Modélisation de la disponibilité d'un système énergétique décentralisé par automates stochastiques modulaires

Gilles Deleuze, Tesnim Abdellatif, Nicolae Brînzei, Jean-François Pétin, Gaël Hequet, Vincent Martin

Institut pour la Maîtrise des Risques
Sûreté de Fonctionnement - Management - Cindyniques

Modélisation de la disponibilité d'un système énergétique décentralisé par automates stochastiques modulaires

Dependability model of a decentralized energy system using stochastic automata

Gilles Deleuze
Tesnim Abdellatif
EDF R&D
91140 Palaiseau, France
gilles.deleuze@edf.fr
tesnim.abdellatif@edf.fr

Nicolae Brînzei
Jean-François Pétin
Université de Lorraine, CNRS, CRAN
54000 Nancy, France
nicolae.brinzei@univ-lorraine.fr

Gaël Hequet
Vincent Martin
Université de Lorraine, ENSEM
54505 Vandœuvre-lès-Nancy Cedex, France

***Résumé*—De nombreux entrepreneurs et experts en énergie considèrent la technologie Blockchain comme un moyen efficace pour supporter des places de marché décentralisées. Dans cette étude de faisabilité de la R&D d'EDF, un système « socio cyber physique » à base de Blockchain est modélisé afin d'en étudier le fonctionnement et la disponibilité.**

***Summary*—Entrepreneurs and energy experts consider Blockchain technology as an effective solution to support decentralized marketplaces. In this feasibility study by EDF R&D, a Blockchain-based "socio-cyber physical" system is modelized in order to study its operation and availability.**

Mots-clés : Automates Stochastiques, Blockchains, Défaillances de Cause Commune, Modèle d'Atwood, Défaillances Byzantines, Microgrids

I. INTRODUCTION

L'émergence des productions électriques décentralisées et d'un parc de véhicules électriques significatif est un défi pour le système électrique. Des évolutions réglementaires accélèrent la recherche de nouveaux modèles économiques autour de la notion de « place de marché d'énergie». Une approche classique consiste à utiliser un système d'information centralisé géré par une entité unique. En parallèle, chercheurs et entrepreneurs développent des systèmes d'information décentralisés à base de registres distribués de type Blockchain. Ils sont décrits comme particulièrement attractifs pour des places de marché avec des microtransactions nombreuses ou des besoins de résilience.

II. DÉFINITIONS ET CONTEXTE

Depuis 2016, on observe un engouement autour de la technologie Blockchain [1]. Les Blockchains sont des registres distribués dans un réseau pair à pair (Distributed Ledger Technologies, DLT), structurés en blocs de données et horodatés. Elles sont le fondement technologique des cryptomonnaies, en particulier du Bitcoin [2]. Des protocoles de consensus assurent l'unicité et l'intégrité du registre, en assurant que toutes les transactions sont enregistrées dans le même ordre sur toutes les machines du réseau. Il y a de nombreux types d'implémentation de Blockchain, selon les consensus employés: Proof of Work (PoW), Proof of Stake (PoS), Proof of Authority (PoA), et de nombreuses variantes de protocoles BFT (Byzantine Fault Tolerant) [43].

La technologie Ethereum [3] apporte la possibilité de coder des scripts (dits smart contracts) exécutables par la Blockchain, qui devient une machine virtuelle distribuée permettant d'exécuter du code complet au sens de [4]. Avec Ethereum et ses variantes, comme d'autres technologies comme Hyperledger, le champ d'application de cette technologie est devenu très large. Le fonctionnement décentralisé d'une Blockchain et sa capacité à faire fonctionner des systèmes monétaires complètement digitalisés lui confère une grande attractivité pour le développement de systèmes cyberphysiques associés à des systèmes économiques ouverts. Il s'agit typiquement de créer un environnement de commerce et de facturation fiable sans autorité centralisée [5]. Les principaux avantages attendus par rapport à une solution impliquant un tiers de confiance technique et «organisationnel» centralisé sont la réduction des coûts et une meilleure résilience.

Le système étudié dans cet article est une place de marché d'énergie, désignée ici par le terme Local Energy

Marketplace (LEM) [6]. Typiquement, une LEM est associée à un réseau électrique de petite taille (microgrid), indépendant, ou intégré à un réseau global via une connexion par un poste basse tension. Des utilisateurs du microgrid possèdent ou partagent un moyen de production d'énergie local éventuellement avec stockage. Du point de vue système d'information, une LEM à base de Blockchain est une plateforme pair à pair par laquelle les utilisateurs du réseau se vendent et s'achètent des valeurs correspondant à des quantités énergie. Du point de vue sociétal, une LEM est une organisation formalisée avec des objectifs et des règles dépendant du contexte local. Ainsi en France, les LEM sont régies par la loi relative à l'autoconsommation collective [7] dans le but de développer la production décentralisée d'énergie renouvelable [8]. Le mouvement est très progressif, fin 2019, en France, les installations d'autoconsommation collective (ACC) représentent une poignée de cas sur les 46000 installations d'autoconsommation Photovoltaïques (PV).

Pour tracer les échanges économiques d'énergie entre ses utilisateurs, la LEM est gérée par des smart contracts qui calculent les prix (achat et vente, …) et effectuent les paiements de manière sécurisée en suivant un algorithme d'allocation prédéfini. [9] expose les avantages des LEM avec emploi d'une Blockchain en support. Selon [10], elle permettrait à des consommateurs d'acheter leur électricité directement auprès de producteurs indépendants d'énergies renouvelables. [11] présente cette technologie comme une solution aux défis générés par la généralisation de l'autoproduction, et l'utilisation massive de voitures électriques. Dans d'autres contextes, l'objectif est d'améliorer la résilience du système énergétique [12].

La Blockchain est considérée comme prometteuse pour simplifier les systèmes de comptage et de facturation dans l'énergie [13], faciliter le commerce d'énergie en pair à pair. Associée à des cryptomonnaies, elle peut également inciter les utilisateurs à favoriser des moyens de production décarbonés ou locaux [14]. L'idée générale est d'utiliser des Unités de Compte (des Tokens) techniquement identiques à des cryptomonnaies qui permettent de comptabiliser les échanges d'énergie. Les Tokens incitent les utilisateurs à adopter des modes d'usages de l'énergie sans passer par des systèmes de tarification habituellement proposés par les services publics ou les agrégateurs. Les Tokens peuvent aussi avoir une fonction de certificats numériques et simplifier la mise en œuvre de systèmes de comptabilité décarbonée parfois extrêmement complexes (tels que les Low Carbon Fuel Standard Program, USA).

Son rôle s'étend à la gestion de la flexibilité au niveau de la couche physique de la LEM avec l'emploi de batteries [15], [16] [17] ou de véhicules électriques [18]. [19] propose la gestion d'un système industriel par une LEM entre machines.

