Plateforme de coordination et collaboration pour échanger l'énergie, réguler et prédire la production/consommation dans le cadre d'un réseau électrique intelligent

Philippe Glass
encadré par Giovanna Di Marzo Serugendo

UNIGE, Genève, Suisse

Résumé Ce rapport présente la conception d'un modèle de coordination basé sur SAPERE pour permettre les échanges d'énergie électrique entre producteurs et consommateurs à l'échelle d'une maison. En utilisant l'auto-composition de service, les agents producteurs et consommateurs génèrent de manière autonome des contrats d'approvisionnement sous forme de transaction. En parallèle, des agents de supervision contribuent à réguler l'énergie, d'une part en détectant les seuils de dépassement actuels et d'autre part en prédisant les pics de production et consommation à venir en appliquant un modèle d'apprentissage sur l'historique de l'état du nœud. Ce modèle de coordination est conçu également pour permettre les échanges d'énergie entre voisins du même micro-réseau en favorisant la proximité.
Mots-clés: Modèle de coordination, auto-composition, SAPERE, agents, LSA, nœud, Wattage, KWH, Chaîne de Markov, Smart-grids.

Abstract. This report presents the design of a coordination model based on SAPERE to enable the exchange of electrical energy between producers and consumers at the house level. By using service self-composition, producer and consumer agents autonomously generate supply contracts in the form of a transaction. At the same time, supervisory agents help regulate energy, on the one hand by detecting the current exceedance thresholds and on the other hand by predicting future production and consumption peaks by applying a learning model on the node state history. This coordination model is also designed to allow energy exchanges between neighbors of the same micro-grid by promoting proximity.

Keywords: Coordination model, self-organization, SAPERE, agents, LSA, node, Wattage, KWH, Markov chain, Smart-grids.

1 Introduction

Depuis l'avènement des énergies renouvelables, les systèmes de distribution d'énergie entre particuliers sont en plein essor. Cela contribue à améliorer la qualité de vie

des citoyens et à diminuer les coûts en énergies. Ce nouveau type d'énergie offre de nombreuses opportunités économiques et contribue à améliorer la qualité de l'environnement. Le projet « Change38 » lancé à Zurich peut être cité en exemple parmi les nouvelles initiatives qui vont dans ce sens. On développe des applications « Peer-to-peer » sur lesquelles des producteurs et consommateurs d'énergie sont mis en relation pour pouvoir effectuer des transactions : Un consommateur peut décider de démarrer un appareil en utilisant telle offre de tel producteur. Le consommateur et le producteur sont alors informés de la quantité d'énergie échangée et le producteur est rémunéré en conséquence. Certains systèmes d'échange d'énergie offrent également la possibilité de négocier les prix entre voisins et utilisent parfois une monnaie énergétique décentralisée, par exemple le "NRG-coin".

1.1 Le projet TELEGRAM

Dans le cadre de la politique de transition en faveur des énergies renouvelables lancée en Suisse, le projet de recherche TELEGRAM vise à mettre en place des systèmes intelligents de gestion de l'énergie permettant la collecte de données, la prédiction de consommation/production d'énergie, la distribution d'énergie et la négociation de services entre différentes entités tierces pouvant échanger de l'énergie. De tels systèmes doivent répondre à certaines exigences :

- Contraintes techniques : Les agents doivent respecter les lois de la physique qui régissent un réseau électrique : contrôle de la tension, de la fréquence et de l'inertie, contraintes imposées par les entreprises d'électricité, etc.
- Réglementations concernant la tarification de l'énergie : les tarifs devront être de plus en plus fixés par des négociations décentralisées de type "Peer-to-Peer".
- Capacité d'autoadaptation : dans un environnement où les besoins évoluent constamment, les agents doivent être en mesure de prendre en compte tout changement et de collaborer entre eux pour apporter une solution adaptée à la situation.
- Capacité à prédire les pics de consommation en utilisant les technologies de l'information : Les agents doivent apprendre et anticiper la consommation/production locale et du voisinage : une prévision précise est nécessaire pour assurer la stabilité du réseau. Pour ce faire, les agents doivent utiliser des algorithmes appropriés pour un apprentissage global et collaboratif à travers la plateforme TELEGRAM.
- Critères d'acceptation sociale : l'acceptation peut constituer un obstacle à l'adoption de cette nouvelle technologie. La réussite de la transition énergétique dépend aussi des adaptations comportementales.

Le projet TELEGRAM fait à la fois intervenir :

- des institutions publiques telles que l'université de Genève (UNIGE), le regroupement d'écoles HES-SO, l'école supérieure de Lucerne (HSLU), l'école royale polytechnique de Stockholm (KTH)
- des partenaires privés tels que les entreprises Groupe E, SI Nyon, SixSq, EXNATON
- des consortiums tels que l'alliance suisse pour les services intensifs en données

A ce jour, le projet est en cours d'évaluation et l'UNIGE n'a pas encore obtenu de financement.

1.2 Missions du stage

Dans le cadre du stage, on se place sur un modèle de coordination décentralisé permettant l'échange d'énergie entre agents et respectant les recommandations fixées dans le

projet TELEGRAM. Dans ce modèle, les entités peuvent donc coopérer à un niveau local (micro-grid).

Les missions du stage correspondent aux contributions suivantes :

- Mise en place d'un système collaboratif d'échange d'énergie composés d'agents autonomes
- Mise en place d'une application web de type client léger permettant de superviser l'ensemble des agents producteur/consommateur d'énergie sur l'ensemble des maisons
- Mise en place de simulateurs de test permettant d'exécuter un scénario réaliste en se basant sur des statistiques à l'échelle d'une maison
- Mise en place d'un modèle d'apprentissage pour la prédiction de la consommation/production d'énergie au niveau local à une maison
- Mise en place d'un modèle d'apprentissage pour la prédiction de la consommation/production d'énergie du voisinage
- Mise en place d'une stratégie de type "load-balancing" pour répartir un flux de distribution sur plusieurs entités

2 Positionnement dans l'état de l'art

2.1 Les modèles de coordination

Traditionnellement, les systèmes informatiques suivent des architectures centralisées de type client/serveur et utilisent un serveur unique pour traiter toutes les demandes. Sur de tels systèmes, les règles de traitement sont prédéfinies, centralisées en un point unique et la configuration des appareils reliés au système doit être prédéfinie.
Depuis ces dernières années, une multitude d'appareils informatiques ont proliféré et sont devenus omniprésents dans notre environnement quotidien (que ce soient des capteurs, des actionneurs intégrés, des téléphones intelligents, des écrans publics interactifs, des objets intelligents et autres). Cette nouvelle situation conduit à l'émergence de systèmes informatiques denses ayant une infrastructure décentralisée : ces systèmes requièrent une multitude de services qui nécessitent une adaptation permanente à des conditions changeant dynamiquement [12]. Ces nouveaux scénarios ont montré les limites des systèmes centralisés, surtout en termes d'évolutivité, de temps de réponse et de tolérance aux pannes. Ils ont amené à rechercher des solutions alternatives qui répondent à plusieurs exigences :

- La localisation : les services dépendent désormais du temps et de l'espace, étant donné qu'ils sont liés à l'environnement physique et social. Centraliser l'information serait trop coûteux pour la maintenir à jour.
- L'autoadaptation : afin de faire face à l'imprévisibilité de l'environnement, les entités composant le système doivent faire preuve d'autonomie et de proactivité
- La diversité : le modèle doit permettre l'intégration d'appareils hétérogènes et l'injection à la volée de nouveaux services et composants
- La durée : une infrastructure de systèmes omniprésents doit rester évolutif à long terme pour répondre aux nouveaux besoins et aux évolutions technologiques.

Pour répondre à un tel défi, on a cherché à concevoir des modèles de coordination dans lesquels on décentralise les responsabilités (d'où l'émergence de l'auto-organisation). De tels modèles offrent aux systèmes une meilleure évolutivité, une meilleure robustesse et une meilleure répartition des charges de calcul sur les différentes entités.

2.1.1 Une solution qui s'inspire des écosystèmes

L'approche multi-agent a permis de répondre à ce type d'exigence en proposant des

systèmes décentralisés et adaptatifs, favorisant l'interaction et la collaboration entre les entités. De tels systèmes génèrent des services construits à la volée pour réponde spontanément à une demande. De là est né le modèle de coordination : il fournit un cadre composé d'une couche d'abstraction et d'une infrastructure associée qui apporte les technologies de base pour garantir les interactions entre des composants hétérogènes. De tels modèles permettent en particulier de déployer des système multi-agent ouverts à grande échelle. Chaque appareil fournit alors un ensemble de services et envoie des données simples le concernant (propriétés et capacités).
Les modèles de coordination se sont largement inspirés de la nature, et en particulier des systèmes biologiques [17]. De tels systèmes présentent des caractéristiques intéressantes car robustes, résilients, capables de s'adapter aux changements environnementaux et capables d'obtenir des comportements complexes en appliquant des règles élémentaires.
Le modèle biologique peut être représenté en deux couches : les organismes et l'environnement. Afin de créer un modèle informatique inspiré du modèle biologique, une couche supplémentaire a été ajoutée (cf. figure 6). Celle-ci, appelée couche infrastructure, est nécessaire car l'agent logiciel doit être hébergé dans un périphérique.
Les entités utilisées dans le modèle de calcul sont :

- Les Agents : ce sont des entités logicielles autonomes et pro-actives s'exécutant chez un hébergeur.
- L'infrastructure : entité composée d'un ensemble d'hôtes et d'agents d'infrastructure connectés. Une entité hôte fournit des capacités de calcul et communication, des capteurs, des actionneurs tandis qu'un agent d'infrastructure agit au niveau de l'infrastructure pour mettre en œuvre les comportements environnementaux présents dans la nature, tels que la diffusion, l'évaporation et l'agrégation.
- L'environnement : c'est l'espace du monde réel où se trouve l'infrastructure. Un évènement peut être détecté par les agents à l'aide des capteurs fournis par les hôtes.

Des patrons de conception ont été conçus en analysant les interactions entre les mécanismes dans les systèmes auto-organisés existants. Ils ont été obtenus en expérimentant des mécanismes de haut niveau déjà bien connus dans la littérature scientifique.

2.1.2 Première inspiration : le concept de stigmergie

La stigmergie est le premier mécanisme reconnu dans la définition des systèmes de coordination spatiale. Il a été introduit à l'origine par le zoologiste Pierre Paul Grassé [4] et a été observé dans les colonies d'insectes comme les fourmis qui utilisent la phéromone. Le principe de la stigmergie est que la trace laissée dans l'environnement par l'action initiale d'un agent stimule une action suivante, effectuée par le même agent ou un agent différent. De cette façon, les actions successives ont tendance à se renforcer et conduisent ainsi à l'émergence spontanée d'une activité cohérente, apparemment systématique. Ce principe a été abondamment utilisé dans des systèmes intelligents, notamment en robotique, pour coordonner les mouvements d'essaims de robots ou pour aider à trouver des directions dans un environnement inconnu, et dans le domaine des réseaux dynamiques, pour calculer le routage le plus efficace entre deux nœuds.

2.1.3 Les inspirations de la chimie

La chimie a été également une des premières sources d'inspiration dans la définition des modèles de coordination. Elle a été un point de départ pour réaliser des schémas de composition dynamique de service ou d'agrégation de connaissances. Dans cette approche, les réactions chimiques sont vues comme de simples lois qui régulent l'évolution

de phénomènes physiques complexes, coordonnant ainsi les comportements d'un grand nombre de composants. Cela contribue à faire évoluer des systèmes complexes tels que les organismes biologiques ou météorologiques. Gamma [1] a été le premier modèle "chimique" : il considère une machine chimique abstraite dans laquelle les molécules (représentant des entités coordonnées) interagissent selon un ensemble de réactions chimiques. Dans ce système, les états sont vus comme des solutions chimiques.
Ce modèle a été utilisé par exemple dans le routage dynamique avec l'approche des "Fraglet" [14] : Dans cette application, les "Faglet" définies comme des échantillons d'informations et d'instructions de routage circulent à travers le réseau et évoluent après l'application de réactions chimiques successives.

2.1.4 Les inspirations de la physique

D'autres modèles de coordination se sont inspirés de la façon dont les masses physiques se déplacent et s'auto-organisent selon les champs gravitationnels et électromagnétiques [7]. Ces modèles font évoluer les agents à partir de règles de calcul analogues au calcul d'une résultante de vecteurs force en physique.
Par exemple, le modèle Co-fields [8] exploite des champs de calcul composites pour coordonner le mouvement des utilisateurs et des robots dans un environnement.
D'autres approches suggèrent l'adoption de champs de force virtuels pour contrôler la forme de l'auto-assemblage matériel. Le modèle TOTA ("Tuples On The Air") permet de gérer la coordination des agents robots (que ce soit dans les déplacements ou la reconfiguration) en s'appuyant sur des tuples qui se propagent dans l'environnement.

2.1.5 Les inspirations de la biochimie

Les modèles biochimiques sont plus évolués, bien que basés aussi sur les réactions chimiques. Ils permettent de répondre à certaines limitations qui émanent des modèles purement chimiques. L'espace de tuples biochimique [16] améliore l'espace de tuples chimique en y introduisant la distribution et la topologie. Il introduit le compartimentage de l'environnement et la diffusion (empruntés aux modèles de coordination physique) comme notions de premier ordre pour la coordination. Il introduit également la notion de voisinage, permettant de structurer la topologie de coordination, et met à disposition des espaces locaux pour l'exécution de réactions chimiques. Les modèles de coordination biochimiques semblent très flexibles pour permettre la formation spatiale de modèles d'activité à la fois localisés et distribués. Ils ont été exploités dans de nombreux middleware omniprésents pour des usages particuliers, tels que la gestion de la mobilité des foules ou la détection participative.
A ce jour, le modèle SAPERE semble être le seul modèle biochimique qui permet de soutenir une coordination utilisant à la fois la stigmergie, la chimie et la physique. Le middleware de SAPERE est alors choisi comme logiciel de coordination entre agents dans le cadre du stage.

2.2 Approche agent pour les échanges d'énergie

D'après une revue complète de la littérature sur les réseaux intelligents [6], des modèles de réseaux électriques intelligents ont été élaborés en utilisant des agents intelligents comme blocs de construction de contrôle distribué et comme entités clé pour résoudre les différents problématiques à traiter : la sécurité du réseau pour empêcher les intrusions, la gestion de la charge (stockage ou approvisionnement des batteries suivant les besoins), la gestion de l'offre et la demande, la vente de l'énergie, la tarification en fonction de l'affluence de la demande, la gestion des pannes, le contrôle de la fréquence et de la tension. Chaque agent est spécialisé sur une problématique spécifique à traiter et des niveaux hiérarchiques sont utilisés pour les décisions plus impactantes. La

mise en place de telles solutions exige des passerelles entre de nombreux domaines de l'ingénierie.
Une approche holonique :
L'article "A Generic Holonic Control Architecture for Heterogeneous Multi-Scale and Multi-Objective Smart Micro-Grids" [13] présente une architecture holonique générique pour le contrôle des réseaux intelligents. Le modèle présenté permet l'intégration récursive de contrôles hétérogènes à différentes échelles du réseau (au niveau d'un appareil électrique, d'une maison, d'un quartier).
Cette contribution propose un modèle visant à répondre à un certain nombre d'exigences clés du réseau intelligent, telles que :

- la présence d'autorités multiples
- la nécessite de répondre à des objectifs parfois contradictoires
- l'intégration de systèmes hétérogènes
- la nécessite d'adaptation permanente

Une implémentation de prototype a été testée à l'aide d'un simulateur de réseau intelligent, ce qui a permis de montrer :

- la capacité à atteindre plusieurs micro et macro-objectifs, en intégrant des contrôleurs hétérogènes opérant à l'échelle d'une maison et d'un quartier
- la possibilité aux administrateurs d'ajuster dynamiquement les priorités de leurs objectifs afin de rééquilibrer les résultats aux échelles micro et macro
- la possibilité à plusieurs administrateurs de poursuivre des objectifs contradictoires, à différentes échelles, tout en préservant la cohérence au niveau macro (où se situent les principaux intérêts économiques).

2.3 Modèles d'apprentissage utilisés dans la prédiction d'énergie

Le déploiement de plus en plus répandu de capteurs dans les réseaux intelligents pour surveiller la consommation d'énergie a entraîné la production d'une quantité de données sans précédent. A cela, s'ajoute une contrainte de grande volatilité des données dans le temps en raison de l'avènement des énergies renouvelables. Des méthodes d'apprentissage appropriées sont alors nécessaires pour répondre à un tel défi : elles doivent permettre de comprendre des données à la fois volumineuses, dimensionnelles et volatiles, tout en s'exécutant depuis un appareil embarqué comportant des limitations en espace mémoire.

