Etude de la disponibilité d'un système d'échange énergétique décentralisé à base de Blockchain

Gaël Hequet, Nicolae Brinzei, Gilles Deleuze

Etude de la disponibilité d'un système d'échange énergétique décentralisé à base de Blockchain

Gaël HEQUET*, Nicolae BRINZEI*,, Gilles DELEUZE*****

**ENSEM, Université de Lorraine, 2 avenue de la forêt de Haye, 54500 Vandœuvre-lès-Nancy*

***, CRAN, Université de Lorraine, CNRS UMR 7039, 2 avenue de la forêt de Haye, 54500 Vandœuvre-lès-Nancy*

***EDF R&D, Boulevard Gaspard Monge, 91120 Palaiseau*

Résumé : Dans un monde en perpétuel changement et en constante communication partout autour du globe, les systèmes distribués et réseaux deviennent omniprésents. Avec l'apparition des voitures électriques et des systèmes de production décentralisés, de nouveaux défis énergétiques apparaissent régulièrement. Dans ce contexte, EDF tente de trouver des solutions innovantes afin de faire face aux problématiques de demain. L'autoconsommation collective définie en France par la loi (LOI2017-227, 2017) permet de mettre en place des solutions d'autoproduction et de partage local d'énergie. Pour ce faire, EDF étudie comment utiliser la technologie Blockchain afin de créer un système d'échange. L'une des solutions étudiées est celle d'un système distribué. Malgré les propriétés de résilience, des questions de disponibilité et de sécurité du système se posent, notamment à propos des défaillances byzantines. Cet article est la suite des travaux effectués l'année précédente (Martin, 2018) et d'un projet de fin d'étude (Hequet, 2019). Cette étude propose des modélisations de comportements byzantins mais aussi des modèles fonctionnels et dysfonctionnels du système étudié avec la présence de DCC et défaillances byzantines. Ces modélisations, implémentées dans le logiciel PyCATSHOO, permettent d'évaluer l'impact des défaillances byzantines sur la disponibilité d'un système distribué en utilisant la méthode de Monte Carlo pour les simulations.

Keywords: Blockchain, Micro-grid, DCC, Disponibilité, Atwood, Défaillances byzantines, Systèmes distribués

REMERCIEMENTS

Avant d'entrer dans le vif du sujet, je souhaite tout d'abord remercier Nicolae Brinzei et Gilles Deleuze pour leurs conseils et leur aide pour l'écriture de cet article. De plus, je remercie EDF et l'ENSEM pour m'avoir permis de travailler sur ce sujet très intéressant.

1. INTRODUCTION

De nos jours, les systèmes distribués sont utilisés de plus en plus dans des domaines variés. Ces systèmes, bien que complexes à construire, sont très intéressants grâce à leurs propriétés de résiliences face aux pannes. Dans l'essor de l'industrie 4.0 avec l'utilisation de ces réseaux de communication, les systèmes distribués sont presque devenus la norme. Avec l'arrivée de « l'IoT » ou « Internet of Things », l'utilisation de systèmes distribués n'a fait qu'augmenter. Dans ce contexte, on retrouve la blockchain qui a fait couler beaucoup d'encre dans l'actualité depuis la création de la crypto-monnaie « Bitcoin » (Nakamoto, 2008). Les cas d'usage sont variés et de nombreuses entreprises tentent leur chance dans cette technologie. Cette technologie permet d'avoir un système gardant un log, un historique des échanges effectués dans le réseau. Ce log est entretenu par tous les utilisateurs du réseau et si l'un d'entre eux n'est plus capable d'effectuer sa mission, le système sera toujours disponible grâce aux autres utilisateurs. La technologie blockchain peut être utilisée dans le domaine financier avec les crypto-monnaies mais aussi dans le support de base de données ou encore l'échange d'énergie.

C'est pour ce cas d'usage d'échange énergétique qu'EDF s'intéresse à mettre en place une micro-grid d'échange énergétique décentralisé en utilisant la blockchain (Pialot, 2017). Dans le cadre du projet DURIN (Design & Use Reliable BlockchaINs), EDF souhaite comprendre et s'approprier la technologie blockchain en étudiant la faisabilité de ce cas d'usage et la disponibilité d'un tel système.

Le travail présenté dans cet article s'inscrit dans ce contexte en prenant du recul en s'intéressant globalement aux systèmes distribués ainsi qu'à un type de défaillance : les défaillances byzantines.