Fin 2019, des LEM sont en cours de développement ou de test, avec ou sans Blockchains, aux États-Unis [15], au Royaume-Uni projets [20] et CommUNITY de EDF Energy à Brixton [21], en Allemagne [22], à Perth en Australie [14], à Alès en France [24], et Perpignan [25]. La Blockchain est employée à Brooklyn, Perth, Brixton, Perpignan.[25,Dans ces expérimentations, la Blockchain ne gère pas *directement* le flux d'énergie réel dans le réseau. Mais elle gère la LEM du point de vue comptable : la transaction est faite après le transport réel de l'énergie. Elle oriente de façon *indirecte* le comportement des utilisateurs et donc le fonctionnement du réseau. Les risques ont donc d'abord des conséquences économiques et réglementaires (erreurs comptables, non conformités..). Mais si ces systèmes se déploient en nombre, les comportements d'utilisateurs induits en erreur par des LEM défaillantes ou mal conçues, pourraient avoir des conséquences physiques sur le réseau électrique global.

III. ETAT DE L'ART

Au-delà d'expérimentations locales et de petite taille, la faisabilité technique, l'évolutivité et l'impact économique à grande échelle n'ont pas encore été évalués à partir de données réalistes. En particulier, pour déployer à une échelle significative de tels systèmes, des études de disponibilité et de sécurité sont à faire, pour montrer qu'ils sont au niveau actuel des solutions sans architecture distribuée. De plus, il y a un manque de protocoles standard pour l'évaluation technique de performances d'une Blockchain au sein d'un système cyberphysique.

Des travaux ont été publiés sur la modélisation des Blockchains en tant que système informatique, et sur la vérification formelle du code des smart contracts. En particulier, [26] propose une description formelle générique de la Blockchain comme une composition de type abstrait. [27] propose une implémentation par automates stochastiques visant à retrouver par la simulation les résultats de l'article précédent. [28] propose une modélisation, d'exécution de Smart Contracts.

L'utilisation de Blockchains dans les LEM s'est concentrée sur la simulation fonctionnelle des algorithmes de trading ou les mécanismes de transfert à base de cryptomonnaies [29], [6], [9], [30]. [13] présente un modèle d'optimisation pour une LEM basé sur une Blockchain. [9] propose une modélisation en 7 composants fondamentaux pour implémenter une LEM fondée sur un mécanisme d'enchères doubles cadencé par un smart contract sur une Blockchain de type Ethereum. Des agents gestionnaires de comptes, représentent les utilisateurs, et passent des ordres avec une stratégie d'enchères. Au fur et à mesure que les agents passent des ordres, des unités de compte d'énergie sont bloquées puis libérées pour le règlement pour garantir que chaque offre est couverte. Le modèle simule 100 foyers résidentiels avec des pas de temps d'enchères de 15 minutes.

[31] modélise une LEM basée sur un mécanisme de double enchère continue (CDA) qui met en correspondance acheteurs et vendeurs par détection d'offres compatibles. Un mécanisme de double enchère recueille les offres sur un intervalle de temps puis remet à zéro l'enchère à l'expiration. Les agents calculent les quotas en fonction de stratégies d'agressivité adaptative afin d'optimiser leur revenu. Une simulation faite sur Simulink représente une petite communauté (8 consommateurs et 6 producteurs) et 100 tours d'enchères. Le Bitcoin est utilisé comme solution de paiement, avec un protocole par preuve de travail. [11] et [Horta, 2017] se concentrent sur la simulation de l'infrastructure électrique et la performance de la solution Blockchain en prenant en compte les ressources informatiques, la consommation d'énergie et les coûts de transaction. Il monte qu'un protocole de consensus par preuve de travail (Proof of Work, PoW) énergivore et relativement lent, n'est pas la solution appropriée.

A notre connaissance, très peu d'articles adoptent un point de vue d''ingénierie de la sûreté de fonctionnement. Aucun article ne considère l'ensemble LEM + Blockchain en tant que système socio cyber physique géré par une infrastructure de calcul distribuée. [33] emploie une modélisation par Bloc Diagrammes Dynamiques (DRBD) équivalente à un réseau de Petri Stochastique modulaire. Mais il ne représente que la partie logicielle d'une Blockchain de type Hyperledger.

Il est donc pertinent d'évaluer la faisabilité d'une étude de disponibilité d'un système cyberphysique particulier : une LEM gérée par Blockchain. Dans cette étude de faisabilité menée par la R&D d'EDF, il s'agit de modéliser un système « socio cyber physique » à base de Blockchain afin d'en étudier le fonctionnement et la disponibilité. Le but est de comparer les effets des diverses incertitudes sur les choix de modélisation et les paramètres.

IV. DESCRIPTION DU SYSTÈME

Le fonctionnement de la LEM étudiée est décrit dans [34]. La partie physique est un quartier en France de 10 à 200 foyers avec une production d'énergie renouvelable (des panneaux photovoltaïques) individuelle et/ou partagée. Le but de la LEM est de maximiser la consommation locale d'énergie renouvelable et de réguler les échanges avec le réseau. Les habitants sont incités à acheter et vendre au niveau local par la LEM. Le quartier n'est pas autosuffisant ou peut être parfois en excédent; et il est connecté au réseau électrique global. L'équilibre du réseau et la facturation sont sous la responsabilité d'Enedis, distributeur d'électricité au niveau national (réseaux basse et moyenne tension).

Chaque foyer possède un compteur numérique type Linky. Il mesure les consommations et productions et les transmet à un Oracle (fournisseur de données de référence) qui vérifie les mesures et élabore les factures. Dans cette étude, dans le contexte France, l'Oracle est Enedis, représenté dans le modèle comme une base de données sécurisée.

Chaque foyer possède un équipement relié au compteur numérique : hardware intégré au compteur, Raspberry Pi, clé USB sécurisée, équipement domestique de type Box Internet... Chaque équipement héberge un Nœud de la Blockchain (Node).

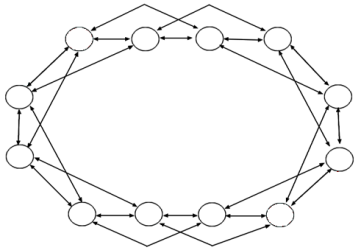
La technologie Blockchain employée est Ethermint. Cette variante d'Ethereum emploie Tendermint, un protocole BFT (Byzantine Fault Tolerant) [35] [Buchman, 2016]. Tendermint réalise des transactions rapidement avec une basse consommation d'électricité. Tendermint permet de se passer de protocoles de type preuve de travail (PoW) ou preuve d'enjeu (Proof of Stake, PoS) adaptés aux Blockchain publiques, mais peu efficaces du point de vue énergétique. Le protocole BFT est adapté aux Blockchains dites de consortium, avec accès contrôlé et nombre limité de nœuds de validation. Il consomme peu d'énergie tout en mitigeant les risques d'attaque par double-dépense ou par déni de service, les plus critiques pour ce type de technologie. Par construction, Tendermint tolère la défaillance (physique, logicielle) d'un tiers de ses nœuds de validation. Autrement dit, un bloc est ajouté à la suite de la Blockchain s'il a été validé par plus de 2/3 des nœuds de validation. Tendermint se décompose en un protocole de consensus et une interface logicielle. L'interface logicielle permet de coder les smart contrats dans un langage compris par la Blockchain Ethereum, comme par exemple [4].

Le réseau ainsi constitué comporte N=10 à 200 Nœuds connectés selon une topologie en i+1, i+2. Il est performant en terme de transactions traitées par seconde et de besoin de puissance de calcul, mais sa connectivité est limité, ce qui peut le rendre vulnérable à certaines topologies de défaillances multiples (cf. section VIII).