2.3.1 Clustering sur des séries temporelles

L'une des méthodes les plus utilisées pour analyser les données d'énergie sont basées sur du clustering appliqué sur des séries temporelles. L'algorithme de regroupement de données nommé "D-Stream" utilise une approche basée sur la densité des grappes. Cet algorithme positionne chaque enregistrement de données d'entrée dans une grille puis calcule la densité de la grille et regroupe les grilles en fonction de la densité obtenue. Cet algorithme adopte une technique de décroissance de densité pour capturer les changements dynamiques d'un flux de données. Les clusters obtenus sont ainsi ajustés continuellement. Une version de cet algorithme a été mise au point par l'université de San Diego pour gérer la consommation d'énergie résidentielle de maisons situées à Pecan Street (Austin, Texas) [9]. D'après des tests, cette implémentation permet de gérer efficacement les données de consommation d'énergie à grand volume et permet de prédire la consommation quotidienne pour chaque maison. Chaque usager peut alors comparer sa consommation avec celle de ses voisins ou détecter toute anomalie telle qu'un appareil défectueux. Les consommateurs peuvent également être regroupés par profil, afin d'améliorer les politiques de réponse à la demande.

2.3.2 Machine de Boltzmann restreinte

D'autres modèles utilisés dans les réseaux intelligents sont basés sur les machines de Boltzmann. L'algorithme CRBM ("Conditional Restricted Boltzmann Machine") est une méthode d'apprentissage automatique stochastique dérivée des machines de Boltzmann qui a été introduite avec succès pour modéliser les problèmes des séries temporelles. Une machine de Boltzmann est un réseau de neurones artificielles comportant des unités d'énergie sur l'ensemble du réseau. Chaque unité, considérée comme une variable stochastique, produit un résultat binaire. Ce modèle introduit la fonction d'énergie totale qui est calculée en considérant toutes les interactions possibles entre les neurones et les poids et biais. Une étude menée par le laboratoire LTI ENSAJ a montré que la version "FCRBM" ("Factored Conditional RBM") de cet algorithme de prédiction permet d'obtenir des temps de prédiction de moins d'une seconde pour des simulations appliquées à des immeubles et des appareils de domotique [11].

2.3.3 Chaîne de Markov

Des études ont utilisé des chaînes de Markov pour modéliser la production d'énergie renouvelable. Une chaîne de Markov est définie comme un processus aléatoire de Markov à temps discret, dont les transitions sont données par une matrice stochastique. Ces processus vérifient la propriété de Markov, c'est-à-dire que la distribution conditionnelle de probabilité des états futurs ne dépend que de l'état présent et non pas des états passés : cela caractérise "l'absence de mémoire". Ce modèle stochastique permet d'être continuellement entraîné avec des données nouvellement collectées (à la fois un vecteur de poids et des chaînes). Il a été utilisé par l'université de Mashhad pour modéliser la production d'énergie par cellules solaires [18]. Afin de réduire l'erreur de prédiction, les chaînes de Markov sont continuellement entraînées par des données historiques. Les résultats obtenus par l'université confirment les performances des chaînes de Markov pour la production d'énergie. Ce modèle a également l'avantage de pouvoir être glissant dans le temps en lui appliquant une fenêtre d'apprentissage.

3 Modèle de coordination pour l'échange, la régulation d'énergie et la prédiction des pics d'utilisation

3.1 Préambule : présentation des concepts clé

Auto-composition de services :
Dans le cadre d'une application distribuée, une auto-composition de service est un mécanisme qui génère un service à la volée sans intention préalable et sans l'initiative d'une entité de supervision. L'auto-composition utilise les mécanismes suivants :

- un mécanisme basé sur la sémantique qui permet de déterminer la correspondance la plus appropriée en utilisant un ensemble de données de descriptions de services
- une technique évolutive et distribuée qui permet de sélectionner dynamiquement la manière la plus appropriée de composer un service,
- une évaluation de la composition de service pour laquelle la qualité de la composition est basée uniquement sur la réussite de son exploitation.

Un tel type de service a été mis en place pour une application de pilotage de foule [10].

Agents logiciels (ou entités de coordination) :
Entités logicielles actives représentant l'interface entre l'espace des tuples et le monde extérieur, y compris tout type d'appareil (par exemple, des capteurs), de services et d'applications.

Les lois de coordination ("éco-lois" ou "eco-law") :

Mécanismes qui s'appliquent sur les entités présentes dans l'environnement. Les mécanismes existants de la littérature ont été classifiés en tant que patron de conception. Ils sont répertoriés dans 3 couches : la couche inférieure qui contient les mécanismes atomiques, la couche intermédiaire qui contient les mécanismes utilisant ceux de la couche inférieure et la couche supérieure qui contient les mécanismes utilisant ceux des deux couches inférieures.

Modèle de coordination de SAPERE :
Modèle de coordination favorisant l'auto-composition et inspiré des dynamiques bio-inspirées. Il utilise notamment le principe de stigmergie, par le biais d'un tableau noir permettant l'échange d'informations asynchrones, et les réactions chimiques, par le biais des lois de coordination. Dans ce modèle, on définit un service spatial comme un agent fournisseur de service qui contient un ensemble de noms d'entrée et de noms de sortie. Les données d'entrée sont soit générées par un objet intelligent lié à son environnement, soit fournies par d'autres services (éventuellement traitées dynamiquement par des mécanismes d'auto-organisation). Ces données sont réparties géographiquement grâce à un modèle de coordination mettant en œuvre les mécanismes d'auto-organisation. Chaque service a sa propre logique qui est généralement contrôlée par un agent logiciel. Les services spatiaux sont alors le résultat d'interactions collectives entre plusieurs entités basées sur des agents dans un réseau.

Version "Middleware" de SAPERE :
Dernière version de SAPERE, qui est utilisée dans le cadre du stage : elle a été reprise dernièrement par Houssem Ben Mahfoudh [5] pour mettre au point un modèle de coordination pour l'auto-composition spontanée de service dans un système distribué. Cette version héberge un ensemble de mécanismes pré-implémentés qui peuvent être modifiés et améliorés pour divers cas d'utilisations. Les lois de coordination utilisées dans la version Middleware de SAPERE sont les suivantes :

- Bonding : relie un agent à une donnée fournie par un autre agent : l'agent client fait référence à cette donnée car celle-ci le concerne.
- Decay (évaporation) : diminue régulièrement la pertinence des données et supprime finalement les données obsolètes.
- Spreading (propagation) : diffuse des informations au sein d'un réseau à travers les différents nœuds. C'est une diffusion avec des sauts de propagation fixes.

Espace de tuples :
Espace partagé contenant tous les tuples d'un nœud. Il existe un espace partagé pour chaque nœud. Le nœud peut être un Raspberry-pi, un smartphone ou tout autre type d'appareil.

LSA (Live Semantic Annotation) :
Tuple de données et de propriétés dont la valeur peut changer dans le temps. Par exemple la valeur de température injectée par un capteur est mise à jour régulièrement. Un LSA est contrôlé par un agent logiciel.

Wattage :
On définit le "wattage" comme la puissance électrique d'un appareil en Watt à l'instant t. Elle peut varier entre les puissances min et max définies au niveau de l'appareil.

Réseaux intelligents ("smart- grids") :
Ce sont des réseaux d'électricité qui, grâce à des technologies informatiques, ajustent les flux d'électricité entre fournisseurs et consommateurs. En collectant des informations sur l'état du réseau, ils contribuent à une adéquation entre production, distribution et consommation.

KiloWattHeure (KWH) :
Unité d'énergie utilisée sur les réseaux d'échange d'énergie électrique pour évaluer

les consommations ou productions. On obtient le nombre de KWH en multipliant le wattage par la durée en heures de la production ou consommation.
Nœud :
Dans SAPERE : entité locale de l'environnement. On lui associé une adresse réseau et un ensemble de nœuds voisins directs. Dans le cadre du stage, un nœud représente une maison.
État du nœud :
Dans le cadre du stage, on associe un état global au nœud : il comprend les 5 composantes suivantes : le wattage demandé (somme des wattages des requêtes), le wattage produit (somme des wattages des producteurs), le wattage consommé (somme des wattages des requêtes satisfaites), le wattage manquant (demandé - consommé) et le wattage disponible (produit - consommé).

3.2 Conception et implémentation

Le modèle de coordination mis en place utilise les différents mécanismes décrits précédemment pour permettre les échanges d'énergie entre différents agents. Dans la version actuelle du modèle, le périmètre des échanges est celui d'un nœud (représentant d'une maison). Il est possible d'étendre ce mécanisme à l'échelle d'un lotissement en utilisant la loi de coordination "Spreading" (propagation entre nœuds voisins).

3.2.1 Modélisation des acteurs sous forme d'agents autonomes

La modélisation comporte 5 types d'agents (cf. figure 1) : les **consommateurs**, les **agents-contrat**, les **producteurs**, l' **agent d'apprentissage** et le **régulateur**. Ils héritent tous du type générique d'agent défini dans le noyau de SAPER : ils comportent donc les informations suivantes :

- Le nom d'agent
- La liste des noms d'entrée. Les entrées sont les propriétés que l'agent récolte en tant qu'agent abonné après l'exécution de la loi de coordination Bonding (Un LSA abonné récupère une propriété d'un LSA cible)
- La liste des noms de sortie. Les sorties sont les propriétés que l'agent peut fournir à d'autres agents en tant qu'agent de service.
- LSA : comporte les propriétés envoyées dans l'espace de tuples. Elles peuvent être récoltées par les autres agents après l'exécution des lois de coordination (Bonding et Spreading).

Les agents **consommateur**, **contrat** et **producteur** participent aux échanges d'énergies et ont plusieurs instances tandis que les agents **d'apprentissage** et **régulateur** supervisent le nœud et ont une instance unique au niveau du nœud. Ces deux deniers sont instanciés au démarrage du micro-service.
Voici l'ensemble des actions effectuées pour chaque type d'agent :
Producteur :
Un agent **producteur** correspond à un appareil produisant de l'énergie électrique. Il effectue les actions suivantes :

- Il signale tout évènement concernant son activité (démarrage, désactivation, arrêt)
- Il génère des offres afin de répondre aux requêtes d'énergie postées par les consommateurs.
- Après la réception d'un nouveau contrat d'approvisionnement, il envoie une confirmation pour donner son accord. Cette même action est répétée périodiquement, tant que le contrat est valide.
- Après la réception d'un ordre d'interruption provenant du **régulateur**, il met son activité en pause.

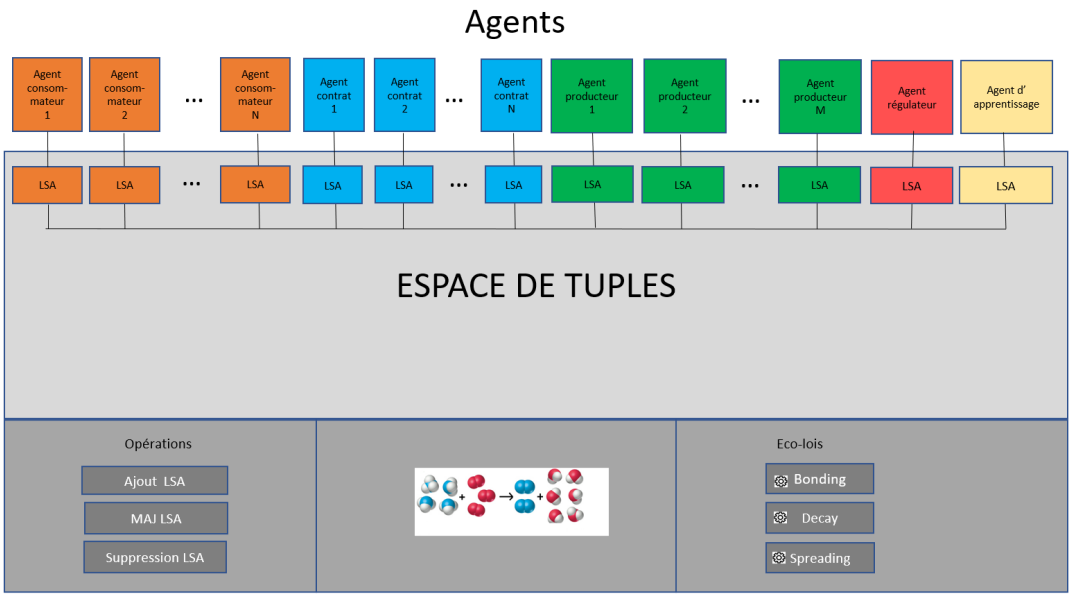


FIGURE 1: Les différents agents

Lorsque l'appareil ne produit plus ou est arrêté, l'instance de l'agent est toujours présente mais son LSA est retiré de l'espace de tuples. Cela permet de ne pas surcharger la quantité de données échangées et de limiter le nombre d'exécutions des lois de coordination. (Chaque loi peut potentiellement être exécutée sur toutes les instances d'agent dont le LSA est présent dans l'espace de tuples).

Consommateur :
Un agent **consommateur** correspond à un appareil consommant de l'énergie électrique. Au niveau du micro-service, le **consommateur** est le seul type d'agent qui émet des requêtes et attend un service en retour. Il hérite alors du type **agent de requête**. Tous les autres types d'agent héritent du type **agent de service** Un **consommateur** effectue les actions suivantes :

- Il signale tout évènement concernant son activité (démarrage, désactivation, arrêt).
- Tant qu'il est insatisfait et que le contrat n'est pas en cours de validation, il s'assure que sa requête est bien émise.
- Tant qu'il est insatisfait, il vérifie si l'ensemble des offres reçues lui permet de satisfaire sa demande. Le cas échéant, il génère un nouveau contrat et le fait valider par l'agent contrat.
- Après la réception du contrat validé, il indique qu'il est satisfait.
- Après la réception d'un ordre d'interruption provenant du **régulateur**, il met son activité en pause.

Lorsque l'appareil ne demande plus d'énergie ou est stoppé, l'instance est toujours présente mais le LSA est retiré de l'espace de tuples.

Agent contrat :
L'**agent contrat** est étroitement lié à un agent **consommateur** : il gère son contrat d'approvisionnement et s'assure que les **producteurs** impliqués sont toujours en accord avec les conditions du contrat, que ce soit lors de la validation du contrat ou durant la période de validité du contrat. On utilise l'agent contrat pour les deux raisons suivantes :

- D'une part, pour décharger l'agent **consommateur** ce certaines actions, comme la vérification des confirmations de producteurs
- D'autre part, le **consommateur**, en tant qu'agent de requête, ne peut déclencher certaines notifications de la loi de coordination Bonding, chose que peut réaliser l'**agent contrat** en tant qu'agent de type service.

L'**agent contrat** effectue les actions suivantes :

- Après la réception d'un nouveau contrat émis par le **consommateur**, il le fait valider par les **producteurs** concernés.
- Tant que le contrat est toujours valide, il le fait reconfirmer périodiquement par les **producteurs** concernés.
- Après la réception d'une confirmation provenant d'un **producteur**, il met à jour le statut du contrat. Il le stoppe immédiatement en cas de désapprobation.
- Après la réception d'un ordre d'interruption provenant du **régulateur**, il stoppe le contrat.

Agent d'apprentissage :
L'agent **d'apprentissage** capitalise les différents évènements survenus au niveau du nœud et génère des données de prédiction.

- Après la réception d'un évènement émis par un **producteur**, **consommateur** ou **agent contrat**, il actualise l'état global du nœud comportant les wattages demandé, produit, consommé, manquant et disponible. Cette donnée est enregistrée en base après chaque actualisation, afin de pouvoir être utilisée ultérieurement soit par l'application web pour afficher l'historique de l'évolution du nœud, soit par l'agent **régulateur** pour vérifier le dernier état du nœud afin de détecter les surconsommation/surproduction.
- A chaque cycle de rafraîchissement, il actualise l'état global du nœud ainsi que les données d'apprentissage du nœud (pour chaque composante : l'état de Markov courant, l'état de Markov précédent et le nombre d'observations). Le modèle d'apprentissage est relancé afin de régénérer une nouvelle prédiction de l'état du nœud à horizon d'une heure.

Agent de régulation :
L'agent **régulateur** supervise le nœud afin d'éviter tout dérèglement. Il effectue les actions suivantes :

- A chaque cycle de rafraîchissement, il récupère l'état du nœud et des agents afin de détecter les dépassements de seuil de production/consommation ou des problèmes divers.
- Après une détection de dépassement, il envoie un ordre de désactivation à certains agents pour pouvoir repasser en dessous du seuil.
- Après une détection de problèmes techniques (LSA hors de l'espace de tuples), il envoie un ordre de désactivation de l'agent pour régulariser le problème.
- Après la récupération d'une prédiction qui alerte une surconsommation ou surproduction à horizon d'une heure, il envoie un ordre pour différer certaines activités afin de pouvoir éviter le dépassement de seuil de manière pro-active.

Implémentation :
Les différents types d'agent sont implémentés dans des classes java qui héritent de la classe mère SapereAgent (cette dernière est définie dans la librairie du middleware de SAPERE).