1.1 Etat de l'art

Un état de l'art a été établi précédemment lors d'une étude bibliographique sur les défaillances byzantines (Hequet, 2019). Cette étude s'intéressait principalement au comportement global des défaillances byzantines et a permis de déterminer un modèle simplifié sous la forme d'un diagramme d'activité (Fig. 1).

Dans le domaine des systèmes distribués, les défaillances byzantines sont connues et des algorithmes tentent de trouver diverses solutions face à ces défaillances. Depuis l'énoncé des « généraux byzantins » établi par Lamport (Lamport et al., 1982), des algorithmes « Byzantine-proof » sont proposés pour tolérer ce type de défaillance (Dolev, 1982) (Lamport et al., 1982) (Rabin, 1983) (Bracha, 1987) (Sota and Higaki, 2015) (Miller et al., 2016) (Buchman et al., 2018). Ces algorithmes, par différents moyens (watchdog, messages signés, retour d'expérience, votes, …), tentent de conserver au maximum les performances des systèmes tout en tolérant et/ou détectant le plus facilement et rapidement possible les défaillances byzantines.

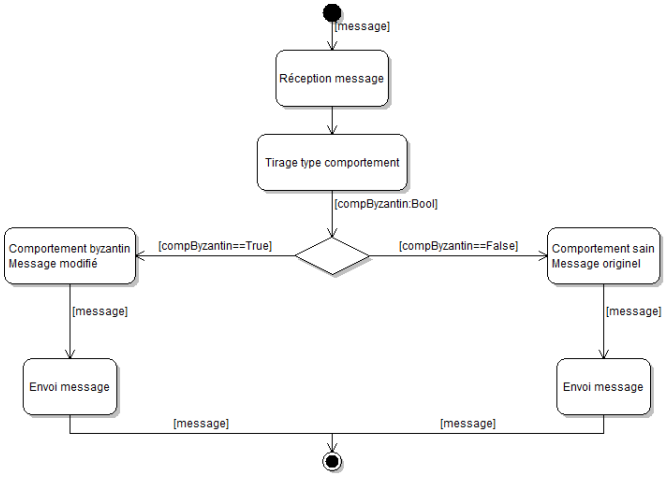

Fig. 1. Diagramme d'activité du comportement byzantin simplifié.

Des conditions de tolérances byzantines ont aussi été établies (Lamport et al., 1982) (Dolev, 1982). Soit « b » la connectivité globale du système (tous les nœuds ont la même connectivité), « n » le nombre de nœuds le constituant et « m » le nombre de nœuds byzantins, le système ne sera pas byzantin tant que $m < \frac{n}{3}$ ou $m < \frac{b}{2}$. En général, la littérature blockchain s'accorde sur la condition $m < \frac{n}{3}$.

Enfin, dans le cadre de cette étude, une modélisation dysfonctionnelle sur le système étudié a déjà été construite sur PyCATSHOO (Martin, 2018). Cette modélisation suit un système de n nœuds construits comme représenté ci-après (Fig. 2) sur une durée d'un an. Le système peut être soumis à diverses DCC (Défaillance de Cause Commune).

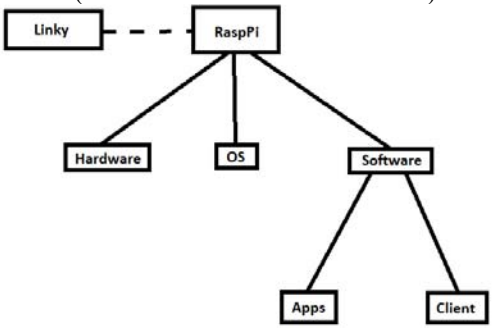

Fig. 2. Composants d'un nœud dans la modélisation dysfonctionnelle.

1.2 Le système étudié et les défaillances byzantines

Dans cette étude, il a été pris un certain recul par rapport au système réel. Le but est de modéliser un système distribué général pouvant être atteint par des défaillances de cause commune et des défaillances byzantines. Ce système sera donc modélisé par 25 nœuds avec une connectivité de 4. La topologie étudiée est appelée i+1, i+2 avec i l'ID du nœud comme représentée en figure 3.

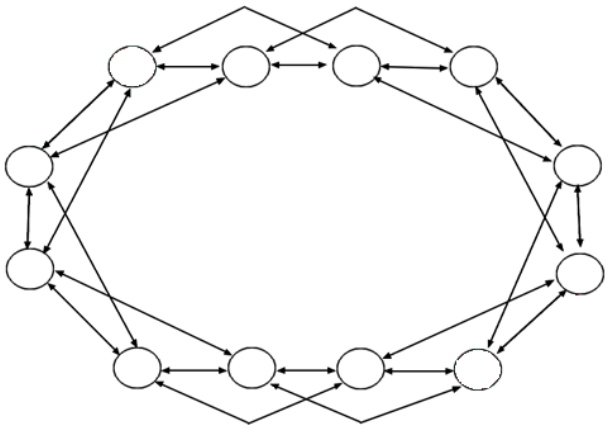

Fig. 3. Système distribué de 12 nœuds en topologie i+1, i+2.