Du point de vue fonctionnel, un Nœud comprend:

- une couche logicielle qui comprend une couche logicielle système (OS) et une couche applicative appelée Apps, qui gère les comptes (Wallets) , crée et mémorise les blocs;
- une couche logicielle appelée Client qui exécute les smart-contracts.

FIG 1. Réseau de 12 nœuds en topologie i+1, i+2



Chaque Nœud héberge le Wallet (Compte) de l'utilisateur, et l'agent qui agit en son nom sur la LEM. Une partie des Nœuds (les Validateurs) gèrent aussi le contenu du registre distribué selon le protocole Tendermint. Ce sont les seuls qui ont accès à l'ensemble des informations contenues dans le registre et peuvent construire des blocs. Ces nœuds dits « lourds » requièrent plus de puissance de calcul que les nœuds dits « légers » qui permettent seulement de gérer et exécuter les smart-contracts et accéder aux Wallets. Les nœuds Validateurs sont les plus critiques et la simulation se concentre sur eux.

L'algorithme de la LEM développé par EDF R&D et SystemX est de type double-enchère [34]. La LEM fonctionne avec deux Smart Contracts: le contrat Wallet gère les comptes utilisateur et les Tokens; le contrat LEM gère la place de marché et agit comme un « commissaire-priseur » qui pilote les enchères. Dans ce modèle, l'Oracle transmet des données de production ou consommation validées, la partie Client des nœuds est notifiée et appelle le contrat LEM avec les nouvelles données, qui effectue les transactions, en suivant la fréquence d'appel des tours d'enchère.

Pour la modélisation fonctionnelle, une Plateforme Agent simule le comportement des foyers. Elle s'appuie sur une base de données de profils de consommation et de production simulés et réalistes de 200 foyers de deux villes françaises (Lille et Marseille) durant une semaine en été et une semaine en hiver. Ces données ont été fournies par l'outil de simulation EDF SMACH [36].

V. MODÈLE FONCTIONNEL

Le modèle fonctionnel résulte des travaux de [37]. L'objectif n'est pas de simuler une Blockchain en détail mais de s'intéresser à son impact au niveau du système. Il est une

étape nécessaire au modèle dysfonctionnel. En particulier il permet d'étudier l'impact de la topologie du réseau sur les conditions de tolérances byzantines et de vérifier des résultats de la littérature.

L'outil de modélisation fonctionnelle et dysfonctionnelle employé est PyCATSHOO, une librairie développée par EDF [38]. Il permet de modéliser des systèmes à l'aide d'automates distribués stochastiques hybrides qui sont une forme de représentation de PDMP (Piecewise Deterministic Markov Processes). L'équivalence formelle entre les automates stochastiques hybrides et les PDMP a été établie par [39]. La librairie peut être utilisée en version Python ou bien en C++. Pour cette étude, la librairie Python a été utilisée. Les simulations des systèmes sont réalisées par la méthode de Monte Carlo.

Note : L'étude n'exploite pas la partie continue de PyCATSHOO. Elle pourra être employée dans des études ultérieures pour modéliser des phénomènes de vieillissement ou de dérives de moyens de mesure, des aléas sur la production renouvelable, ou les courbes de consommation

Une difficulté est de modéliser un processus stochastique ayant un grand nombre d'états et des transitions entre ceux-ci (les transitions pouvant suivre tout type de loi: Weibull, Normale, etc.). PyCATSHOO permet de développer des automates stochastiques modulaires. Cette approche a été employée de façon concluante dans nos travaux précédents pour des systèmes programmés à haut niveau de redondance, avec des questions de réparabilité et de propagation de Défaillances de Cause Commune (DCC) [40].

Une autre difficulté est de travailler avec un niveau de modélisation suffisant pour des observations précises tout en ayant une complexité de modèle et des temps de calculs acceptables pour PyCATSHOO. Ainsi des hypothèses simplificatrices sur le fonctionnement du réseau sont faites dès cette étape.

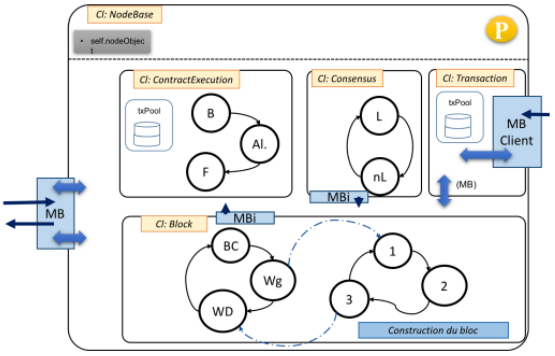
- Le temps est discrétisé sous forme de pas. Un pas représente un envoi et/ou une réception de message.
- Les connections réseau entre les composants du système sont de type TCP/IP et sont parfaites : il n'y a pas de pertes de paquets de données, les messages sont toujours reçus. On suppose que ces risques sont gérés par les couches réseaux non modélisées ici. Un réseau comme celui de la figure 1 est un graphe virtuel, issu du routage, Il est supporté par une couche physique, et la relation entre les deux couches n'est pas considérée.
- Les communications entre les nœuds et avec l'Oracle sont sans latence. Chaque appel de contrat de la LEM est immédiatement connu par tous les nœuds. On suppose que ces risques sont gérés par les couches réseaux non modélisées ici.
- Il ne peut pas y avoir de Fork spontané (plusieurs versions de la Blockchain présentes en parallèle sur une longue durée au sein du réseau). On suppose que ces risques sont gérés par une Blockchain de consortium.
- L'organisateur de la LEM n'intervient pas dans le modèle: on suppose que les contrats sont déployés une fois pour toute et qu'il n'y a pas d'intervention (changement de version…).

A partir de ces hypothèses, le comportement fonctionnel d'un Nœud est modélisé par:

- L'automate Bloc qui modélise la création des blocs de données ajoutés à chaque pas de temps au registre distribué.
- L'automate Consensus qui modélise le choix du bloc ajouté au registre et accepté par le réseau. Il se synchronise avec les nœuds du réseau et avec l'automate Bloc du même nœud.
- L'automate Contrat qui exécute les smart-contracts.
- L'automate Transaction qui permet de construire et envoyer/réceptionner des informations (smart-contracts).

Les entrées de la simulation sont les consommations et productions des foyers et l'état de la Blockchain. La sortie est un nouvel état. Les données relevées par le compteur numérique de chaque foyer, sont dans des tableaux de données lus au cours de la simulation.

FIG 2. Modélisation fonctionnelle avec PyCATSHOO



A. *Messages colorés*

Pour le modèle, des mécanismes ont été introduits pour représenter et échanger des types spécifiques à la Blockchain: les transactions, les blocs, les Wallets.

À l'instar des jetons colorés dans les réseaux de Petri, des messages dit "colorés" ont été introduits. Les messages colorés sont des objets convertis dans un tableau d'octets, puis déclarés comme des variables PyCATSHOO et partagés entre les automates.

Les messages colorés sont aussi utilisés dans le modèle dysfonctionnel. Ainsi, dans une certaine mesure, nous avons pu donner à PyCATSHOO le pouvoir d'expression des réseaux de Petri colorés stochastiques [41] [42].