Agent	Type	Classe java	Déclenchement de l'instanciation
Consommateur	Requête	ConsumerAgent	connexion d'un appareil au nœud
Agent contrat	Service	ContractAgent	idem
Producteur	Service	ProducerAgent	idem
Agent d'apprentissage	Service	LearningAgent	démarrage du micro-service du nœud
Régulateur	Service	RegulatorAgent	démarrage micro-service du nœud

TABLE 1: classes des agents

3.2.2 Architecture logicielle du modèle

Dans le modèle mis en place, le micro-service constitue la clé de voûte : c'est le processus qui fait exécuter le modèle de coordination au niveau d'un nœud, c'est à dire à l'échelle d'une seule maison. Il initialise également les agents qui évoluent ensuite de manière autonome dans l'environnement. Un micro-service est interrogé par des processus clients qui envoient des requêtes pour créer ou mettre à jour ou retirer un agent. Un processus client peut être un simulateur de test ou à une application web qui permet de superviser le réseau intelligent à l'échelle du nœud (cf. figure 9). Un micro-service utilise le noyau de SAPERE comme modèle de coordination.
L'architecture logicielle du modèle comporte donc 3 couches :

- La couche client : processus qui envoie les requêtes pour créer, modifier et retirer des agents
- La couche service : exécute le modèle au niveau du nœud (les agents et leur environnement). Elle utilise le noyau de SAPERE.
- Le noyau de SAPERE (middleware) : librairie qui exécute le modèle de coordination

La couche service, le noyau de SAPERE et les simulateurs de test ont été développés en java. Le framework "Spring boot 2.3.2" a été utilisé pour créer un micro-service sous forme d'API RESTful. L'application web de supervision a été développée sous Angular 9, javascript et D3JS. Tous les échanges de données entre les processus clients et le micro-service se font au format JSON. Chaque contenu JSON est encapsulé dans une requête HTTP qui suit le protocole REST.
Le code source est sauvegardé dans un dépôt git de https://bitbucket.org (branche : "energy")

3.2.3 Les flux d'informations échangées entre les acteurs

Le modèle de coordination utile le principe de stigmergie pour générer des flux d'informations entre les différents acteurs : des agents déposent des données dans l'espace de tuples et après l'exécution des lois de coordination (Bonding ou Spreading), ces données sont récoltées par d'autres agents.
Les échanges de données entre producteurs, consommateurs et agents contrat permettent d'aboutir à un contrat d'approvisionnement en énergie (cf. figure 2). Au départ, la requête est envoyée par le consommateur et les producteurs la récupère dans leurs LSA après l'exécution de la loi de coordination Bonding. Ensuite, les producteurs génèrent et émettent les offres dans leur LSA. A son tour, le consommateur récupère les offres et vérifie si celles-ci permettent de répondre à sa requête. Si les offres conviennent, le consommateur génère un nouveau contrat. L'agent contrat récupère ensuite le nouveau contrat et le fait valider par les producteurs concernés. Chaque producteur envoie sa confirmation et le contrat devient validé dès lors que toutes les approbations sont reçues par l'agent contrat.
Certains échanges d'informations concernent particulièrement l'agent d'apprentissage et l'agent de régulation (cf. figure 3). L'agent d'apprentissage récolte tous les évènements émis par les agents consommateur, contrat et producteur (propriété "EVENT" postée dans le LSA). Les évènements sont historisés en base et permettent de mettre à jour les données d'apprentissage.
L'agent régulateur supervise l'ensemble des productions/consommations à l'échelle

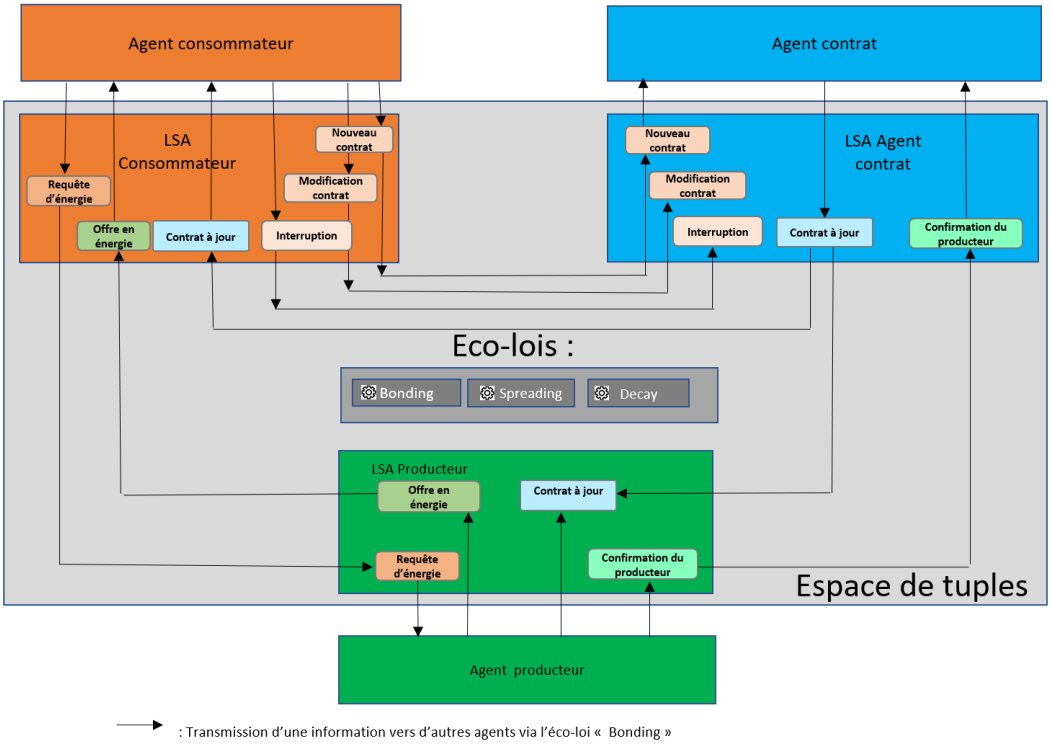


FIGURE 2: Données échangées entre les producteurs, consommateurs, agent-contrat

du nœud. Il génère un ordre de régulation lorsqu'il lève une alerte (Par exemple, la détection d'une surproduction). Une propriété "WARNING" est alors postée dans le LSA. Elle est ensuite récoltée par l'ensemble des agents. Un champ de cette propriété mentionne les agents concernés. De la même manière, le régulateur envoie une replanification d'activité lorsqu'une prédiction de pic est survenue. L'envoi de cette information se fait par le biais de la propriété "RESCHEDULE" qui est ensuite récoltée par les autres agents.

3.2.4 Modélisation des informations échangées entre les acteurs

Par souci de simplicité, cette sous-section ne décrit pas la classe directement déposée dans le LSA car cette dernière gère également la sécurisation des données. De ce fait, les noms de classes indiqués ne comportent pas le préfixe "Protected" qui caractérise une classe de protection de données (cf. section "Protection des données échangées").

NB : toutes les dates sont au format java.util.Date et comportent l'heure avec une précision au millième de seconde et les tables sont au format java.util.Map (table de correspondance clé-valeur).

- Offre émise par un **seul** producteur et envoyée à un **seul** consommateur (classe SingleOffer) Cette classe comporte : le nom du producteur, la requête du consommateur, la date de création, la date d'expiration de l'offre (date création + 10 secondes), la date d'utilisation, la date d'acceptation.
- Confirmation d'un producteur (classe ConfirmationTable) : nom de l'agent producteur, table des confirmations par agent contrat. Une confirmation est modélisée par la classe ConfirmationItem qui comporte l'indicateur d'approbation (OK/KO), la date de confirmation et la date d'expiration de la confirmation.

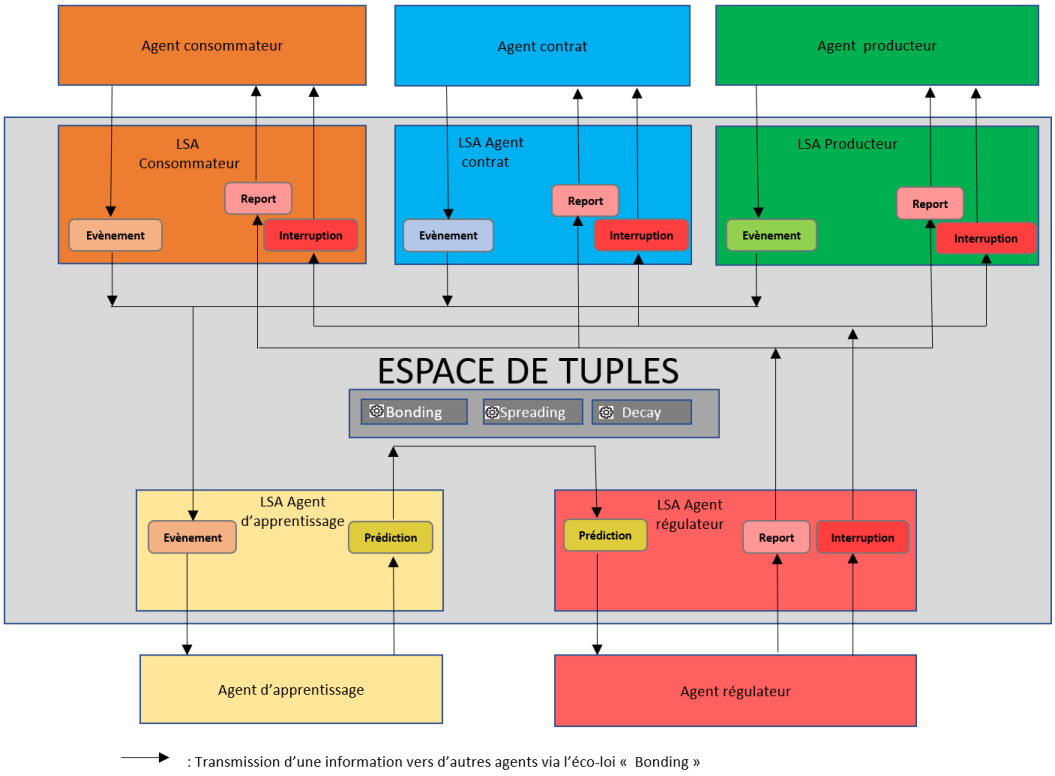


FIGURE 3: Données échangées avec les deux agents de supervision (agent d'apprentissage et agent régulateur

- Contrat d'approvisionnement d'énergie entre un consommateur et des fournisseurs (classe Contrat). Cette classe comporte : le nom de l'agent consommateur, le nom de l'agent contrat, la requête du consommateur, la date de début et de fin de contrat, la table du wattage fourni par chaque agent producteur, la date d'expiration pour la validation du contrat et les approbations/désapprobations des parties prenantes du contrat.
- Données de régulation (classe RegulationWarning). Cette classe comporte le type d'alerte (surproduction, surconsommation, demande d'interruption, demande de modification, LSA non présent dans l'espace de tuples), les noms des agents concernés, la date d'alerte, la date d'expiration de l'alerte et la requête en énergie (renseignée uniquement dans le cas d'une demande de modification de requête ou d'approvisionnement).
- Données des planifications (classe RescheduleTable) : Cette classe est composée d'une table de type clé-valeur. La clé correspond au nom de l'agent et la valeur correspond à la replanification associée. Une replanification est modélisée par la classe RescheduleItem et comporte le wattage ainsi que la plage d'arrêt (date-heure de début et de fin de l'arrêt).
- Données de prédiction (classe PredictionForm). Cette classe comporte la date à laquelle la prédiction est demandée, l'horizon, la date cible (date demandée + horizon), la liste des pas de temps utilisés pour calculer la prédiction, les valeurs de l'état initial du nœud, les états de Markov qui correspondent à l'état initial du nœud et le résultat obtenu, c'est à dire la liste des vecteurs de prédiction de l'état cible. On considère qu'un état du nœud correspond aux 5 composantes suivantes :

wattage demandé, wattage produit, wattage consommé, wattage manquant et wattage disponible.

Le tableau en figure 7 répertorie toutes les données échangées dans l'espace de tuples. L'étiquette correspond au nom de la propriété qui figure dans le LSA.
NB : Certains noms de propriété du LSA du consommateur comportent l'identifiant du consommateur, noté #id. Ces noms de propriété sont donc distincts pour chaque consommateur. Une telle nomenclature a été choisie car elle permet de limiter le nombre de notifications de la loi Bonding aux seuls agents concernés, comme c'est le cas pour les échanges entre l'agent consommateur #id et l'agent contrat #id. Par exemple, le consommateur numéro 3 comporte la propriété "NEW_CTR_3" et seul l'agent contrat n°3 comporte ce même nom de propriété dans les données d'entrée.

3.2.5 Application de l'auto-composition pour générer un contrat

Cette section montre comment l'auto-composition permet d'aboutir à un nouveau contrat valide. Les agents gèrent également les demandes de modification de contrats en cours mais dans ce paragraphe, nous nous limitons à la création d'un nouveau contrat.
Fonctionnement d'un contrat :
Dans la solution proposée, on considère un contrat comme une transaction entre un seul consommateur et plusieurs producteurs. La validation d'un nouveau contrat, qui est gérée par l'agent contrat, requiert l'accord de **toutes les parties prenantes**. L'agent contrat s'assure que les parties prenantes sont bien en accord avec le contrat, que ce soit à la création du contrat pour le valider, ou périodiquement, durant toute la vie du contrat. Le contrat prend fin dès lors que l'une des parties prenantes désapprouve ce dernier.
Dans ce modèle, le consommateur est rattaché à un contrat **unique** et celui-ci doit couvrir l'intégralité du wattage demandé, et ce sur une période qui correspond soit à la plage demandée, soit à l'intersection des plages temporelles des offres. En revanche, un producteur peut être partie prenante de plusieurs contrats et ce dans la limite de ses captativités d'approvisionnement. Si son wattage le permet, un seul producteur peut potentiellement approvisionner tous les consommateurs du nœud.

Les interactions successives :

La figure 4 illustre l'enchaînement des différentes interactions qui permettent d'aboutir à un contrat valide. Afin de simplifier la représentation, on se place dans le cas où l'on a un seul consommateur. Les rectangles colorés représentent les propriétés de LSA qui sont distribuées d'un agent à un autre via la loi de coordination "Bonding". Les rectangles blancs représentent les traitements effectués par les agents.
La génération d'un contrat valide se fait via les étapes suivantes :
1 : Envoi de la requête par le consommateur :
Le consommateur poste sa requête dans la propriété "REQ" de son LSA
2 : Envoi des offres par les producteurs :
La requête du consommateur (propriété "REQ") est récoltée par les différents producteurs.
Après avoir récolté les requêtes (1), chaque producteur génère des nouvelles offres élémentaires pour répondre aux requêtes. Une politique est choisie afin de définir les critères d'ordonnancement des requêtes à traiter en priorité. Une politique peut par exemple classer les requêtes par niveau d'urgence décroissant, puis par wattage croissant. Le producteur génère des offres dans la limite de son wattage disponible (cf.

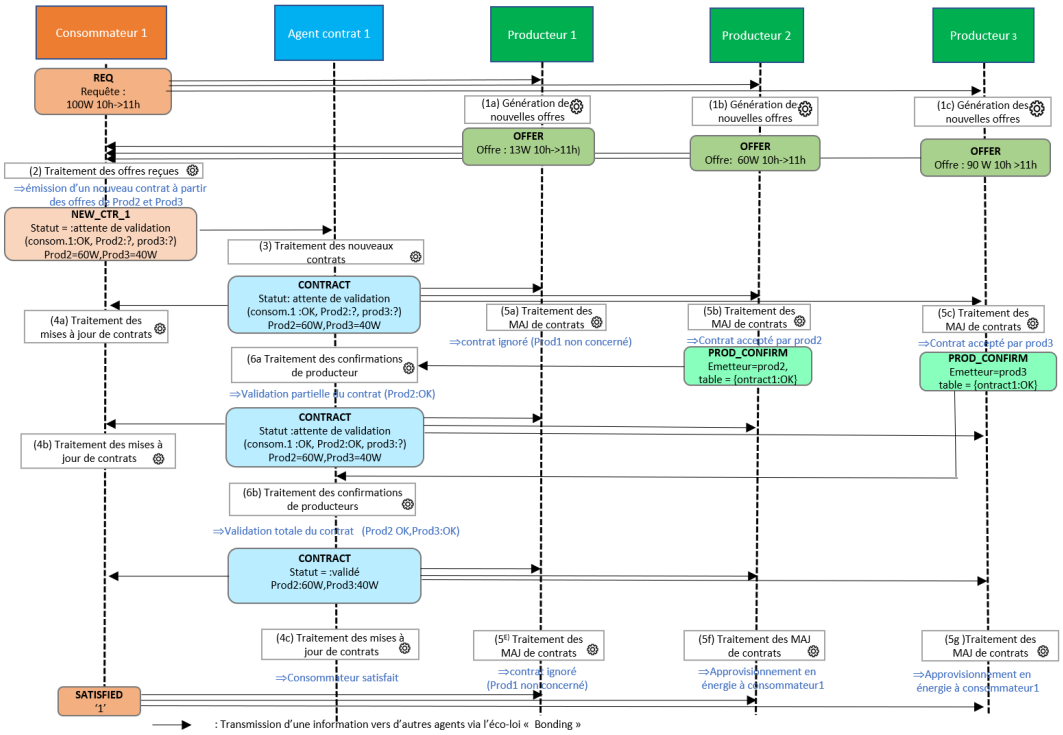


FIGURE 4: Interactions successives pour générer et valider un contrat.