Afin d'effectuer les différentes modélisations, une continuation des travaux sur les défaillances byzantines (Hequet, 2019) a été effectuée. Ces travaux ont permis de définir différents comportements byzantins selon leurs origines. Les valeurs des effets des comportements (probabilité de propagation, taux de messages byzantins, …) ont été définies arbitrairement pour représenter les différences entre comportements. Ainsi, il est plus facile d'observer un byzantin non coordonné qu'un byzantin coordonné.

Table 1. Comportement des Défaillances Byzantines selon leurs origines (Type de défaillance byzantine et intention initiale)

Type de défaillance	Volontaire	Non volontaire
Défaillance indépendante	DBNC (Défaillance Byzantine Non Coordonnée	DBNC
Défaillance Byzantine de Cause Commune Létale (DBCC Létale)	DBCL (Défaillance Byzantine Coordonnée Létale)	DBNCL (Défaillance Byzantine Non Coordonnée Létale)
Défaillance Byzantine de Cause Commune Non Létale (DBCC Non Létale)	DBC (Défaillance Byzantine Coordonnée)	DBNC

Table 2. Définition des effets des comportements byzantins

Nom	Propagation	Taux de messages byzantins envoyés
DBNC	Aucune propagation	4/5 de messages byzantins
DBCL	Forte propagation	¼ de messages byzantins
DBNCL	Forte propagation	4/5 de messages byzantins
DBC	Faible propagation	¼ de messages byzantins

Un ordre de prévalence entre les comportements a été établi. Par exemple, un comportement non coordonné et létal sera plus fort et visible qu'un comportement coordonné non létal. La prévalence est la suivante : DBNCL > DBNC > DBCL > DBC.

2. MODELISATIONS

2.1 Reprise du modèle dysfonctionnel

La modélisation créée par V. Martin (Martin, 2018) a été reprise et modifiée pour obtenir le diagramme de classe suivant :

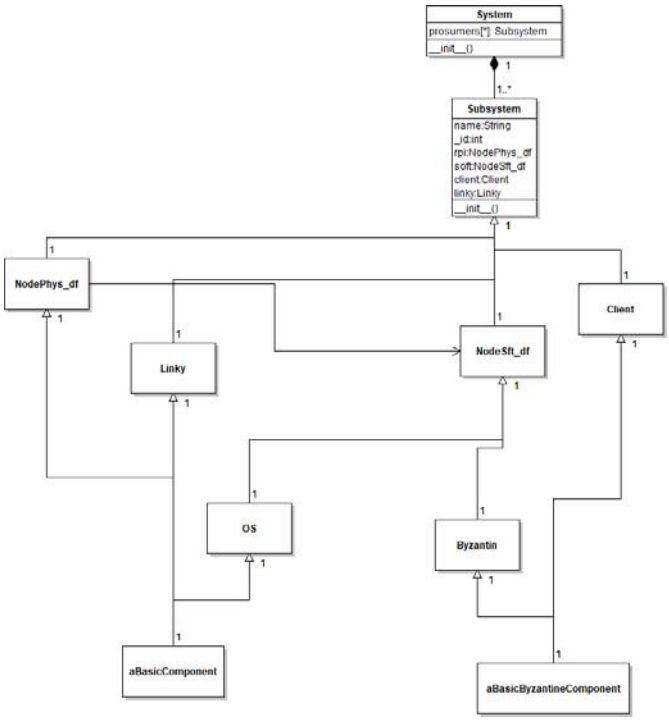

Fig. 4. Diagramme de classes de la modélisation dysfonctionnelle

Il a été choisi de pouvoir rendre la partie « Software » byzantine au travers des classes « Byzantin » et « Client ». Ainsi, lorsqu'une DCC ou DBCC surviendra, le composant touché pourra devenir byzantin. Une défaillance byzantine occasionnée sur la partie hardware ou sur le système d'exploitation sera visible au travers du composant « Byzantin » ou « Client » voire les deux. Pour simplifier le modèle il a été choisi que seules 2 classes seront des composants pouvant devenir byzantins avec la classe « aBasicByzantineComponent ». L'automate régissant ces classes a donc été modifié.

