi-saut, Internet des Objets, Service Level Agreement, rmance, Fiabilité, Gestion de Réseaux, Observation de Réseaux, Ordonnancement, 6TiSCH

che : CITI, Orange Labs

ce Valois

e Owezarski

erry Turletti, Pascale Minet, Pascal Thubert, Philippe Owezarski, Isabelle Guérin-Lassous, Dominique nvité : Fabrice Theoleyre.