équation wattage disponible au niveau d'un nœud). Tant que son wattage restant est positif, il génère une nouvelle offre, même si celle-ci ne peut couvrir l'intégralité du wattage de la requête.

3 : Émission d'un nouveau contrat par le consommateur :
Le consommateur récolte les offres émises via la propriété "OFFER". De manière périodique, il fait un point sur l'ensemble de ses offres collectées en choisissant une politique qui détermine le critère d'ordonnancement des offres à traiter en priorité. Par exemple elle peut favoriser les énergies solaires et géothermique par rapport aux autres types d'énergie. Le consommateur parcourt les offres de manière itérative en suivant la politique choisie : une offre globale est alors agrégée progressivement. Dès lors que l'offre agrégée permet de répondre au besoin, le consommateur émet un nouveau contrat en se basant sur les offres sélectionnées. Le consommateur poste le nouveau contrat dans son LSA (propriété NEW_CTR_1) afin que l'agent contrat puisse le faire valider. Dans cet exemple, le consommateur a trié les offres de manière aléatoire (Prod2, Prod3, Prod1). L'agrégat (Prod2 + Prod3) permet de satisfaire la requête. Le nouveau contrat engage donc Prod2 et Prod3 comme producteurs.

4 : Validation d'un contrat par l'agent contrat :
L'agent contrat prend connaissance du nouveau contrat en récoltant la propriété "NEW_CTR_#id" émise par le consommateur. Pour faire valider le contrat, l'agent contrat poste ce dernier dans son LSA (propriété "CONTRACT") afin que chaque producteur impliqué puisse recevoir le contenu à jour du contrat et donner ensuite son approbation en retour.
A chaque réception successive d'une confirmation émise par un producteur (propriété

"PROD_CONFIRM"), l'agent contrat met à jour le statut du contrat en y intégrant l'approbation (ou désapprobation) du producteur. Le nouveau contenu du contrat est alors reposté dans le LSA afin que toutes les parties prenantes du contrat aient la version à jour.
Le contrat est validé uniquement lorsque tous les producteurs ont répondu positivement. Le cas échéant, toutes les parties prenantes reçoivent un contrat validé : le contrat prend alors effet et les producteurs peuvent lancer les approvisionnements en énergie. Si l'une des parties prenantes désapprouve le contrat, ce dernier est annulé et un nouveau cycle d'appel d'offres démarre aussitôt.

5 : Confirmation du contrat par le producteur :
Le producteur se met à l'écoute de la propriété "CONTRACT" émise par chaque agent contrat. A chaque réception de donnée de contrat le concernant, le producteur vérifie si le wattage disponible lui permet d'approvisionner le contrat. Il intègre le résultat de cette vérification sous forme de confirmation (approbation ou désapprobation) dans sa table de confirmation. Il émet ensuite le nouveau contenu de sa table de confirmation dans la propriété "PROD_CONFIRM" de son LSA.

3.2.6 Protection des données échangées

Afin de répondre aux critères d'acceptation social, des règles de confidentialité sont appliquées sur les données échangées au travers de l'espace de tuples, comme les offres, les contrats ou les confirmations de producteurs. Ces règles consistent à protéger individuellement chaque propriété d'un tuple : l'accès à la propriété est restreint aux seuls agents concernés. On part du principe qu'une information confidentielle concernant un agent ne peut être rendue visible à ses homologues.
Par exemple, sur un contrat, la table du wattage fourni par chaque producteur n'est pas être accessible à l'ensemble des producteurs. Un producteur peut simplement savoir s'il est concerné ou non par le contrat, et le cas échéant, il peut connaître uniquement le wattage qu'il doit approvisionner (cf. figure 8). Il n'a donc pas accès à l'information concernant les autres éventuels producteurs impliqués. De manière analogue, une offre émise par un producteur à un consommateur n'est pas accessible à un autre producteur ou à un autre consommateur.
Implémentation :
Pour envoyer des propriétés protégées dans un LSA, on encapsule l'objet à protéger dans un objet dont la classe gère la protection des données. Par exemple, un contrat (classe Contract) est encapsulé dans un objet de la classe ProtectedContrat. Chaque classe de protection comporte l'objet à protéger comme attribut privé et implémente les méthodes d'accès aux informations sensibles. L'agent demandant l'accès à la donnée doit figurer en paramètre de la méthode.

De cette manière, la méthode d'accès vérifie si l'agent est bien authentifié au niveau du micro-service et le cas échéant, elle détermine la visibilité de l'agent sur la donnée sollicitée. La donnée retournée est restreinte suivant la visibilité obtenue et une exception de sécurité est envoyée si l'agent n'a aucun droit d'accès sur la donnée (cf. figure 5).

3.2.7 Régulation

- **Détection des surconsommations/surproductions** :
  La détection des pics de production ou consommation s'effectue par l'agent de régulateur. Les étapes sont les suivantes :
  Détection des dépassements

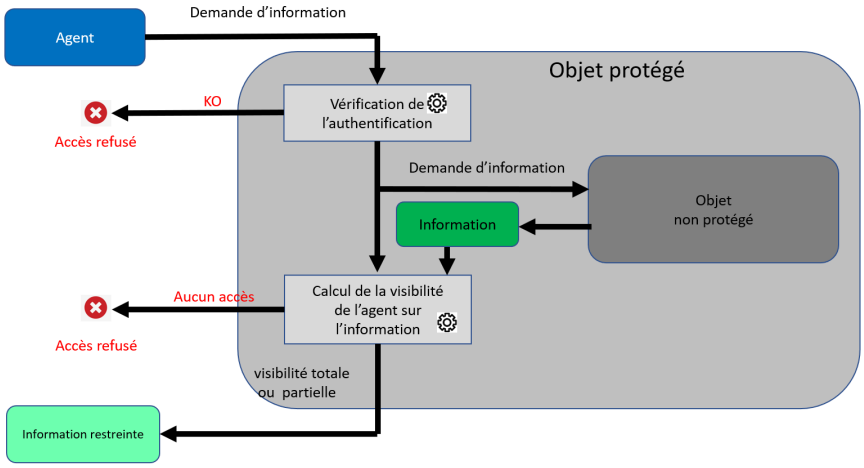


FIGURE 5: Protection d'un objet envoyé dans un LSA

A intervalles de temps régulier, le régulateur récupère l'état global du nœud qui a été enregistré en base par l'agent d'apprentissage. A partir des différents totaux, le régulateur détecte les productions et consommations qui dépassent le seuil d'alerte (3 000 Watts).

Choix des agents à désactiver

En cas de dépassement, une politique est appliquée pour déterminer les agents à stopper en priorité afin que le nœud puisse repasser en dessous du seuil d'alerte. Le régulateur obtient alors une liste ordonnée d'agents. Par exemple, le régulateur peut stopper en priorité les producteurs ayant la puissance la plus élevée, ou au contraire, les producteurs ayant la puissance la moins élevée, ou des producteurs choisis de manière aléatoire. Une fois la liste ordonnée obtenue, le régulateur choisit les premiers agents de la liste afin de pouvoir redescendre en dessous du seuil d'alerte (cf. algorithme 1) dans le cas d'une surproduction).

Désactivation des agents en dépassement

En cas de dépassement, un objet de classe RegulationWarning est instancié et posté dans la propriété "WARNING" du LSA. Une fois la loi de coordination Bonding appliquée, chaque agent abonné aux propriétés "WARNING" récupère à son tour l'objet RegulationWarning posté précédemment et vérifie s'il est bien concerné. Le cas échéant, l'agent stoppe son activité de production/consommation pendant un certain laps de temps. Cela entraîne l'interruption immédiate du (ou des) contrat(s) impliquant l'agent. Par exemple, en cas de surproduction, le producteur arrêté par le régulateur doit stopper immédiatement tous les contrats pour lesquels il est concerné. De ce fait, chaque consommateur qui étaient précédemment approvisionné par ce producteur doit emmètre à nouveau sa requête dans l'espace de tuples afin de rechercher un autre approvisionnement.

Durée de la désactivation

L'agent désactivé vérifie de manière périodique si l'état actuel du nœud lui permet à nouveau de relancer son activité. Le wattage actuel doit être suffisamment en dessous du seuil d'alerte pour permettre à l'agent de s'activer à nouveau sans entraîner un nouveau dépassement.

– **Prédiction des pics de consommation/production**
Le régulateur récupère périodiquement le résultat de la prédiction émise dernièrement par l'agent d'apprentissage (propriété "PRED"). Lorsqu'un pic est prévu à ho-

rizon d'une heure, le régulateur détermine la liste des agents à replanifier de manière à réduire la production (ou consommation) de 20% du seuil maximal, soit 600 Watts. Il choisit une politique pour prioriser les agents à replanifier, par exemple par ordre de wattage croissant. Ensuite, la table de replanification est modifiée, puis postée dans le LSA du régulateur (propriété "RESCHEDULE"), puis récupérée par l'ensemble des agents. Les agents concernés modifient alors leurs dates de fin d'activité de manière à stopper 10 minutes avant l'heure du pic prévu. Par défaut, ils pourront reprendre leurs activités respectives seulement après la fin de la plage d'arrêt (soit une heure après l'heure du pic prévu). Les contrats liés aux agents concernés peuvent être également impactés suivant la plage horaire de l'arrêt prévu (anticipation de la date de fin de contrat).

3.2.8 Base de données

Une base de données relationnelle de type MariaDB est utilisée pour enregistrer et historiser certaines données générées par le micro-service : ces données concernent les évènements des agents d'énergie (producteurs/consommateurs/contrats), les offres générées par les producteurs, l'historique de l'état du nœud (puissance demandée, produite, consommée ...), les données d'apprentissage des chaînes de Markov, le référentiel des appareils électriques et des types d'énergie.
Des clés primaires de type auto-incrément sont utilisées pour identifier chaque donnée et des contraintes de clé étrangère sont utilisées pour relier certaines tables (relations de composition entre 2 données).
Les figures 10 et 11 représentent les deux principales grappes du diagramme de relations entres les entités :

- Celle qui concerne l'historique des échanges d'énergie : les données sont articulées autour de la table des évènement survenus sur les agents consommateur/producteur/contrat (Event).
- Celle qui concerne les données d'apprentissage : les données sont articulées autour de la table des matrices de transition : on a une matrice de transition par tranche horaire et par composante du nœud (les 5 composantes "requested", "produced", "missing", "consumed", "available"). La table transition_matrix_cell stocke le contenu de chaque élément d'une matrice de transition, la table transition_matrix_iteration identifie un cycle d'itération qui se rattache à la matrice de transition.

3.2.9 Implémentation d'une application web de suivi

Une application web a été implémentée pour pouvoir suivre l'état d'un nœud à l'instant t et son historique depuis le lancement du micro-service. Cette application utilise les technologies Angular 9, javascript, Node.js et D3JS. Chaque page web suit le patron de conception MVC (modèle-vue-contrôleur). Il a l'avantage de définir un couplage faible entre la présentation, les données, et les composants métier. La partie « vue » est déclarée dans une version étendue du HTML traditionnel qui comporte de nouvelles balises (tags) et attributs. Ce langage HTML étendu est utilisé pour déclarer une liaison de données bidirectionnelle entre les modèles et les vues. Ainsi, les données sont synchronisées automatiquement. Les services sont fournis au composant via le système d'injection de dépendances. Les structures des vues sont codées dans les fichiers #nom_module.component.html et sont associés aux contrôleurs codés dans les fichiers #nom_module.component.ts.
Les principaux composants sont :

- La vue instantanée du nœud (cf. figure 19) : permet de faire un suivi des appareils consommateurs et producteurs rattachés au nœud ainsi que du total du

nœud (wattage demandé, produit, consommé, manquant et disponible). Cette page donne la possibilité d'effectuer des opérations sur les agents du nœuds : ajout d'un nouvel appareil producteur/consommateur, modification d'un appareil (par exemple le wattage), désactivation, redémarrage.
- L'historique du nœud (cf. figure 12) : affiche sous forme de table la chronologique de tous les évènements survenus sur le nœud. Les lignes affichent l'état du nœud à chaque date-heure d'enregistrement : le wattage total demandé, produit, consommé, manquant et disponible ainsi que la liste des requêtes non satisfaites, la liste des requêtes "en alerte" (c'est à dire non satisfaite et dont le wattage est inférieur au total disponible du nœud). Une visualisation graphique comporte les courbes d'évolution de chaque composante de l'état du nœud au cours du temps. Elle est générée à l'aide de la librairie d3js.
- La visualisation instantanée des LSA rattachés au nœud avec le contenu de chaque propriété
- Le contenu instantané de l'ensemble des matrices de transition des chaînes de Markov (cf. figure 14). On a une matrice par tranche horaire et par composante de l'état du nœud. Chaque matrice comporte les probabilités de chaque transition possible d'un état de départ vers un état d'arrivée.

3.3 Apprentissage des données de l'état du nœud

Dans la pratique, le modèle d'apprentissage des chaînes de Markov est utilisé par un agent fournissant un service pour prédire un état qui correspond à une tranche de disponibilité de service, et ce à une heure, jour et saison donnés. Dans un système de coordination, chaque agent prédit la disponibilité de son propre service et peut utiliser la prédiction cumulative afin d'agréger son résultat avec les prédictions des autres agents.
Une base multi-agents de services spatiaux collecte et propage les requêtes et prédictions vers les agents voisins contenu dans l'environnement. Par exemple, ce modèle a été utilisé pour tester un système de prédiction de disponibilité de différents parcs de stationnement d'un centre commercial pour aider à désengorger la circulation [15]. Chaque agent de parking génère la prédiction de sa disponibilité et peut agréger ses résultats avec ceux des autres agents de parking, afin de pouvoir fournir une solution optimale pour répondre à une requête d'un usager.
Dans la contribution du stage, le modèle des chaînes de Markov est utilisé pour prédire chacune des 5 composantes de d'état du nœud (wattage demandé, produit, consommé, manquant et disponible).
Ce modèle peut être glissant dans le temps, c'est à dire que l'on peut se baser sur les N dernières itérations pour générer la matrice de transition qui définit l'ensemble de probabilités de passage d'un état de Markov à un autre.

3.3.1 Définition des états de Markov

On définit l'ensemble des états de Markov de manière identique pour les 5 composantes du nœud : c'est l'ensemble des états que chaque composante peut potentiellement prendre. L'ensemble des états représente le niveau de granularité de la prédiction. Plus le nombre d'états est grand, plus la prédiction est précise mais plus les ressources sont sollicitées. Afin de rester sur un volume de données et un temps de calcul raisonnables, on se limite à 7 états, chaque état représentant un intervalle de valeurs que peut prendre la composante du nœud à prédire (cf. tableau 6). Le premier état correspond à la valeur 0, les 5 états suivants correspondent aux 5 tranches dont la longueur correspond à 20 % du seuil d'alerte et la dernière tranche correspond aux valeurs au-delà du seuil d'alerte.

3.3.2 Mémorisation des nombres d'observations

Dans le modèle de prédiction, on définit le pas de la variable temps à une minute : il correspond à l'intervalle de temps entre deux observations de l'état du nœud. On définit une itération comme l'intervalle de temps durant lequel les observations sont rattachées à la même matrice de transition. Afin de rester sur une quantité de données raisonnable, chaque jour est décomposé en 19 tranches horaires qui correspondent à des créneaux d'une heure hormis la plage [0h-6h]. On a donc les 19 tranches horaire suivantes : [0-6h], [6h-7h], [7h-8h], ... , [23h-0h]. Si certaines variables varient beaucoup à l'intérieur d'une même plage située en heures de pointe, peut-être qu'il serait judicieux de redécouper certaines tranches. Une itération correspond à un jour et une plage horaire donnés. A chaque jour suivant on a donc 19 nouvelles itérations pour chacune des 5 composantes.
NB : l'implémentation n'intègre pas encore les variables météorologique (température, pression, humidité). De ce fait on se limite à une matrice de transition par tranche horaire et par composante (soit 5 x 19 = 95 matrices).
A chaque minute, l'agent d'apprentissage rafraîchit l'état du nœud en calculant le total du wattage demandé, produit, consommé, manquant et disponible. A chaque valeur en Watt correspond un (et un seul) état de Markov. Pour chacune des 5 composantes, l'agent d'apprentissage calcule la transition d'état qui correspond au couple [état de Markov précédent (qui a été mémorisé), l'état de Markov actuel]. Le nombre d'observations associé à l'itération courante et à la transition [état de Markov précédent, état de Markov actuel] est alors incrémenté de 1. Les nombres d'observations sont enregistrés en base dans la table transition_matrix_cell_iteration.