B. *Etude de l'impact de la topologie du réseau*

Le modèle fonctionnel est combiné avec le modèle dysfonctionnel pour représenter l'effet de la localisation des défaillances byzantines dans le réseau (section VIII).

VI. MODÈLE DYSFONCTIONNEL

Ce modèle résulte des travaux de [37]. Les hypothèses de modélisation dysfonctionnelle sont les suivantes:

- Seuls les nœuds et les compteurs peuvent être dysfonctionnels. Dans une première étape, ils ont deux états : Marche (OK) ou défaillance complète de type Crash (KO). Un état KO de la couche physique entraîne un état KO de tous les éléments de la couche logicielle.

- En plus des défaillances individuelles fonctionnelles type Crash, il existe des défaillances Byzantines, avec des particularités liées à un système distribué (section VIII), et des Défaillances de Cause Commune (DCC), représentées par le modèle d'Atwood [44] (section VII).
- On néglige le risque de défaillance du réseau électrique et des panneaux PV.
- On néglige le risque de défaillance de l'Oracle (Enedis), et des communications réseau entre l'Oracle et les nœuds par rapports aux autres risques. Les données de production et de consommation relevées par Enedis sont toujours disponibles pour la partie Client d'un nœud.
- Les nœuds légers ne sont pas considérés. Leur risque est supposé négligeable par rapport à celui des nœuds validateurs.
- Les composants ne sont pas réparables. On considère que le système étudié est remis à neuf à chaque cycle de fonctionnement. Un cycle correspond à une période de 1 mois à un 1 an de fonctionnement selon la simulation.

Ainsi, le comportement dysfonctionnel d'un Nœud est modélisé par plusieurs automates stochastiques, avec des états OK/KO, et Byzantins (section VIII) :

- La couche physique représente l'équipement (RaspberryPi, clé USB sécurisée, Box Internet…) qui supporte le client logiciel d'un nœud de la Blockchain.
- L'élément logiciel Client qui construit envoie et reçoit les données (smart-contract)
- L'élément logiciel dApp, qui traite les données, crée et propose des blocs et contient la partie OS de la Blockchain (dApp signifie Distributed App, c'est-à-dire application fonctionnant dans une architecture répartie au sens de [51])

Le système est disponible tant que les critères suivants sont respectés :

- Pas plus d'1 compteur n'est défaillant,
- Au moins un nœud avec un ensemble Client-Node est disponible pour réaliser la tâche d'appel du smart contract de la LEM.
- Une proportion minimale de nœuds actifs n'a pas de comportement byzantin (Section VIII), fonction du protocole de consensus (Tendermint, Proof of Work, pas de protocole).

Quatre automates observateurs ont accès à tous les états dysfonctionnels de tous les composants représentés dans la simulation dysfonctionnelle. Ils gèrent une matrice d'états dysfonctionnels des composants. Les critères de disponibilité suivants sont implémentés dans ces automates.

- All : disponibilité en absence de défaillances byzantines (seuls les états KO sont pris en compte). Elle correspond à un protocole parfaitement tolérant aux défaillances byzantines, cas le plus optimiste.
- 2/3, Tendermint et Proof of Work considèrent la présence de défaillances byzantines avec respectivement des critères de 2/3, 1/3 et 51% des nœuds défaillants, byzantins nécessaires et suffisants pour causer une indisponibilité du système. 2/3 est le critère avec logiquement la meilleure disponibilité en présence de défaillances byzantines.

Cette modélisation purement dysfonctionnelle permet de suivre l'évolution de grandeurs globales, en particulier le nombre de nœuds en fonctionnement, de nœuds en panne et de nœuds byzantins. Cependant leur localisation dans le réseau, la topologie du réseau, n'est pas prise en compte. Pour cela, une « modélisation mixte » qui combine fonctionnel et dysfonctionnel est nécessaire, avec des automates observateurs dédiés aux défaillances byzantines (Section VIII).

VII. Défaillances de Cause Commune

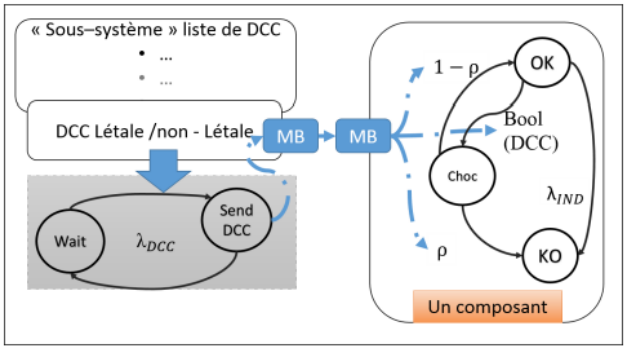
Les Défaillances de Causes Communes (DCC) impactent quasi simultanément et de façon corrélée plusieurs composants similaires d'un système. Les types de DCC, leur comportement et leurs conséquences sont définies en utilisant les méthodes établies par [40]. Le modèle d'Atwood [44] est employé pour représenter les DCC. Il fait partie de la famille des modèles de chocs. Les chocs peuvent être de deux types : létaux ou non létaux. Dans le premier cas, tous les composants du système sont impactés (DCC létale), sinon chaque composant à une probabilité ρ de tomber en panne suite au choc (DCC non-létale). Atwood fait l'hypothèse que les trois types de défaillances (indépendantes, létales et non létales) sont indépendantes. Par conséquent, il exprime le taux de défaillance global d'un composant par l'équation (1):

$$\lambda tot = \lambda ind + \omega + \mu * \rho \quad (1)$$

TAB 1. Paramètres du modèle d'Atwood

λind	Taux d'occurence d'une défaillance indépendante
ω	Fréquence d'occurrence des chocs létaux
μ	Fréquence des d'occurrence des chocs non létaux
ρ	Probabilité conditionnelle de défaillance d'un élément suite à un choc non létal

FIG 3. Modélisation des DCC avec PyCATSHOO



La figure 3 représente l'implémentation des DCC au niveau d'un automate. Une liste de DCC est instanciée dans un sous-système d'automates générateurs de chocs : Evènement climatique (foudre), Incendie, Usurpation non Byzantine, DCC sur compteurs numériques, etc.... Chaque DCC s'applique à un ensemble ou à un sous ensemble de composants similaires. Par exemple, les équipements d'un même immeuble de la LEM sont regroupés dans une DCC de sous-ensemble. On peut ainsi modéliser un « choc orienté »: par exemple, si la foudre tombe sur l'immeuble,

seuls les équipements de l'immeuble sont impactés. Cette implémentation des DCC est possible grâce aux messages colorés.