3.3.3 Matrices de transition

Les matrices de transition d'état sont construites à partir des expériences recueillies en notant la transition [État précédent, Nouvel état]. Pour chaque composante et chaque tranche horaire, la matrice de transition résulte des nombres d'observations récoltées sur les "N" dernières itérations : N correspond à la fenêtre d'apprentissage associée à la chaîne de Markov. Cela signifie que l'on se limite aux itérations des "N" derniers jours. Dans no implémentions, la fenêtre d'apprentissage par défaut est de 100.
La matrice de transition est calculée en deux étapes :

- Somme des nombres d'observations pour chaque transition (État précédent, État suivant) en restant dans la fenêtre d'apprentissage
- Normalisation des valeurs obtenues afin d'obtenir des probabilités de transition : nombre d'observations état i vers j / Somme (nombre d'observations en ligne i)

Couverture de l'ensemble des états :
Pour que le modèle soit utilisable quelle que soit la tranche horaire et composante, la matrice de transition générée doit être complète, c'est à dire sans aucune ligne "vide" (une ligne vide comporte uniquement des 0). Cela signifie que pour l'ensemble des 19 tranches horaires et des 5 composantes, chaque état doit figurer en tant qu'état de départ dans au moins une transition ayant un nombre d'observations supérieur à 0. Dans la plupart des scénarios, certains états ne sont pas explorés ce qui est particulièrement le cas pour les états "extrêmes" (l'état 1 : 0 Watt et l'état 7 : >3000 Watts). Pour remédier à ce problème, des scénarios de test spécifiques ont été créés afin d'aider un nœud à atteindre certains états relativement peu accessibles, ce qui permet d'obtenir des nouvelles transitions depuis n'importe quel état. Par exemple, un tel scénario peut générer un nouveau producteur de 3500 Watts : la composante "produced" du nœud atteint alors l'état 7 en dépassant les 3000 Watts (surproduc-

tion).
L'ensemble des 95 matrices de transition générées à l'instant t figure sur une page de l'application web (cf. figure 14). Chaque colonne correspond à l'une des 19 tranches horaires et chaque ligne correspond à l'une des 5 composantes.

3.3.4 Mécanisme de prédictions

Pour chaque composante, on peut calculer une prédiction à une date cible en utilisant les matrices de transitions. Le résultat de la prédiction est obtenu de manière itérative. C'est un vecteur à 7 composantes qui correspondent aux probabilités d'obtenir chacun des états de Markov depuis l'état initial. Au départ, on lui affecte sa valeur initiale : des valeurs nulles, hormis un 1 situé à l'index de l'état de Markov actuel.
On découpe la plage de temps [heure de départ - heure cible] en pas élémentaires d'une minute. A chaque pas, on détermine la matrice de Markov associée (en considérant la plage horaire dans laquelle le pas est situé) et on actualise le vecteur de prédiction en le multipliant par la matrice de Markov associée. Le contenu de la matrice à multiplier change uniquement lorsque l'on change de tranche horaire. Si l'implémentation prenait en compte les variables météorologiques, la matrice à multiplier changerait également à chaque changement de tranche de température, de pression, d'humidité etc... A la dernière itération, on obtient alors le vecteur de prédiction qui correspond à la date cible. Il constitue le résultat de la prédiction.

3.3.5 Report des consommations et productions

L'agent régulateur récupère le résultat de prédiction en récoltant la propriété "PRED" précédemment postée par l'agent d'apprentissage. Lorsqu' un dépassement en production ou consommation devient fortement probable dans l'heure qui suit, une alerte de régulation est postée afin de demander à certains agents de reporter leurs activités prévues pour éviter la plage horaire concernée. Pour ce faire, l'agent régulateur s'assure que l'agent concerné peut effectivement différer son activité, ce qui n'est pas le cas pour certains appareils "prioritaires" comme les réfrigérateurs. La possibilité de différer ou non l'activité correspond au niveau de priorité du consommateur/producteur. Ce mécanisme permet donc d'éviter les pics de production/consommation de manière pro-active en différant l'activité de certains agents et en se basant sur la prédiction.

3.4 Implémentation de simulateurs de test

Plusieurs simulateurs ont été développés afin de voir et comprendre comment les agents réagissent pour répondre aux différentes demandes. Chaque simulateur est un processus client du micro-service : en exécutant son scénario, il injecte des données dans une requête HTTP pour créer des agents, puis éventuellement les modifier, les stopper ou les redémarrer. Les données sont envoyées au format JSON. A chaque boucle itérative du simulateur, la constitution du nœud évolue : des appareils s'ajoutent, d'autres sont stoppés, d'autres modifient la puissance demandée. L'application web permet de suivre l'état courant et l'historique du nœud.
Plusieurs simulateurs ont été développés afin de pouvoir tester le modèle dans différentes conditions et en se focalisant sur des problématiques différentes.

Scénario 1 :
Il s'agit d'un test ponctuel. Il lance simultanément 10 consommateurs de wattage légèrement différents (environ 30 Watts chacun) et 10 producteurs (25 Watts chacun). Ce test se focalise sur le temps d'alerte maximal : il évalue le temps mis par

l'ensemble des agents pour satisfaire au mieux les consommateurs dans une situation de manque au niveau du nœud. (Wattage produit = 250, wattage demandé = 300). Une variante de ce scénario (le 1-bis) ajoute une difficulté en émettant une nouvelle requête de priorité maximale, une fois que toutes les autres requêtes en alerte soient satisfaites. Tous les producteurs sont ensuite obligés d'interrompre les approvisionnements en cours afin de répondre en priorité à la nouvelle requête (y compris pour la politique aléatoire). Cette variante teste donc la capacité des agents à répondre à une situation d'urgence dans un contexte plutôt difficile.

Scénario 2 :
Ce scénario s'inscrit dans la durée. Il génère périodiquement des agents de manière aléatoire et en plus grand nombre. Il favorise le wattage demandé plutôt que le wattage produit, ce qui entraîne également un manque au niveau du nœud. Cela rend donc la génération de contrats plus difficile (La difficulté augmentant également avec le nombre de requêtes à traiter).

Scénario 3 :
Ce scénario s'inscrit également dans la durée et cherche à reproduire des données de consommation et production en étant le plus réaliste possible. Le simulateur se base sur des statistiques réelles de productions et consommations qui ont été intégrées en base de données [3]. Ces données comportent la puissance moyenne consommée ou produite à chaque heure et pour chaque catégorie d'appareil d'une maison. Le simulateur effectue un traitement périodique pour créer et mettre à jour l'ensemble des consommateurs et producteurs rattachés au nœud. A intervalle de temps régulier, le simulateur effectue le traitement suivant :

- Récupération des moyennes statistiques de production/consommation par catégorie d'appareil qui correspondent à l'heure courante
- Traitement suivante appliqué pour chaque catégorie d'appareil :
  - Choix aléatoire de la puissance totale "cible" associée à la catégorie. Le tirage aléatoire utilise une loi normale gaussienne centrée sur la moyenne statistique et dont la déviation est réglée de manière à avoir 70 % de chance de rester dans la plage +-10 % de la moyenne statistique
  - Calcul de la puissance totale des appareils en cours de fonctionnement sur la catégorie
  - Ajustement des puissances des appareils en cours de fonctionnement de manière à se rapprocher de la puissance cible (Sur chaque appareil, la puissance est réglable entre une valeur minimale et maximale)
  - Tant que [puissance réelle > cible] : arrêts d'appareils pour se rapprocher de la cible
  - Tant que [puissance réelle < cible] : démarrages d'appareils pour se rapprocher de la cible
  - Si [valeur absolue (puissance réelle - cible) > 0.05 X cible] : re-application du même traitement en repartant de 0 appareils.

Une variante de ce scénario (le 3-bis) utilise un mode "dégradé" en appliquant un coefficient pour réduire les wattages des producteurs : cela génère un manque au niveau du nœud.
D'autre scénarios ont également été implémentés dans le but de tester la capacité à répondre à certaines problématiques, par exemple la gestion d'un dépassement de seuil d'alerte (surproduction, surconsommation).

4 Présentation des résultats de la contribution

4.1 Évaluation des "échecs" en termes de satisfaction de requête

Le premier objectif de la contribution est de pouvoir satisfaire un maximum de consommateurs tout en évitant les dépassements de seuil en termes de production ou de consommation.
On qualifie une requête comme étant en alerte si celle-ci est non satisfaite alors que le wattage disponible au niveau du nœud permet encore de la satisfaire. Les requêtes que l'on ne peut pas satisfaire (c'est à dire dont le wattage est supérieur au wattage total disponible du nœud) ne sont pas concernées. On considère que le nœud est "en échec" si au moins une des requêtes est en alerte au niveau du nœud. Dans un premier temps, on évalue la durée maximale d'alerte sur l'ensemble des requêtes : cela correspond au maximum des temps mis pour chaque requête en alerte à être satisfaite. Ensuite, on évalue le taux d'échec qui quantifie l'ensemble des échecs survenus sur l'historique. Le taux d'échec se base

- soit sur les durées : rapport entre le total des durées durant lesquelles le nœud est en échec et la durée totale (cf. formule base temps)
- soit sur l'énergie : rapport entre le total en KWH des requêtes en alerte et le total produit en KWH sur la durée totale de l'historique (cf. formule base kwh)

En termes d'évaluation, le taux base temps est beaucoup plus "sévère" car il considère le nœud comme étant totalement en échec si au moins une requête est en alerte. Il ne prend pas en compte la proportion des requêtes en alerte par rapport au total produit.
Afin de pouvoir obtenir une évaluation, ces résultats sont affichés sur la page d'historique du nœud. Une amélioration possible consiste à enregistrer périodiquement en base ces résultats.

Des tests ont été exécutés pour comparer différentes politiques de génération des offres :

- Politique "aléatoire" : Les requêtes sont ordonnées de manière aléatoire pour un même niveau de priorité.
- Politique de "priorisation par niveau d'alerte" : Les requêtes sont ordonnées par niveau de priorité décroissant, durée d'alerte décroissante et wattage croissant. La durée d'alerte correspond à la durée (en secondes) depuis laquelle la requête est en alerte, c'est à dire insatisfaite alors que le wattage disponible est suffisant au niveau du nœud.
- Politique "hybride" : par défaut, elle utilise la politique aléatoire et lorsque qu'une requête insatisfaite atteint le seuil d'alerte de 8 secondes, elle bascule automatiquement sur la politique de priorisation par niveau d'alerte.

La suite du paragraphe présente les résultats obtenus sur les différents scénarios.

Scénario 1 (test ponctuel avec un manque) :

	scénario 1			scénario 1-bis		
Politique	Meilleur score	Moins bon score	Moyenne	Meilleur score	Moins bon score	Moyenne
Aléatoire	8	63	23	15	182	55,9
Priorisation par niveau d'alerte	16	17	16,5	16	17	16,7
Hybride	9	17	14,2	9	18	15,8

TABLE 2: Temps d'alerte max (sec.) obtenus sur 10 tests des scénarios 1 et 1-bis.

Le tableau 2 montre les temps maximums d'alerte obtenus pour les scénarios 1 et 1-bis. La politique aléatoire peut obtenir de très bons résultats mais s'avère risquée

car elle obtient parfois des temps très mauvais, en particulier pour le 1-bis (182 secondes).
la politique de priorisation par niveau d'alerte se focalise sur les requêtes à traiter en priorité. C'est la stratégie la plus sûre mais elle ne peut obtenir de très bons résultats, étant donné que les agents se focalisent tous sur un seul consommateur, le "plus en alerte" à l'instant t. Au niveau du nœud, les requêtes sont alors traitées "une par une" et non en parallèle comme c'est le cas pour la première politique.
La politique hybride semble être un bon compris : elle obtient souvent d'assez bon résultats tout en retirant le risque d'ignorer les requêtes les "plus en alerte" car elle bascule dynamiquement vers la politique de priorisation à partir d'un certain seuil d'alerte. La comparaison entre les deux scénarios montre que la politique aléatoire est nettement impactée par l'augmentation du nombre de contraintes, ce qui n'est pas le cas pour les autres politiques qui maintiennent des performances quasi similaires.

Scénario 2 (test dans la durée avec des manques) :
Le tableau 3 présente les résultats obtenus sur des sessions de 10 minutes.

	Durée d'alerte max (sec)			% échecs base durée			% échecs base KWH		
Politique	Meilleur score	Moins bon score	Moyenne	Meilleur score	Moins bon score	Moyenne	Meilleur	Moins bon score	Moyenne
Aléatoire	80	181	141	50,66	82,83	67,62	2,51	8,63	5,92
Priorisation par niveau d'alerte	9	15	11,75	8,68	12,8	10,5	0,39	0,84	0,61
Hybride	21	17	19	25,37	29,23	26,7	1,54	2,49	1,84

TABLE 3: Résultats obtenus sur 10 tests du scénario 2 exécute pendant 10 min.

Les résultats sont différents par rapport à ceux des scénarios 1 et 1-bis : ils positionnent en tête la politique de priorisation par niveau d'alerte. On peut remarquer que l'utilisation de l'aléatoire est encore plus problématique que pour le scénario 1-bis et entraîne des temps d'alerte très longs (moyenne à 141 secondes, maximum à 181 secondes). Étant donné le volume de requêtes plus important et le stock disponible généralement insuffisant, il est plus difficile de fournir des offres qui permettent de répondre à la requête ayant le plus grand niveau d'alerte. En effet, il faut généralement plusieurs offres destinées à cette même requête (voir plus d'une dizaine) pour obtenir le wattage suffisant. Dans le cas d'une politique aléatoire, le consommateur concerné a donc une faible probabilité d'obtenir suffisamment d'offres.

Scénario 3 (simulateur réaliste) :
En raison du nombre d'agents plus important que pour les scénarios précédents, (entre 100 et 150), un certain nombre de problématiques techniques ont été identifiés et résolues après différents tests.
Le premier bloc du tableau 4 présente les derniers résultats obtenus pour le scénario 3.

	Scénario 3			Scénario 3-bis (3 "dégradé")		
Politique	Temps d'alerte max(sec)	% échecs base temps	% échecs base KWH	Temps d'alerte max(sec)	% échecs base temps	% échecs base KWH
Aléatoire	25	18,94	0,59	225	76,27	6,12
Priorisation par niveau d'alerte	13	16,31	0,61	29	33,18	2,19
Hybride	13	17,43	0,64	38	43,03	2,48

TABLE 4: Résultats obtenus sur les scénarios 3 et 3-bis testés sur 60 min.

Les politiques "priorisation par niveau d'alerte" et "hybride" semblent obtenir des résultats comparables alors que la politique aléatoire obtient des résultats toujours moins bons mais largement corrects en comparaison des très mauvais résultats obtenus dans le scénario 2. Sur les différents tests effectués, il s'avère que les manques sont

moins fréquents que pour le scénario 2. En se basant sur des valeurs proches des statistiques fournies, les quantités produites permettent de répondre généralement aux besoins. De ce fait, les manques sont moins fréquents et la stratégie aléatoire est moins pénalisée par la difficulté à obtenir un contrat lorsque l'on a plusieurs requêtes à satisfaire.
Des tests comparatifs ont été réalisés sur le scénario "3-bis" qui correspond au 3 dégradé (productions diminuées de moitié, de manière à entraîner des manques). Les résultats obtenus ont permis de confirmer une dégradation très nette des performances de la politique aléatoire lorsque des manques sont fréquents ou systématiques (cf. tableau 4).

4.2 Régulation des pics de production/consommation

4.2.1 Régulation à l'instant présent

Différents tests ont été réalisés afin de montrer que le modèle est en mesure de gérer dynamiquement les surproductions et surconsommations. Après la désactivation d'un producteur, des nouveaux contrats sont générés pour satisfaire les approvisionnements qui ont été interrompus.
Par exemple, le scénario 4 teste une surproduction en démarrant d'un état initial en dessous du seuil (cf. figure 15). Le régulateur choisit la politique qui stoppe en priorité les agents de wattage maximal. Un nouveau producteur de 1000 Watts (n°4) se greffe au nœud à posteriori, ce qui déclenche un dépassement de seuil : le régulateur stoppe alors le producteur 1 qui comporte le plus grand wattage. Par conséquent, les contrats des consommateurs 1,2 et 3 sont stoppés mais le producteur n°4 a un wattage suffisant pour approvisionner les consommateurs (cf. figure 16). Des nouveaux contrats sont alors régénérés dynamiquement avec le producteur 4. On obtient une durée maximale d'alerte de 4 secondes (cf. figure 17 et 18).