VIII. DÉFAILLANCES BYZANTINES

Un calculateur qui a un comportement byzantin envoie des messages contradictoires, mais crédibles, à ces destinataires et cause des incohérences dans le réseau. La particularité d'un nœud byzantin est qu'il a un comportement en apparence fonctionnel, ce qui le rend difficile à détecter (par exemple, modification malveillante de l'algorithme, condition imprévue qui ne stoppe pas le processus mais ne donne pas un résultat attendu…) [47]. Le terme Byzantin a été employé comme métaphore par Lamport [48]. Ce comportement peut être issu d'une action volontaire ou être le fruit du hasard [52]. Bien que très rares, ces défaillances sont particulièrement intéressantes à modéliser dans le cas de systèmes distribués résilients et tolérants aux défaillances indépendantes [49]. A partir de [48] et d'exemples concrets [49], une étude bibliographique sur les défaillances byzantines a été effectuée. Elle a permis de proposer un modèle pour décrire des comportements possibles selon différents scénarios et d'étudier des propositions d'algorithmes de tolérance aux défaillances byzantines. Les protocoles BFT (Byzantine Fault Tolerant) ont étés développés afin de tolérer ces défaillances [48]. Le compromis à trouver combine le temps d'exécution des algorithmes, les performances de système et sa capacité de tolérance byzantine. Les conditions de tolérances byzantines dérivent des travaux de Dolev [50]. Elles se résument ainsi:

Soit b la connectivité du système, n le nombre de nœuds dans le réseau et m le nombre de nœuds byzantins, alors le système sera disponible tant que $m < n/3$ *ou* $m < b/2$. *La condition* $m < n/3$ *est utilisée pour les cas d'application à des architectures Blockchains qui ont une connectivité forte.*

A. *DCC Byzantines*

Des types de DCC byzantines, ont étés créées (notées DBCC) :

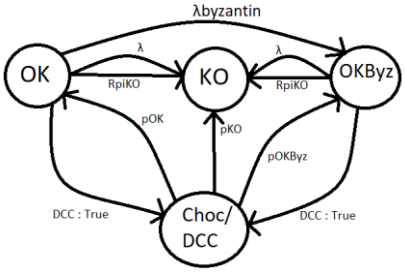
- *Hardfork* représente un changement du code au niveau du protocole de la Blockchain. Le Hardfork est une DBCC non-létale, car il agit sur la partie logicielle de d'une partie des nœuds seulement.
- *Echec de maintenance* représente une défaillance logicielle qui peut être induite par exemple par une architecture devenue obsolète, problèmes de stockage mémoire… ou une mise à jour logicielle inappropriée de la couche application.
- *Usurpation* représente le vol des clefs privées des Wallets d'une partie des comptes des utilisateurs de la LEM.
- *Attaque des Nœuds* représente une attaque informatique de type attaque de déni de service (DOS) qui peut rendre une partie des nœuds inopérants
- *Corruption Byzantin, Corruption Client*.

Les hypothèses suivantes sont faites :

- Si un compteur numérique devient byzantin, ce problème est détecté par l'Oracle (ici, Enedis) et corrigé par des tests de cohérence lors des transactions suivantes.
- Certaines défaillances de la couche physique d'un nœud peuvent entraîner un comportement byzantin au niveau d'un élément logiciel (hors OS).
- Les défaillances de l'OS d'un nœud sont critiques (arrêt, crash) et ne peuvent être byzantines. Le Nœud concerné est inactif au sein de la LEM et sa défaillance détectée.
- Les éléments dApp d'un nœud (partie traitement des données et construction des blocs, hors OS) et Client (smart-contracts) peuvent être byzantins.

Un automate spécifique, « aBasicByzantineComponent », en figure 4, a été ajouté à l'automate de base « aBasicComponent ». Ainsi un nœud peut passer dans l'état KO ou bien OKByz.

FIG.4. Automate « *aBasicByzantineComponent* » avec Défaillances byzantines

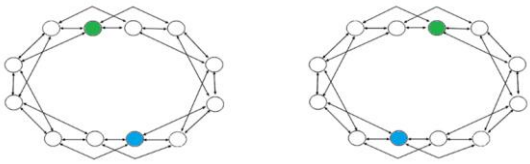


B. *Modélisation mixte des défaillances byzantines*

Cette modélisation permet d'étudier l'effet de la localisation des défaillances byzantines dans le réseau lorsqu'il a une faible connectivité, ce qui est le cas pour ce système en topologie i+1, i+2 (cf figure 1).

Le système est initialisé et suivi dans la modélisation dysfonctionnelle. Lorsqu'un nœud change d'état, le module dysfonctionnel récupère les données de chaque nœud et les envoie au module fonctionnel, qui lance alors une simulation de fonctionnement de la Blockchain selon le modèle du section V. Pour cela, il simule une série de configurations de liaisons émetteur-récepteur. Elles sont choisies dans un graphe qui représente la topologie du réseau des Nœuds, parmi les plus longues possibles pour être le pessimiste. Seuls des émetteurs OK sont choisis. Sinon, le module passe à la configuration suivante.

FIG. 5 Configurations successives d'un émetteur (Vert) et d'un récepteur (Bleu)



Chaque Nœud possède un registre local de messages. Les messages sont reçus par le récepteur sont analysés. Il est possible que lors du passage par un nœud défaillant KO, le message soit perdu en chemin. Le module effectue N simulations de transmissions (N dépend de l'ordre de grandeur du risque à estimer). Le critère de fonctionnement correct d'une connexion est le suivant : un récepteur doit recevoir le message d'un émetteur sans modification de la

part des nœuds intermédiaires. Il est respecté tant qu'un nombre minimal de nœuds sont fonctionnels et non byzantins.

Enfin, le modèle dysfonctionnel reçoit le pourcentage de transmissions correctes parmi N simulées. La modélisation dysfonctionnelle reprend son cours jusqu'au prochain changement d'état d'un nœud ou jusqu'à la fin de la période d'observation.

Plusieurs paramètres doivent être définis pour la simulation :

- Le nombre de nœuds
- La connectivité du graphe et l'orientation des liaisons (bidirectionnelle ou non)
- La topologie: topologie fixée de façon aléatoire ou déterministe, topologie variable à chaque séquence de façon aléatoire ou déterministe… [43], [45]
- Le nombre et la localisation des nœuds KO et des nœuds byzantins

Grâce à cette modélisation, il a été montré que la localisation des défaillances byzantines est importante, en particulier si un nœud correct est entouré de nœuds byzantins. Ainsi, pour la topologie i+1 ; i+2 quelques défaillances byzantines situées de façon particulière, par hasard ou de façon volontaire, suffisent pour perturber le système.

C. *Automates observateurs*

Trois automates sont dédiés à l'observation des défaillances byzantines.

- Un observateur Functional considère que le système est disponible si le pourcentage de transmissions correctes est supérieur à 95%. Ce taux maximal de transmissions incorrectes (5%) que le système peut supporter avant de se comporter globalement de façon byzantine a été choisi empiriquement. Il prend en compte l'incertitude autour de la détection de défaillances byzantines et d'autres aléas, non byzantins, non modélisés (perte de paquets, timeout, etc).

Deux observateurs nommés B2_Byz et B2_KO permettent d'observer des variables locales.