4.2.2 Régulation à priori en utilisant la prédiction

Des tests effectués sur plusieurs scénarios ont permis de montrer que le régulateur a la capacité de différer certaines activités lorsque l'agent d'apprentissage prédit un dépassement à t+60 minutes (surproduction ou surconsommation). Dans l'évaluation, on vérifie que la replanification prévoit bien un arrêt de l'activité en cours à t2 - 10 minutes (t2 étant l'heure prévue du dépassement de seuil) et que l'activité replanifiée couvre au moins 20% du seuil maximal (soit 600W). Par exemple sur un scénario simple, (cf. figure 19), 4 producteurs sont initialement intégrés au nœud : le premier pour une longue durée, les 3 suivants jusqu'à 23H51. Par la suite, l'agent d'apprentissage prédit un dépassement de seuil à 22H52. Le régulateur replanifie alors l'activité de certains producteurs (soit prod_2, prod_3 et prod_4) pour retirer au moins l'équivalent de 600 W. Dans cet exemple, il replanifie en priorité les producteurs de plus petit wattage. Sur l'état instantané du nœud ainsi que sur l'historique des évènements (cf. figure 20 et figure 21), les dates de fin de production sont modifiées (de 23H51 à 22H42, soit l'heure prédite – 10 minutes). Le contrat du consommateur 2 est impacté étant donné qu'il devait initialement s'arrêter au-delà de l'heure de dépassement de seuil.
NB : Le consommateur 4 qui est également approvisionné par prod_2 n'est pas impacté car son contrat devait initialement se terminer avant l'heure de l'arrêt.

4.3 Qualité de la prédiction

Pour évaluer la qualité de la prédiction, on exécute le modèle avec le scénario le plus réaliste (n°3) car il permet d'obtenir un état du nœud relativement stable dans

le temps car assez proches des moyennes statistiques de chaque tranche horaire. Le principe est de comparer l'état prédit à t1 avec l'état réel du nœud à l'heure cible de la prédiction (soit à t2 = t1 + horizon de la prédiction). Pour cela, l'agent d'apprentissage effectue un traitement périodique : il génère et enregistre dans une table les prédictions de l'état du nœud à différents horizons (5,10,30 et 60 minutes). Il enregistrement également dans une autre table l'état réel du nœud à chaque rafraîchissement. De cette manière on peut rapprocher la prédiction générée à t1 avec l'état réel à t2. Pour chaque prédiction, on obtient donc l'état de Markov supposé et l'état de Markov réel.
Les résultats sont récoltés pour différents horizons de manières à faire ressortir l'impact du nombre de pas sur la qualité du résultat.
Le tableau 5 fournit les taux de réussites moyens des prédictions par horizon et en utilisant une fenêtre d'apprentissage de 10 jours.

Composante	Horizon 5 min.	Horizon 10 min.	Horizon 30 min.	Horizon 60 min.
Wattage disponible	86,05%	93,90%	84,78%	57,35%
Wattage consommé	81,98%	68,29%	69,57%	69,12%
Wattage manquant	75,00%	67,07%	69,57%	72,06%
Wattage produit	86,05%	81,71%	76,09%	61,76%
Wattage demandé	80,81%	80,49%	60,87%	61,76%

TABLE 5: Taux de réussite moyens d'une centaine de prédictions par horizon, obtenus en utilisant une fenêtre de 10 jours

De manière générale, plus l'horizon est proche, plus la prédiction est fiable. En effet, le modèle étant stochastique, à chaque pas de temps, on ajoute de l'incertitude au résultat. Une difficulté importante a été rencontrée pour fiabiliser les données de prédiction : plusieurs scénarios différents ont été appliqués sur le même nœud, dont des tests techniques : de ce fait les mêmes données de prédiction (dont les nombres d'observations) ont été modifiés à partir de scénarios pour lesquels l'état du nœud est très éloigné d'une situation "réaliste". Il sera possible de remédier à cette situation en identifiant le scénario appliqué au niveau de la table des matrices de transition (chaque scénario a ses propres données dans la base).
Le tableau 7 permet de confirmer la dégradation de la qualité lorsque l'on utilise une fenêtre de 100 jours, c'est-à-dire des données plus anciennes. En effets, les premières données ont toutes été générées par des scénarios différents du scénario réaliste. Cela montre également l'intérêt de la fenêtre d'apprentissage qui permet de ne pas être impacté par les données "anciennes".

5 Discussion, conclusion et perspectives

5.1 Rappels des contributions

Le modèle proposé qui s'appuie sur l'auto-composition de services, permet à des agents autonomes de générer des contrats d'échange d'énergie électrique. Les contributions sont les suivantes :

- Modèle de coordination pour l'échange d'énergie et la régulation (LSA, agents, algorithmes de contrat)
- Régulation de l'énergie (consommation et production) de manière instantanée et pro-active
- Prédiction de production et consommation à l'aide de chaînes de Markov
- Outil de visualisation et de suivi de l'avancement du système
- Simulateur de données synthétiques de production et consommation d'énergie

5.2 Discussion des résultats de taux d'échec

Les différents résultats permettent de montrer que le choix de la politique de gestion de l'offre a un impact important. Une politique aléatoire "pure" peut entraîner de très mauvaises performances, en particulier lorsque les conditions deviennent plus difficiles pour générer un contrat. De manière générale, il faut donc l'éviter. Une politique hybride permet parfois d'obtenir de meilleurs résultats sans courir trop de risques mais d'une manière générale, le gain ne semble pas évident. Le choix de cette politique dépend du scénario et doit se faire en prenant en compte les conditions réelles dans lesquelles le modèle est exécuté.
En choisissant la politique optimale, on peut obtenir des performances correctes en termes de couverture des requêtes, avec des taux d'échec en base KWH inférieurs à 1 %. Les temps d'alerte maximums restent toujours supérieurs à 10 seconds. En effet, le mécanisme d'auto-composition peut s'avérer long :

- En raison du temps de propagation de l'information entre les agents. Chaque cycle d'exécution des lois de coordination est espacé d'une seconde, voire plus, si un problème de lenteur survient ou lorsque le nombre d'agents devient très grand.
- En raison du nombre d'étapes nécessaires pour aboutir à la validation d'un contrat. Le nombre d'étape peut s'avérer très grand si les premières offres reçues par le consommateur ne conviennent pas ou si la validation du contrat échoue.

5.3 Interprétation des résultats de prédiction

Les résultats montrent que l'on obtient une prédiction correcte sur un horizon très proche mais plutôt médiocre au-delà de 30 minutes. Le modèle d'apprentissage n'est pas encore arrivé à maturité et nécessite, d'une part des adaptations pour séparer les données d'apprentissage de chaque scénario utilisé, et d'autre part, de pouvoir emmagasiner un historique de données plus conséquent sur un même scénario. Une autre piste d'amélioration éventuelle pourrait consister à utiliser l'erreur de la prédiction pour corriger les données d'observation en appliquant le principe de descente de gradient. Sur ce modèle, la fenêtre d'apprentissage apporte l'avantage de pouvoir adapter le modèle plus facilement quand le scénario évolue dans le temps.

5.4 Autres types d'apprentissage

L'algorithme d'apprentissage des chaînes de Markov a été choisi car il a déjà été testé et éprouvé dans des modèles de coordinations ayant des mécanismes assez similaires et faisant intervenir des agents de service. Il est relativement facile à intégrer et a la capacité d'être glissant dans le temps.
Cependant, d'autres modèle tels que CRBM("Conditional Restricted Boltzmann Machine") ou l'algorithme de clusturing "D-Strem" à base de la densité ont également tété éprouvés sur des modèles faisant également intervenir des échanges d'énergie. Pour améliorer cette étude, il serait intéressant d'intégrer ces deux algorithmes dans la prédiction de l'état du nœud de manière à comparer leurs performances avec celle des chaînes de Markov. Il faudrait tester les 3 algorithmes sur le même scénario et comparer les taux de réussite obtenus selon les situations.

5.5 Pistes d'amélioration

Le modèle mis en place constitue un premier socle pour gérer des échanges d'énergie à partir d'agent autonomes. Il est capable d'obtenir des temps de réponse corrects (et ce

même dans des situations difficiles), capable de réguler la production/consommation d'énergie à l'instant t ou à priori en utilisant la prédiction à très court terme. En raison de l'hétérogénéité des données d'observations, le modèle d'apprentissage n'obtient pas encore des prédictions suffisamment fiables. Des ajustements sont cependant possibles pour améliorer la qualité des données.
Un certain nombre d'améliorations ont été identifiées afin de pouvoir mieux répondre aux différents besoins dans ce domaine :

- Le modèle se limite à un seul contrat par consommateur. Si on doit augmenter le wattage d'un consommateur en cours d'approvisionnement, on est obligé de stopper le contrat en cours et de relancer un nouveau cycle d'appel d'offres
- Le modèle ne génère pas de contrats pour un besoin ultérieur. Cela pourrait diminuer le nombre d'interruptions de consommation. Cependant, certains arrêts sont imprévus (dans quel cas, l'interruption du contrat est inévitable).
- Le modèle ne permet pas encore de gérer les échanges d'énergie entre voisins
- La tarification de l'énergie n'est pas encore gérée. Elle pourrait être utilisée dans la négociation de contrats entre voisins avec une politique de l'offre qui favorise la consommation pendant les heures creuses. Le prix serait déterminé en fonction de l'offre et la demande.
- L'application web affiche l'état courant du nœud sous forme de tableau. On pourrait imaginer une visualisation plus intuitive qui représenterait par exemple les agents d'énergie sous forme de rectangle plus ou moins grand suivant le wattage fourni, les catégories d'appareil et les types d'énergie sous forme icône et les flux d'approvisionnement sous forme de flèches plus ou moins large suivant le volume échangé etc...
- Au niveau du scénario de test réaliste, la granularité a été poussée au maximum, c'est à dire au niveau de chaque appareil : on a donc potentiellement un smart-meter par appareil. Cela permet de tester le modèle de coordination de manière complète mais cela ne correspond pas forcément aux configurations utilisées d'aujourd'hui. Le modèle permet cependant de définir un agent au niveau d'un agrégat de consommateurs ou producteurs en définissant une nouvelle catégorie.
- Le modèle de prédiction n'intègre pas encore les variables météorologiques
- Le modèle n'a pas encore été déployé sur un environnement réel d'appareils de maisons et de compteurs intelligents (smart-meter)

5.6 Perspectives

Le sujet des réseaux intelligents intéresse les collectivités et certaines communes sont prêtes à expérimenter ce type de solutions. De manière plus générale, les réseaux intelligents ont acquis de la notoriété : ils apportent des gains importants à tous les niveaux, du particulier aux collectivités. Ils favorisent le suivi de la consommation, de la production et de la distribution. Ils contribuent à diminuer les coûts en énergie, à réduire les pics et à décentraliser la distribution en favorisant les échanges entre particuliers.

Références

[1] Jean-Pierre Banâtre et Daniel Le Métayer. "The gamma model and its discipline of programming". In : *Science of Computer Programming* 15.1 (1990), p. 55-77. issn : 0167-6423. doi : https://doi.org/10.1016/0167-6423(90)90044-E. url : https://www.sciencedirect.com/science/article/pii/016764239090044E.

[2] J. L. Fernandez-Marquez, G. Serugendo et Sara Montagna. "BIO-CORE : Bio-inspired Self-organising Mechanisms Core". In : *BIONETICS*. 2011.

[3] Daniel Godoy-Shimizu, Jason Palmer et Nicola Terry. "What can we learn from the household electricity survey ?" In : *Buildings* 4.4 (2014), p. 737-761.

[4] P.-P. Grasse. "La reconstruction du nid et les coordinations interindividuelles chez-Bellicositermes natalensis etCubitermes sp. la théorie de la stigmergie : Essai d'interprétation du comportement des termites constructeurs". In : *Insectes Sociaux* 6 (2005), p. 41-80.

[5] Houssem Ben Mahfoudh. "Learning-based coordination model for spontaneous". Thèse de doct. Université de Genève, 2020.

[6] Farhan H Malik et Matti Lehtonen. "A review : Agents in smart grids". In : *Electric Power Systems Research* 131 (2016), p. 71-79.

[7] Marco Mamei et Franco Zambonelli. *Field-based coordination for pervasive multiagent systems*. Springer Science & Business Media, 2006.

[8] Marco Mamei, Franco Zambonelli et Letizia Leonardi. "Co-fields : Towards a unifying approach to the engineering of swarm intelligent systems". In : *International Workshop on Engineering Societies in the Agents World*. Springer. 2002, p. 68-81.

[9] Akanksha Maurya et al. "Time-series clustering for data analysis in Smart Grid". In : *2016 IEEE International Conference on Smart Grid Communications (SmartGridComm)*. 2016, p. 606-611. doi : 10.1109/SmartGridComm.2016.7778828.

[10] Sara Montagna et al. "Towards a Comprehensive Approach to Spontaneous self-composition in Pervasive Ecosystems." In : *WOA*. Citeseer. 2012.

[11] Mountassir FOUAD et Reda Mali et Mohammed Bousmah. "Machine Learning au service de la prédiction de la demande d'énergie dans les Smart Grid". In : *Revue Méditerranéenne des Télécommunications* 8.2 (2018). url : https://revues.imist.ma/index.php/RMT/article/view/12834.

[12] Karen Rose, Scott Eldridge et Lyman Chapin. "The internet of things : An overview". In : *The internet society (ISOC)* 80 (2015), p. 1-50.

[13] David Menga Sylvain Frey Ada Diaconescu et Isabelle Demeure. "A Generic Holonic Control Architecture for Heterogeneous Multi-Scale and Multi-Objective Smart Micro-Grids." In : *ACM Trans. Autonom.* 2014. doi : http://dx.doi.org/10.1145/0000000.0000000.

[14] C. F. Tschudin T. Meyer L. Yamamoto. "An artificial chemistry for networking". In : *First Workshop on Bio-Inspired Design of Networks* (2007), p. 45-57.

[15] Surafel Luleseged Tilahun et Giovanna Di Marzo Serugendo. "Cooperative multiagent system for parking availability prediction based on time varying dynamic Markov chains". In : *Journal of Advanced Transportation* 2017 (2017).

[16] Mirko Viroli et al. "Spatial coordination of pervasive services through chemical-inspired tuple spaces". In : *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 6.2 (2011), p. 1-24.

[17] Franco Zambonelli et al. "Developing pervasive multi-agent systems with nature-inspired coordination". In : *Pervasive and Mobile Computing* 17 (2015). 10 years of Pervasive Computing' In Honor of Chatschik Bisdikian, p. 236-252. issn : 1574-1192. doi : https://doi.org/10.1016/j.pmcj.2014.12.002. url : https://www.sciencedirect.com/science/article/pii/S1574119214001904.

[18] Reihaneh Haji Mahdizadeh Zargar et Mohammad Hossein Yaghmaee Moghaddam. "Development of a Markov-Chain-Based Solar Generation Model for Smart Microgrid Energy Management System". In : *IEEE Transactions on Sustainable Energy* 11.2 (2020), p. 736-745. doi : 10.1109/TSTE.2019.2904436.

Annexe

.1 wattage disponible

$$\begin{aligned} \text{w\_disponible} = \text{w\_produit} &- \sum \text{w\_conso\_en\_cours} \\ &- \sum \text{w\_offres\_en\_attente} \\ &- \sum \text{w\_contrat\_en\_attente\_validation} \end{aligned}$$

(wattage disponible au niveau d'un nœud)

.2 taux d'échec

$$\text{tx\_echec\_base\_temps} = \frac{\sum_{historique:WattageEnAlerte>0}(t_i - t_{(i-1)})}{\sum_{historique}(t_i - t_{(i-1)})} \qquad \text{(base temps)}$$

$$\text{tx\_echec\_base\_kwh} = \frac{\sum_{historique}(t_i - t_{(i-1)}) * \text{WattageEnAlerte}}{\sum_{historique}(t_i - t_{(i-1)}) * \text{TotalWattageProduit}} \qquad \text{(base kwh)}$$

.3 Sélection des producteurs à stopper

Algorithm 1 Sélection des producteurs à stopper

```
listProducersToStop ← []
while wProduced - sum_w(listProducersToStop)  > MAX_W_PRODUCED do
    nextProducer ← listProducers.remove(0)
    listProducersToStop− > add(nextProducer)
end while
```

.4 Etats de Markov

État	Plage (en Watt)	Commentaire
1	0	On considère la valeur nulle comme état à part entière
2	]0,600[	1ère tranche de 20 % du seuil d'alerte
3	[600,1200[	2e tranche de 20 %
4	[1200,1800[	3e tranche de 20 %
5	[1800,2400[	4e tranche de 20 %
6	[2400,3000[	5e tranche de 20 % : on considère la valeur de 3000 Watt comme le seuil d'alerte (surproduction ou surconsommation)
7	[3000, $+\infty$ [	Plage au-delà du seuil d'alerte

TABLE 6: Définition des états de Markov pour le modèle de prédiction du nœud.