- L'observateur B2_Byz considère un critère de Dolev [51]: soit m le nombre de nœuds byzantins, N_{KO} le nombre de nœuds KO et b la connectivité du système : $m < 0{,}5.(b - 2.N_{KO}/b)$
- L'observateur B2_KO considère un nombre de nœuds KO inférieur à la connectivité moyenne d'un nœud plus 1. Soit N_{KO} le nombre de nœuds KO et b la connectivité du système : $N_{KO} < b+1$

IX. Quantification des paramètres

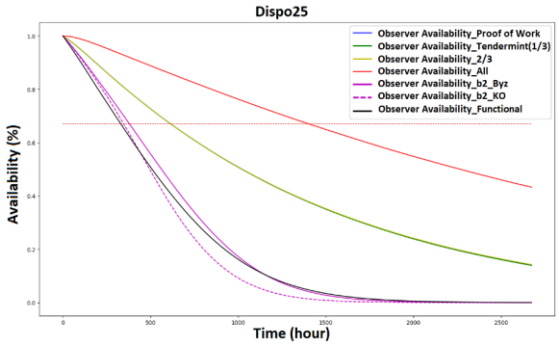
Ces hypothèses et étapes permettent d'étudier la disponibilité en présence des Défaillances de Cause Commune et de Défaillances Byzantines, combinées avec des défaillances indépendantes aléatoires. L'influence de plusieurs paramètres est évaluée, en particulier le nombre de nœuds et la connectivité. Les fréquences d'apparition et taux de défaillance sont calés sur un niveau de référence arbitraire afin de pouvoir observer des comportements intéressants avec des temps de calcul praticables par l'outil PyCATSHOO.

En absence de retour d'expérience sur ce type de systèmes, les valeurs des taux de défaillance et des divers paramètres de Sûreté de Fonctionnement sont estimés par « avis d'expert ». Cela n'est pas rédhibitoire pour cette étude de faisabilité, dont le but est de comparer les effets des diverses sources incertitudes sur la disponibilité globale du système.
Paramètres de la simulation:
Durée observée pour la simulation : 8760 h ou 2700 h pour l'étude de sensibilité
Nombre de séquences : 100000
Taux de défaillance compteur numérique (/h) : 1.10^{-5}
Taux de défaillance global d'un composant (/h) : 1.10^{-4}
Taux de défaillance Byzantin (/h) : $1.\ 10^{-5}$
Connectivité initiale : 4
Nombre de nœuds: 25
Fréquence f de DCC ou DBCC (/an): 0,5 à 2
Proba état KO : 0.035 à 0.125
Proba état Byzantin : 0,076 à 0,142 (0 si DCC)
Taux de messages byzantins: 0,001 à 0,855
Proba de propagation d'un message byzantin : 0 ; 0,25 ; 0,8
Note : Les fréquences d'apparition et taux de défaillance relatifs aux états Byzantins ont été amplifiées pour observer des comportements intéressants avec des temps de calcul raisonnables.

X. Simulations et résultats

Nous avons réalisé une simulation de Monte-Carlo selon les paramètres présentés ci-dessus. L'évolution de la disponibilité est présentée en figure 7. Elle montre les courbes de disponibilité moyenne obtenues selon les 7 observateurs décrits aux sections VI et VIII: All, 2/3, Tendermint, Proof of Work, Functional, B2_Byz et B2-KO. Les observateurs Tendermint et Proof of Work montent l'impact potentiel des défaillances byzantines si elles ne sont pas traitées correctement : si une défaillance sur 10 est byzantine, la disponibilité du système chute de façon significative. Les observateurs B2_Byz et B2_KO sont les plus pessimistes et sont presque superposés. Le système a une connectivité faible (b = 4), il est faiblement tolérant aux défaillances byzantines : d'après la condition m < 0,5.b, il suffit de 2 défaillances byzantines pour que le système soit byzantin, ce qui est observé dans certains cas.

FIG.7 Evolution des disponibilités moyennes en fonction du temps selon les 7 observateurs



XI. ETUDE DE SENSIBILITÉ

Le but est d'identifier les paramètres les plus influents sur la disponibilité du système avec une approche de type plan d'expérience. Les paramètres étudiés sont les suivants:

- Fréquence d'apparition des DCC /DBCC
- Probabilité d'état byzantin issu d'une DBCC
- Probabilité de propagation d'un message byzantin
- Taux de messages byzantins
- Taux de défaillance byzantin (Lambda Byz, vieillissement)
- Connectivité initiale du système
- Nombre de nœuds initialement dans le système

La variable d'intérêt, fixée arbitrairement, est t67%, temps nécessaire (en heures) pour observer une disponibilité moyenne de 67%

Cet article présente les résultats d'une étude de sensibilité simple. Chaque paramètre est fixé à un niveau bas et un niveau haut, puis les impacts sur la variable d'intérêt sont comparés (voir exemple en Tableau 2).

Avec des probabilités de défaillance, qui sont de l'ordre de 10^{-5}, il faut effectuer au moins 10^6 séquences pour un minimum d'exhaustivité. Avec la complexité actuelle de la modélisation sur PyCATSHOO, il faut pour cela environ 12 heures de simulation en utilisant 7 cœurs logiques sur un processeur 8 cœurs à 3,50 GHz. Ainsi, il a été choisi d'augmenter les paramètres initiaux en conservant les écarts relatifs et de suivre une durée d'observation de 2700 heures.

La démarche permet de classer les paramètres selon leur impact sur la disponibilité du système:

- Une ↗ de la fréquence d'apparition des DCC implique une ↘ de la disponibilité Impact fort
- Une ↗ de la fréquence d'apparition des DBCC implique une ↘ de la disponibilité Impact fort
- Une ↗ de la probabilité de défaillance byzantine issue d'une DBCC implique une ↘ de la disponibilité Impact faible
- Une ↗ du taux de messages byzantins implique une ↘ de la disponibilité Impact faible
- Une ↗ de $\lambda/^{\lambda_{Byz}}$ implique une ↘ de la disponibilité Impact fort
- La probabilité de propagation des défaillances byzantines n'a ici pas d'impact significatif
- Une ↗ de la connectivité initiale du système implique une ↗ de la disponibilité Impact fort
- Une ↗ du nombre de nœuds initiaux implique une ↗ de la disponibilité Impact modéré

La fréquence d'apparition des défaillances indépendantes et des DCC non byzantines renforce l'effet des défaillances byzantines. La connectivité initiale est importante car elle détermine la robustesse face aux défaillances byzantines mais aussi aux défaillances standards. L'augmentation des connexions permet de diminuer les chances d'isoler les nœuds et les rends moins sujets à propager de mauvais messages. Enfin, il est observé que plus le nombre de nœuds augmente et moins le système est disponible. En fait, dans le modèle, le nombre de nœuds augmente mais pas la connectivité. Donc, plus il y a de nœuds et plus il y a de chances qu'une défaillance standard ou défaillance byzantine ait un impact fort (Voir Tableaux 2 et 3 en fin d'article).

XII. CONCLUSION ET DISCUSSION

L'étude montre la faisabilité d'employer des approches de Sûreté de Fonctionnement pour étudier la disponibilité d'un système énergétique local appuyé par un système informatique distribué. Pour cela, il a fallu des adaptations et simplifications, et une large palette de concepts et d'outils: Automates Stochastiques modulaires équivalents à des réseaux de Petri colorés Stochastiques, Défaillances de Causes Communes, Modèles de Chocs, Défaillances Byzantines.