.5 Évaluation des prédictions avec une fenêtre de 100 jours

Composante	Horizon 5 min.	Horizon 10 min.	Horizon 30 min.	Horizon 60 min.
Wattage disponible	63,39%	59,52%	58,04%	55,81%
Wattage consommé	80,65%	79,76%	70,54%	64,24%
Wattage manque	69,94%	66,96%	65,48%	60,17%
Wattage produit	80,65%	74,11%	66,37%	61,63%
Wattage demandé	80,65%	72,02%	62,20%	61,34%

TABLE 7: Taux de réussite moyens d'une centaine de prédictions par horizon, obtenus en utilisant une fenêtre de 100 jours

.6 Modèles d'entités

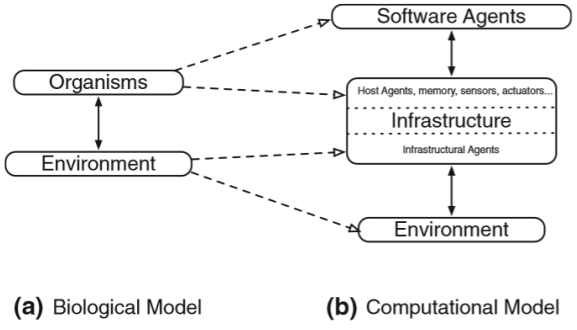


FIGURE 6: Les deux modèles d'entités : biologique et informatique [2]

.7 Données échangées entre les agents

Agent émetteur	Agent récepteur	Etiquette	Classe	Synopsis
Consommateur	Producteur	REQ	EnergyRequest	demande d'approvisionnement en énergie
Consommateur	Producteur	SATISFIED	String ('1')	confirmation de satisfaction
Consommateur	Agent contrat	NEW_CTR_#id	ProtectedContract	nouveau contrat a valider
Consommateur	Agent contrat	UPDATE_CTR #id	ProtectedContract	contrat a modifier
Consommateur	Agent contrat	STOP_CTR_#id	RegulationWarning	alerte a l'origine de la désactivation de l'agent
Producteur	Consommateur	OFFER	SingleOffer	offre d'approvisionnement en énergie envoyée à un consommateur
Producteur	Agent contrat	PROD_ CONFIRM	ProtectedConrmation Table	confirmations émises par le producteur
Agent contrat	Producteur	CONTRACT	ProtectedContract	mise à jour d'un contrat d'approvisionnement
Consommateur Producteur Agent contrat	agent d'apprentissage	EVENT	Event	évènement survenu (démarrage, désactivation, arrêt)
Agent d'apprentissage	Agent régulateur	PRED	PredictionForm	prédiction obtenue a horizon d'une heure (produit, demande, consomme, manque, disponible)
Agent régulateur	Consommateur Producteur Agent contrat	WARNING	ProtectedContract	Ordre de régulation envoyé à un agent (interruption, désactivation, modification forcée)
Agent régulateur	Consommateur Producteur Agent contrat	RESCHEDULE	RescheduleTable	Table de replanification d'activités (clé : agent, valeur : replanification

FIGURE 7: Liste des données échangées entre les agents

.8 Protection des données

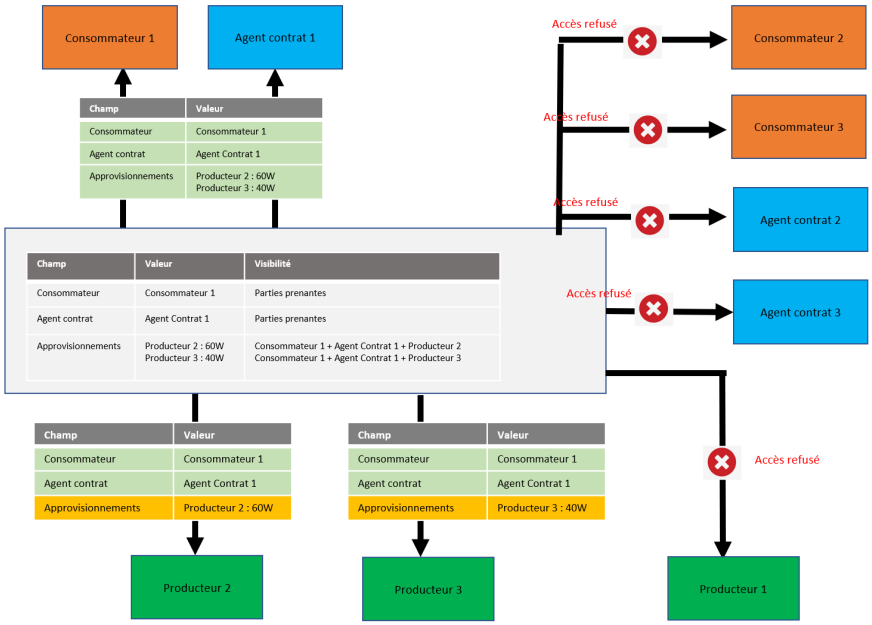


FIGURE 8: Protection des données de contrat

.9 Architecture logicielle

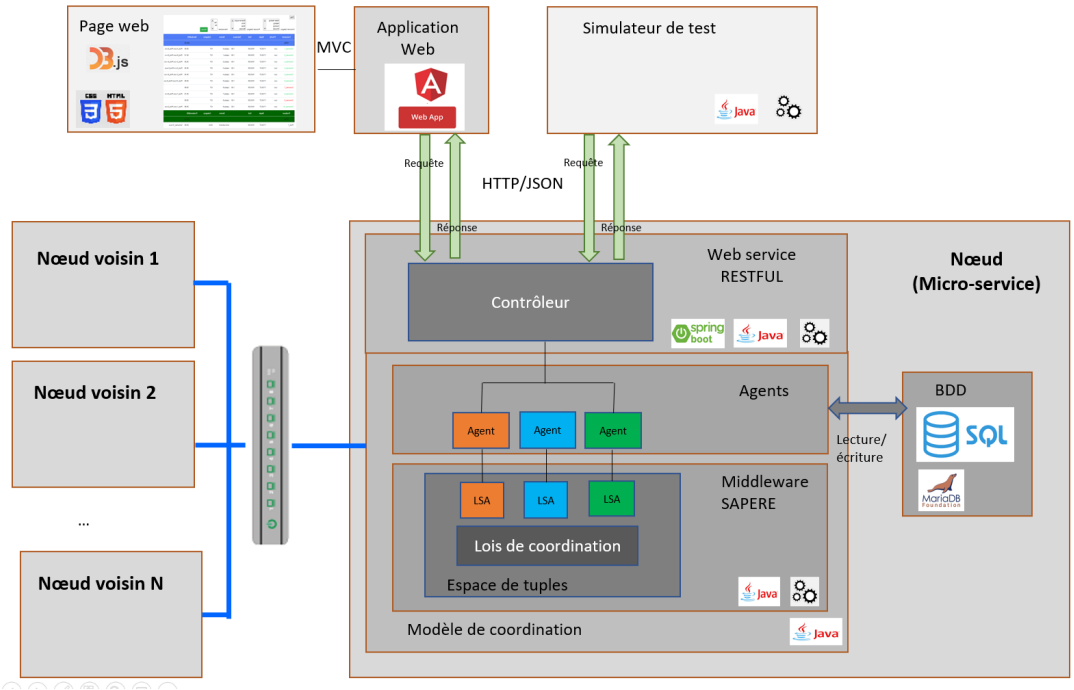


FIGURE 9: Architecture logicielle

.10 Modèle conceptuel de données

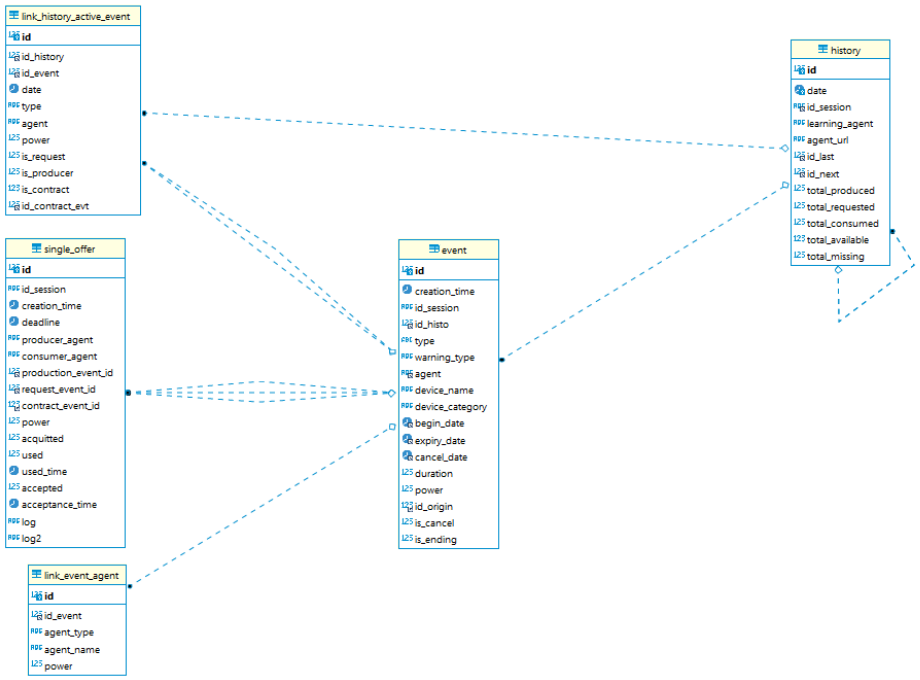


FIGURE 10: Relations entre les tables : historique des échanges d'énergie

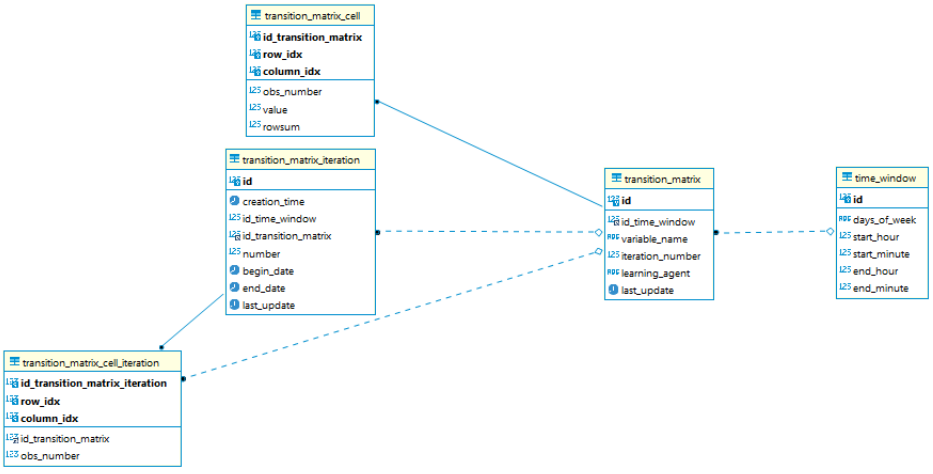


FIGURE 11: Relations entre les tables : données d'apprentissage

.11 Visualisation de l'historique d'un nœud

Time	Evt type	Agent	Begin	End	Power(W)	Requested(W)	Produced(W)	Consumed(W)	Available(W)	Missing(W)	Unsatisfied consumers	Warnings unsatisfied	(Min)
09:33:59	PRODUCTION	Prod_1	09:33:59		31.00		31.00		31.00				
09:34:00	PRODUCTION	Prod_2	09:34:00	10:33:59	18.00		49.00		49.00				
09:34:01	REQUEST	Consumer_1	09:34:01		23.10	23.10	49.00		49.00	23.10	Consumer_1(23.10)	Consumer_1(23.10):0	23.10
09:34:02	REQUEST	Consumer_2	09:34:02	09:35:02	297.21	320.31	49.00		49.00	320.31	Consumer_1(23.10), Consumer_2(297.21)	Consumer_1(23.10):1	23.10
09:34:04	REQUEST	Consumer_3	09:34:04	09:48:04	452.38	772.69	49.00		49.00	772.69	Consumer_1(23.10), Consumer_2(297.21), Consumer_3(452.38)	Consumer_1(23.10):3	23.10
09:34:05	CONTRACT	Contract_1	09:34:05		23.10	772.69	49.00	23.10	25.90	749.59	Consumer_2(297.21), Consumer_3(452.38)		297.21
09:34:06	PRODUCTION	Prod_3	09:34:06	10:08:06	280.85	772.69	329.85	23.10	306.75	749.59	Consumer_2(297.21), Consumer_3(452.38)	Consumer_2(297.21):0	297.21
09:34:09	PRODUCTION	Prod_4	09:34:09	09:48:09	7.61	772.69	337.46	23.10	314.36	749.59	Consumer_2(297.21), Consumer_3(452.38)	Consumer_2(297.21):3	297.21
09:34:10	CONTRACT	Contract_2	09:34:10	09:35:02	297.21	772.69	337.46	320.31	17.15	452.38	Consumer_3(452.38)		452.38
09:34:11	PRODUCTION	Prod_5	09:34:11	09:56:11	156.72	772.69	494.18	320.31	173.87	452.38	Consumer_3(452.38)		452.38
09:34:14	REQUEST	Consumer_4	09:34:14	10:11:14	214.10	986.79	494.18	320.31	173.87	666.48	Consumer_3(452.38), Consumer_4(214.10)		214.10
09:34:16	REQUEST	Consumer_5	09:34:16	10:26:16	14.11	1000.90	494.18	320.31	173.87	680.59	Consumer_3(452.38), Consumer_4(214.10), Consumer_5(14.11)	Consumer_5(14.11):0	14.11
09:34:18	REQUEST	Consumer_6	09:34:18	10:05:18	208.45	1209.35	494.18	320.31	173.87	889.04	Consumer_3(452.38), Consumer_4(214.10), Consumer_5(14.11), Consumer_6(208.45)	Consumer_5(14.11):2	14.11
09:34:22	CONTRACT	Contract_5	09:34:22	09:48:09	14.11	1209.35	494.18	334.42	159.76	874.93	Consumer_3(452.38),		208.45

FIGURE 12: Historique d'un nœud

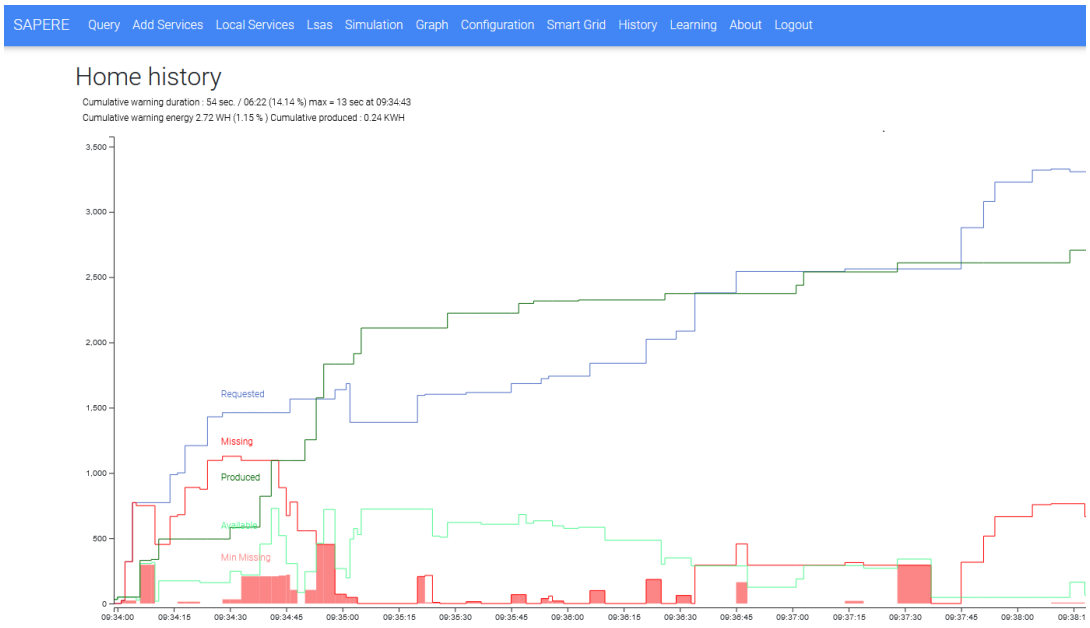


FIGURE 13: Historique d'un nœud sous forme de graphe

.12 Visualisation des matrices de transition

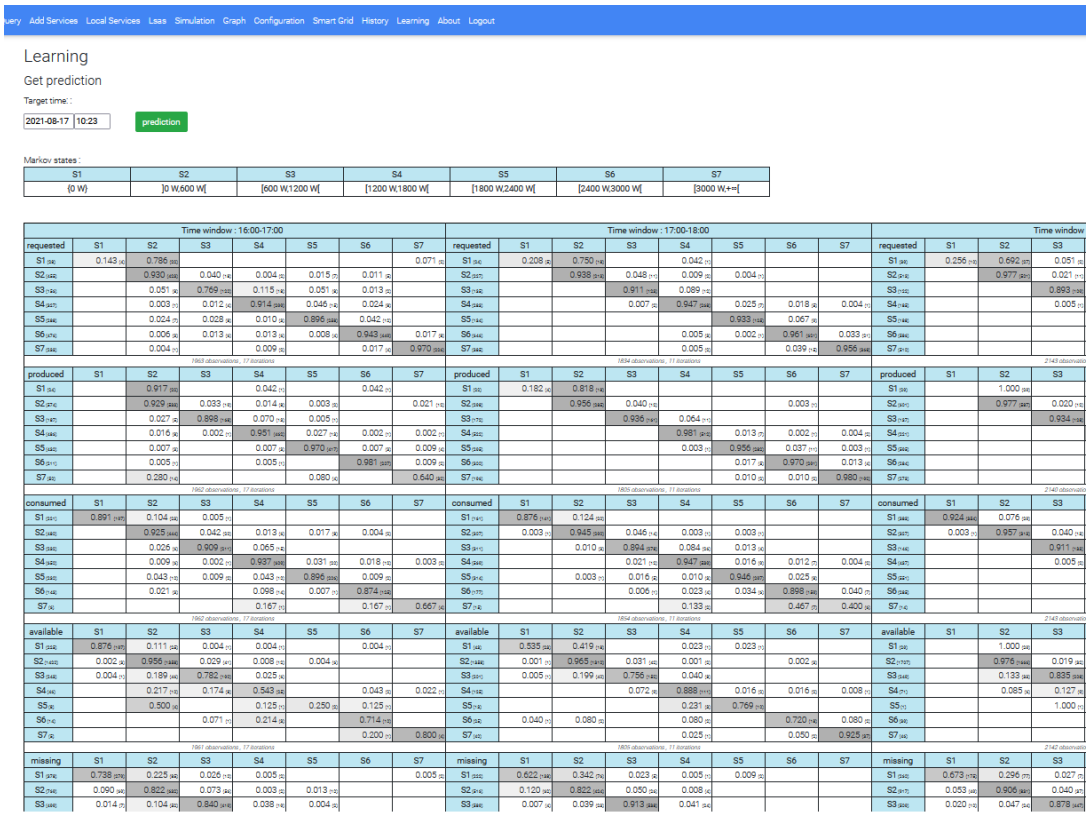


FIGURE 14: Matrices de transition des chaînes de Markov

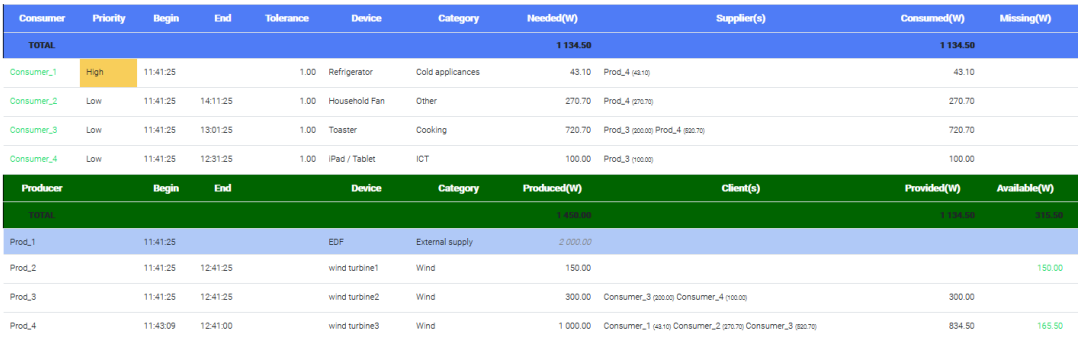
.13 Scénario de surproduction

Filter

Consumer	Priority	Begin	End	Tolerance	Device	Category	Needed(W)	Supplier(s)	Consumed(W)	Missing(W)
TOTAL							1 134.50		1 134.50	
Consumer_1	High	11:41:25		1.00	Refrigerator	Cold applicances	43.10	Prod_1 (43.10)	43.10	
Consumer_2	Low	11:41:25	14:11:25	1.00	Household Fan	Other	270.70	Prod_1 (270.70)	270.70	
Consumer_3	Low	11:41:25	13:01:25	1.00	Toaster	Cooking	720.70	Prod_1 (720.70)	720.70	
Consumer_4	Low	11:41:25	12:31:25	1.00	IPad / Tablet	ICT	100.00	Prod_3 (100.00)	100.00	

Producer	Begin	End	Device	Category	Produced(W)	Client(s)	Provided(W)	Available(W)
TOTAL					3 450.00		1 134.50	1 315.50
Prod_1	11:41:25		EDF	External supply	2 000.00	Consumer_1 (43.10) Consumer_2 (270.70) Consumer_3 (720.70)	1 034.50	965.50
Prod_2	11:41:25	12:41:25	wind turbine1	Wind	150.00			150.00
Prod_3	11:41:25	12:41:25	wind turbine2	Wind	300.00	Consumer_4 (100.00)	100.00	200.00

FIGURE 15: Surproduction : état initial



Consumer	Priority	Begin	End	Tolerance	Device	Category	Needed(W)	Supplier(s)	Consumed(W)	Missing(W)
TOTAL							1 134.50		1 134.50	
Consumer_1	High	11:41:25		1.00	Refrigerator	Cold applicances	43.10	Prod_4 (43.10)	43.10	
Consumer_2	Low	11:41:25	14:11:25	1.00	Household Fan	Other	270.70	Prod_4 (270.70)	270.70	
Consumer_3	Low	11:41:25	13:01:25	1.00	Toaster	Cooking	720.70	Prod_3 (200.00) Prod_4 (520.70)	720.70	
Consumer_4	Low	11:41:25	12:31:25	1.00	IPad / Tablet	ICT	100.00	Prod_3 (100.00)	100.00	

Producer	Begin	End	Device	Category	Produced(W)	Client(s)	Provided(W)	Available(W)
TOTAL					3 450.00		1 134.50	315.50
Prod_1	11:41:25		EDF	External supply	*2 000.00*			
Prod_2	11:41:25	12:41:25	wind turbine1	Wind	150.00			150.00
Prod_3	11:41:25	12:41:25	wind turbine2	Wind	300.00	Consumer_3 (200.00) Consumer_4 (100.00)	300.00	
Prod_4	11:43:09	12:41:00	wind turbine3	Wind	1 000.00	Consumer_1 (43.10) Consumer_2 (270.70) Consumer_3 (520.70)	834.50	165.50

FIGURE 16: Surproduction : état final

Time	Evt type	Agent	Begin	End	Power(W)	Requested(W)	Produced(W)	Consumed(W)	Available(W)	Missing(W)	Unsatisfied consumers	Warnings unsatisfied	(Min)
11:41:25	PRODUCTION	Prod_1	11:41:25		2000.00	1134.50	2450.00		2450.00	1134.50	Consumer_1(43.10), Consumer_2(270.70), Consumer_3(720.70), Consumer_4(100.00)	Consumer_1(43.10):0, Consumer_2(270.70):0, Consumer_3(720.70):0, Consumer_4(100.00):0	43.10
	PRODUCTION	Prod_2	11:41:25	12:41:25	150.00								
	PRODUCTION	Prod_3	11:41:25	12:41:25	300.00								
	REQUEST	Consumer_1	11:41:25		43.10								
	REQUEST	Consumer_2	11:41:25	14:11:25	270.70								
	REQUEST	Consumer_3	11:41:25	13:01:25	720.70								
	REQUEST	Consumer_4	11:41:25	12:31:25	100.00								
11:41:29	CONTRACT	Contract_1	11:41:29		43.10	1134.50	2450.00	1134.50	1315.50				
	CONTRACT	Contract_2	11:41:29	14:11:25	270.70								
	CONTRACT	Contract_3	11:41:29	13:01:25	720.70								
	CONTRACT	Contract_4	11:41:29	12:31:25	100.00								
11:42:26	Refresh					1134.50	2450.00	1134.50	1315.50				
11:43:09	PRODUCTION	Prod_4	11:43:09	12:41:00	1000.00	1134.50	3450.00	1134.50	2315.50				
11:43:11	PRODUCTION_STOP (OVER_PRODUCTION)	Prod_1	11:43:11	11:43:11	2000.00	1134.50	1450.00	100.00	1350.00	1034.50	Consumer_1(43.10), Consumer_2(270.70), Consumer_3(720.70)	Consumer_1(43.10):0, Consumer_2(270.70):0, Consumer_3(720.70):0	43.10
	CONTRACT_STOP (OVER_PRODUCTION)	Contract_1	11:43:11	11:43:11	43.10								
	CONTRACT_STOP (OVER_PRODUCTION)	Contract_2	11:43:11	11:43:11	270.70								
	CONTRACT_STOP (OVER_PRODUCTION)	Contract_3	11:43:11	11:43:11	720.70								
11:43:15	CONTRACT	Contract_1	11:43:15	12:41:00	43.10	1134.50	1450.00	413.80	1036.20	720.70	Consumer_3(720.70)	Consumer_3(720.70):4	720.70
	CONTRACT	Contract_2	11:43:15	12:41:00	270.70								
11:43:18	CONTRACT	Contract_3	11:43:18	12:41:00	720.70	1134.50	1450.00	1134.50	315.50				
11:43:27	Refresh					1134.50	1450.00	1134.50	315.50				

FIGURE 17: Surproduction :historique

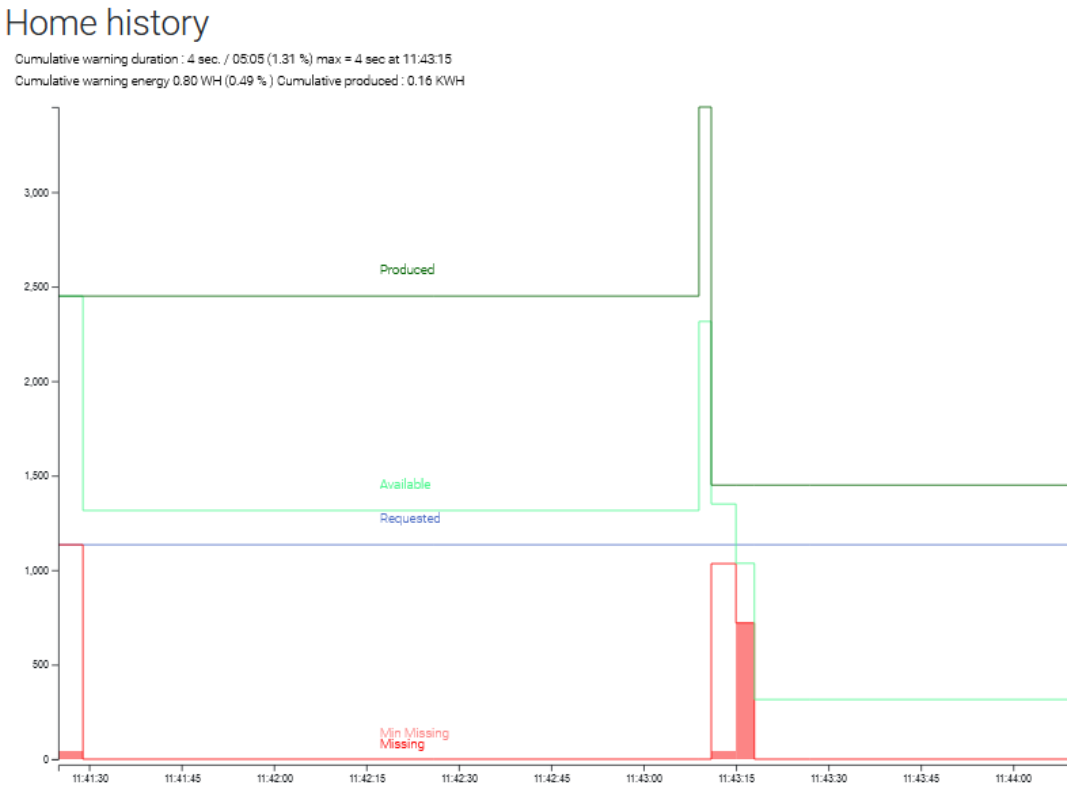


FIGURE 18: Surproduction : historique sous forme de graphe

Consumer	Priority	Begin	End	Tolerance	Device	Category	Needed(W)	Supplier(s)	Consumed(W)	Missing(W)
TOTAL							1 134.50		1 134.50	
Consumer_1	High	21:51:19		1.00	Refrigerator	Cold appliances	43.10	Prod_1 (43.10)	43.10	
Consumer_2	Low	21:51:19		1.00	Household Fan	Other	270.70	Prod_1 (50.00) Prod_2 (6.90) Prod_3 (156.90) Prod_4 (56.90)	270.70	
Consumer_3	Low	21:51:19	23:11:19	1.00	Toaster	Cooking	720.70	Prod_1 (720.70)	720.70	
Consumer_4	Low	21:51:20	22:41:19	1.00	iPad / Tablet	ICT	100.00	Prod_2 (100.00)	100.00	

Producer	Begin	End	Device	Category	Produced(W)	Client(s)	Provided(W)	Available(W)
TOTAL					2 650.00		1 134.50	1 515.50
Prod_1	21:51:19		EDF	External supply	2 000.00	Consumer_1 (43.10) Consumer_2 (50.00) Consumer_3 (720.70)	813.80	1 186.20
Prod_2	21:51:19	23:51:19	wind turbine1	Wind	150.00	Consumer_2 (6.90) Consumer_4 (100.00)	106.90	43.10
Prod_3	21:51:19	23:51:19	wind turbine2	Wind	300.00	Consumer_2 (156.90)	156.90	143.10
Prod_4	21:51:19	23:51:19	wind turbine3	Wind	200.00	Consumer_2 (56.90)	56.90	143.10

FIGURE 19: Surproduction : état initial

Consumer	Priority	Begin	End	Tolerance	Device	Category	Needed(W)	Supplier(s)	Consumed(W)	Missing(W)
TOTAL							1 134.50		1 134.50	
Consumer_1	High	21:51:19		1.00	Refrigerator	Cold appliances	43.10	Prod_1 (43.10)	43.10	
Consumer_2	Low	21:51:19		1.00	Household Fan	Other	270.70	Prod_1 (50.00) Prod_2 (6.90) Prod_3 (156.90) Prod_4 (56.90)	270.70	
Consumer_3	Low	21:51:19	23:11:19	1.00	Toaster	Cooking	720.70	Prod_1 (720.70)	720.70	
Consumer_4	Low	21:51:20	22:41:19	1.00	iPad / Tablet	ICT	100.00	Prod_2 (100.00)	100.00	

Producer	Begin	End	Device	Category	Produced(W)	Client(s)	Provided(W)	Available(W)
TOTAL					2 650.00		1 134.50	1 515.50
Prod_1	21:51:19		EDF	External supply	2 000.00	Consumer_1 (43.10) Consumer_2 (50.00) Consumer_3 (720.70)	813.80	1 186.20
Prod_2	21:53:39	22:42:59	wind turbine1	Wind	150.00	Consumer_2 (6.90) Consumer_4 (100.00)	106.90	43.10
Prod_3	21:53:39	22:42:59	wind turbine2	Wind	300.00	Consumer_2 (156.90)	156.90	143.10
Prod_4	21:53:39	22:42:59	wind turbine3	Wind	200.00	Consumer_2 (56.90)	56.90	143.10

FIGURE 20: Surproduction : état final

Time	Evt type	Agent	Begin	End	Power(W)	Requested(W)	Produced(W)	Consumed(W)	Available(W)	Missing(W)	consumers	Warnings unsatisfied	(Min)
21:51:19	PRODUCTION	Prod_1	21:51:19		2000.00	1034.50	2650.00		2650.00	1034.50	Consumer_1(43.10), Consumer_2(270.70), Consumer_3(720.70)	Consumer_1(43.10):0, Consumer_2(270.70):0, Consumer_3(720.70):0	43.10
	PRODUCTION	Prod_2	21:51:19	23:51:19	150.00								
	PRODUCTION	Prod_3	21:51:19	23:51:19	300.00								
	PRODUCTION	Prod_4	21:51:19	23:51:19	200.00								
	REQUEST	Consumer_1	21:51:19		43.10								
	REQUEST	Consumer_2	21:51:19		270.70								
	REQUEST	Consumer_3	21:51:19	23:11:19	720.70								
21:51:20	REQUEST	Consumer_4	21:51:20	22:41:19	100.00	1134.50	2650.00		2650.00	1134.50	Consumer_1(43.10), Consumer_2(270.70), Consumer_3(720.70), Consumer_4(100.00)	Consumer_1(43.10):1, Consumer_2(270.70):1, Consumer_3(720.70):1, Consumer_4(100.00):0	43.10
21:52:06	CONTRACT	Contract_1	21:52:06		43.10	1134.50	2650.00	1134.50	1515.50				
	CONTRACT	Contract_2	21:52:06	23:51:19	270.70								
	CONTRACT	Contract_3	21:52:06	23:11:19	720.70								
	CONTRACT	Contract_4	21:52:06	22:41:19	100.00								
21:52:59	Refresh					1134.50	2650.00	1134.50	1515.50				
21:53:39	PRODUCTION_UPDATE (OVER_PRODUCTION_FORCAST)	Prod_2	21:53:39	22:42:59	150.00	1134.50	2650.00	1134.50	1515.50				
	PRODUCTION_UPDATE (OVER_PRODUCTION_FORCAST)	Prod_3	21:53:39	22:42:59	300.00								
	PRODUCTION_UPDATE (OVER_PRODUCTION_FORCAST)	Prod_4	21:53:39	22:42:59	200.00								
	CONTRACT_UPDATE (OVER_PRODUCTION_FORCAST)	Contract_2	21:53:39	22:42:59	270.70								

FIGURE 21: Surproduction :historique