Pour cette étude, la notion de Défaillance Byzantine de Cause Commune a été développée pour représenter plusieurs comportements. Dans le modèle, les défaillances byzantines, même relativement rares, peuvent avoir un impact significatif sur la disponibilité du système surtout si elles se combinent avec des pannes aléatoires issues de composants moyennement fiables et un réseau de connectivité réduite. L'étude de sensibilité montre qu'il est intéressant d'augmenter le nombre de nœuds en même temps que la connectivité du système pour améliorer sa disponibilité. Mais un compromis doit être trouvé entre le nombre de nœuds et de connexions et les temps de traitements. Le système distribué étudié suit logiquement le théorème CAP (Consistency, Availability, Partitionning) [46]: les performances d'un système réparti sont liées et il faut trouver un équilibre entre fiabilité, latence et sécurité.

Pour la suite, il sera intéressant de travailler les points suivants :

- Affiner des hypothèses sur la latence de propagation des informations, le taux de transactions perdues, le taux de défaillance byzantine constant, l'ajout d'autres protocoles ou d'autres architectures de réseau …
- Discuter le seuil de 5% fixé empiriquement aux transmissions byzantines.
- Exploiter la partie continue de PyCATSHOO. Elle pourra être employée dans des études ultérieures pour modéliser des phénomènes de vieillissement ou de dérives de moyens de mesure, des aléas sur la production renouvelable, ou les courbes de consommation. De même, l'hypothèse des taux de défaillances constant peut être discutée. Certains cas observés semblent indiquer que l'occurrence de défaillance byzantine d'un composant augmente avec son âge.
- Mener une modélisation comparative pour la partie discrète avec un outil qui permet des preuves de modèles, étant donné la complexité du modèle et l'emploi de développements Python en complément à la composition d'automates stochastiques.

TAB 2. Exemple de résultat d'étude de sensibilité. Effets des variations (en %) des paramètres sur la variable d'intérêt t67% (extrait). Les valeurs avec une différence > 1% sont en gras sur fond jaune.

Observateur	PoW	1/3	2/3	All	B2Byz	B2KO	Functional
Défaut	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Fréq app DBCC * 10	**-79,26**	**-79,24**	**-79,27**	**5,99**	**-82,10**	**-80,52**	**-79,99**
Fréq app DBCC / 10	**56,67**	**56,78**	**56,60**	**-1,08**	**47,09**	**21,38**	**46,28**
Freq app DCC * 10	**-75,95**	**-75,92**	**-75,96**	**-82,51**	**-66,89**	**-67,62**	**-64,32**
PByzDBC/DBCL * 5/7	**-1,39**	**-1,42**	**-1,34**	0,00	-0,03	0,00	0,15
PByzDBC/DBCL * 13/14	0,24	0,31	0,21	-0,03	0,00	0,00	0,09
PbyzDBNC/DBNCL * 5/7	-0,41	-0,42	-0,38	-0,01	0,00	0,00	0,09
PByzDBNC/DBNCL * 24/25	0,00	0,03	0,02	0,01	0,00	0,00	0,24
PpropaDBCL 0,95	-0,02	0,00	0,00	0,00	0,00	-0,03	0,21
PpropaDBCL 0,5	-0,02	0,00	0,00	0,00	0,00	0,00	0,21
PprobaDBNCL 0,98	-0,02	0,10	0,00	0,00	0,00	-0,03	0,18
PprobaDBNCL 0,5	-0,02	0,00	0,00	0,00	0,00	0,00	0,15
PprobaDBC 0,01	-0,02	0,00	0,00	0,00	0,00	0,00	0,18
PprobaDBC 0,0005	-0,02	0,00	0,00	0,00	0,00	0,00	0,15
Lamba byz 1E-6	0,33	0,41	0,28	0,22	**6,37**	**1,18**	**14,85**
Lambda byz 1E-4	**-2,58**	**-3,92**	**-2,02**	**-1,82**	**-38,59**	**-10,60**	**-57,24**
Lambda 1E-3	**-11,85**	**-12,41**	**-11,63**	**-11,28**	**-75,84**	**-81,33**	**-74,46**
Lambda 1E-5	**4,49**	**4,52**	**4,47**	**5,19**	**13,37**	**20,08**	**11,24**
Connectivité initiale 8	-0,02	0,00	0,00	0,00	**62,75**	**23,80**	**28,85**
Connectivité initiale 2	-0,02	0,00	0,00	0,00	**-69,52**	**-27,28**	**-45,67**
Nombre de nœuds 15	**3,70**	**3,58**	**3,77**	**11,40**	**16,74**	**23,80**	**17,95**
Nombre de nœuds 35	**-3,48**	**-3,40**	**-3,52**	**-8,57**	**-16,61**	**-18,01**	**-12,78**

XIII. REFERENCES

[1] E. B. Hamida, K. L. Brousmiche, H. Levard, and E. Thea, Blockchain for enterprise: Overview, opportunities and challenges. ICWMC 2017, p. 91, 2017.

[2] Nakamoto, S. (2008). White paper : Bitcoin : A peer-to-peer electronic cash system. Technical report. URL www.bitcoin.org.

[3] Wood, D.G. Whitepaper (2013). URL https://github.com/ethereum/wiki/ wiki/White-Paper.

[4] Solidity: a contract-oriented, high-level language for implementing smart contracts," accessed: 2018-07-06. [Online]. Available: http: //solidity.readthedocs.io

[5]. Swan M. Blockchain: Blueprint for a New Economy. Sebastopol, CA: O'Reilly Media Inc; 2015

[6] Peck ME, Wagman D. Energy trading for fun and profit buy your neighbor's rooftop solar power or sell your own-it'll all be on a blockchain. IEEE Spectr Oct 2017;54(10):56–61.

[7] Loi-2017-227 (2017). Loi n° 2017-227 du 24 février 2017 - art. 9.

[8] EDF se prépare à conjuguer l'autoconsommation au pluriel, Pialot D., 2017. URL https://www.latribune.fr/entreprises-finance/industrie/energie-environnement/edf-se-prepare-a-conjuguer-l-autoconsommation-au-pluriel-733320.html (accessed 4.12.19).

[9] Mengelkamp, E., "A blockchain-based smart grid: towards sustainable local energy markets." Springer Berlin Heidelberg, 2018.

[10] Vandebron, marktplaats voor duurzame energie 2017. <https://vandebron.nl> [accessed August 3, 2017].

[11] Horta, J., Kofman, D., and Menga, D. (2016). Novel paradigms for advanced distribution grid energy management. https://arxiv.org/abs/1711.09565v1

[12] Blockchain application in renewable energy microgrids: an overview of existing technology towards creating climate - resilient and energy independent communities. Erica Svetec, Lucija Nad, Robert Pasicko, Boris Pavlin. 16th International Conference on the European Energy Market (EEM), September 2019.

[13] Munsing E, Mather J, Moura S. Blockchains for decentralized optimization of energy resources in microgrid networks. In: 2017 IEEE Conference on Control Technology and Applications (CCTA); 2017. p. 2164–71. doi:https://doi.org/10.1109/CCTA.2017.8062773.

[14] Power Ledger. Power ledger white paper. Power Ledger Pty Ltd, Australia, White paper; 2017. https://powerledger.io/media/Power-Ledger-Whitepaper-v3.pdf.

[15] Brooklyn Microgrid. Brooklyn Microgrid 2016. Online [accessed December 4, 2016]: http://brooklynmicrogrid.com/press/

[16] Alexandra Lüth, Jan Martin Zepter, Pedro Crespo del Granado, Ruud Egging. Local electricity market designs for peer-to-peer trading: The role of battery flexibility. Applied Energy. vol. 229. (2018) 1233-1243

[17] Morris, J., 2017. Blockchain in Energy: Powered by EWF.

[18] Real-time renewable energy incentive system for electric vehicles using prioritization and cryptocurrency. Tianyang Zhang, Himanshu Pota, Chi-Cheng Chu, Rajit Gadh. Applied Energy 226 (2018) 582–594.

[19] Sikorski JJ, Haughton J, Kraft M. Blockchain technology in the chemical industry: Machine-to-machine electricity market. Appl Energy 2017;195:234–46.

[20] Open Utility, Democratising energy. Open Util 2017. <https://openutility.com/ > [accessed August 3, 2017].

[21] Goal-Based Automation of Peer-to-Peer Electricity Trading Jordan Murkin, David Ferguson, Ruzanna Chitchyan. From Science to Society: New Trends in Environmental Informatics. Springer, 2017

[22] SonnenCommunity. sonnenCommunity. Sonnen 2016. <https://www.sonnenbatterie.de/en/sonnenCommunity > [accessed July 21, 2017].

[23] Mengelkamp E, Gärttner J, Rock K, Kessler S, Orsini L, Weinhardt C. Designing microgrid energy markets: a case study: the Brooklyn Microgrid. Appl Energy 2018;210:870–80.

[24] C. Rollet, 8 mai 2019EDF ENR et Logis Cévenols inaugurent la plus grande centrale photovoltaïque en autoconsommation collective de France. Catherine Rollet, 8 mai 2019 https://www.pv-magazine.fr/2019/05/08/edf-enr-et-logis-cevenols-inaugurent-la-plus-grande-centrale-photovoltaique-en-autoconsommation-collective-de-france/(accessed 4.12.19).

[25] Strang, K.A., Plaza, C., 2018. La blockchain appliquée à l'autoconsommation collective. https://www.see.asso.fr/e-see1/eventbyyear/all?page=1

[26] Anceaume, E., Pozzo, A.D., Ludinard, R., PotopButucaru, M., and Tucci-Piergiovanni, S. (2018). Blockchain Abstract Data Type.

[27]. Piriou, P.Y. and Dumas, J.F. (2018). Simulation of stochastic blockchain model.

[28] T. Abdellatif, K.L. Brousmiche. Formal verification of smart contracts based on users and blockchain behaviors models. IFIP NTMS International Workshop on Blockchains and Smart Contracts (BSC), Paris, France, February 2018

[29] Energy Demand Side Management within micro-grid networks enhanced by Blockchain. Sana Noor, Wentao Yang, Miao Guo, Koen H. van Dam, Xiaonan Wang. Applied Energy 228 (2018) 1385–1398

[30] Mengelkamp, E., Gärttner, J., Rock, K., Kessler, S., Orsini, L., and Weinhardt, C. (2018). Designing microgrid energy markets. A case study: The Brooklyn Microgrid. Applied Energy, 210, 870–880.

[31] J. Wang, Q. Wang, N. Zhou, and Y. Chi, "A novel electricity transaction mode of microgrids based on blockchain and continuous double auction," Energies, vol. 10, no. 12, p. 1971, 2017.

[32] J. Horta, D. Kofman, D. Menga, and A. Silva, Novel market approach for locally balancing renewable energy production and flexible demand. Dresden, Germany, 2017.

[33] Dependability Evaluation of a Blockchain-as-a-Service Environment. Carlos 33 IEEE Symposium on Computers and Communications (ISCC).

[34] Brousmiche, K.-L., Anoaica, A., Dib., O., Abdellatif, T., Deleuze, G., 2018. Blockchain Energy Market Place Evaluation : an Agent-Based Approach

[35] Kwon, J. (2014). Tendermint : Consensus without Mining. URL http://tendermint. com/docs/tendermint

[36] Haradji, Yvon & Poizat, Germain & Sempé, François. (2012). L'activité humaine et la conception d'une plateforme de simulation sociale.

[37] Hequet, G., 2019. Modélisation des défaillances byzantines et évaluation de leur impact sur la disponibilité d'un système. HAL Id: hal-02359788 https://hal.archives-ouvertes.fr/hal-02359788 Submitted on 12 Nov 2019

[38] Dynamic reliability modeling and assessment with PyCATSHOO : Application to a test case.

[39] G. Babykina, N. Brînzei, J.F. Aubry, G. Deleuze (2016). "Modeling and simulation of a controlled steam generator in the context of dynamic reliability using a Stochastic Hybrid Automaton", *Reliability Engineering and System Safety*, 152, 115-136.

[40] G. Deleuze, N.Brinzei, T. Hodicq. Représentation de défaillances de cause commune orientées dans un système distribué par réseaux de pétri colorés. Congrès Lambda Mu 20, Saint-Malo, septembre 2016.

[41] Petri, C.A. (1962). Kommunikation mit Automaten. Ph.D. thesis, Darmstadt.

[42] Aubry J.F., Brînzei N. (2015). Systems Dependability Assessment: Modeling with graphs and finite state automata, Wiley-ISTE.

[43] Miller, A., Xia, Y., Croman, K., Shi, E., Song, D., 2016. The Honey Badger of BFT Protocols, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, ACM Press, Vienna, Austria, pp. 31–42. https://doi.org/10.1145/2976749.2978399

[44] The binomial failure rate common cause model. Technometrics, 28(2), 139–148.

[45] Rabin, M.O., 1983. Randomized byzantine generals, in: 24th Annual Symposium on Foundations of Computer Science (Sfcs 1983). Presented at the 24th Annual Symposium on Foundations of Computer Science (sfcs 1983), IEEE, Tucson, AZ, USA, pp. 403–409. https://doi.org/10.1109/SFCS.1983.48

[46] Brewer, E.A., 2000. Towards robust distributed systems (abstract), in: Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing. PODC '00, 19th annual ACM symposium, ACM Press, Portland, Oregon, United States, p. 7. https://doi.org/10.1145/343477.343502

[47] - Sota, N., Higaki, H., 2015. Byzantine failure detection in wireless ad-hoc networks, in: 2015 36th IEEE Sarnoff Symposium. Presented at the 2015 36th IEEE Sarnoff Symposium, IEEE, Newark, NJ, USA, pp. 173–178. https://doi.org/10.1109/SARNOF.2015.7324664

[48] Lamport, L., Shostak, R., and Pease, M. (1982). The Byzantine Generals Problem. ACM Transactions on Programming Languages and Systems, 4(3), 382–401. doi:10.1145/357172.357176. URL http://portal.acm.org/citation.cfm?doid=357172.357176.

[49] Driscoll, D., 2012. DASHLink - Real System Failures. URL

https://c3.nasa.gov/dashlink/resources/624/ (accessed 1.7.19).

[50] Coan, B.A., Dolev, D., Dwork, C., Stockmeyer, L., 1989. The Distributed Firing Squad Problem. SIAM J. Comput. 18, 990–1012. https://doi.org/10.1137/0218068

[51] Abbes, H., 2016. Introduction aux systèmes répartis.