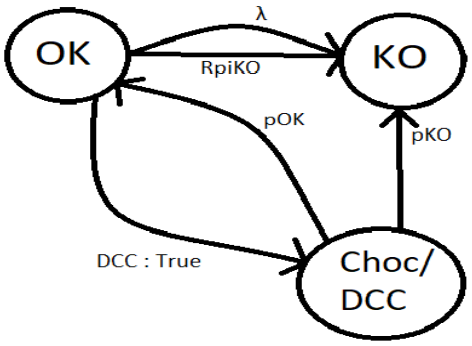

Fig. 5. Automate « aBasicComponent »

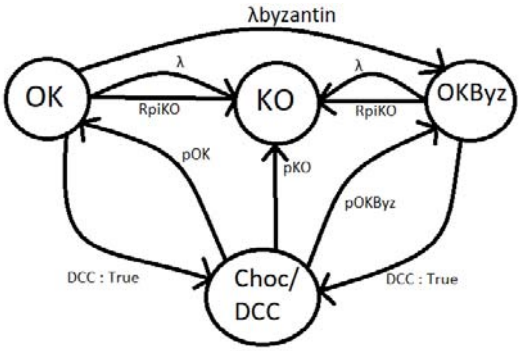

Fig. 6. Automate « aBasicByzantineComponent »

Cet automate (Fig. 6.) possède en plus un attribut non visible ici permettant de définir son comportement si celui-ci devient byzantin lors d'une DBCC ou bien par vieillissement.

Enfin, de nouvelles hypothèses de disponibilité ont été ajoutées sous la forme d'observateurs. Les hypothèses sont telles que le système devient indisponible si :

-All : Tous les nœuds sont KO

-2/3 : 2/3 des nœuds actifs sont byzantins

-Proof of Work : 51 % des nœuds actifs sont byzantins

-Tendermint : 1/3 des nœuds actifs sont byzantins

-B2Byz : $m > \frac{b - \frac{nKO}{2}}{2}$. Le terme 2*KO/b permet de rendre compte de l'apparition de pannes dans le système.

-B2KO : Soit nbKO le nombre de nœuds KO, nbKO > b + 1

Les observateurs « All », « 2/3 », « Proof of Work » et « Tendermint » étaient présents dans la simulation initiale. Les observateurs B2 ont été ajoutés à la suite d'études préliminaires sur la modélisation fonctionnelle décrite plus

bas. Les observateurs B2 sont plus sévères que les autres observateurs car ceux-ci prennent en compte la connectivité d'un nœud contrairement aux autres observateurs utilisant des variables globales du système (nombre de nœuds OK, KO, Byzantins). Chacun des observateurs vérifiera si le système est encore disponible selon ses hypothèses de disponibilité. Malgré ces observateurs, l'impact de la localisation des défaillances byzantines n'est pas pris en compte.

2.2 Modèle fonctionnel

Ce modèle est une continuation des travaux effectués lors d'un projet de fin d'étude (Hequet, 2019). Le but de cette modélisation est d'observer plus directement l'impact de la position dans le réseau des défaillances byzantines sur la disponibilité du système. Dans ce modèle, le niveau de modélisation est plus élevé pour simplifier les calculs et observer le comportement byzantin. Un nœud n'a ni composants hardware ni software. Un nœud est ici défini par son ID, son état, ses connexions, les messages reçus et par qui ils ont été envoyés. Chaque nœud suit l'automate montré en figure 7.

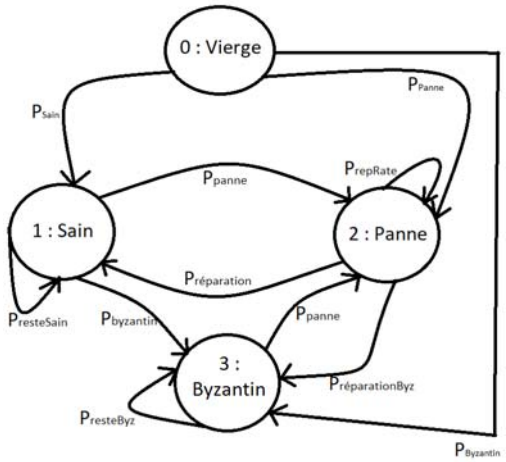

Fig. 7. Automate régissant l'état d'un nœud avec l'état 0 étant l'état d'initialisation

La topologie du système peut être choisie au début de la simulation ainsi que la présence d'un protocole byzantine-proof ou non. Pour construire cette modélisation, plusieurs hypothèses ont été faîtes :

-Le temps de propagation n'est pas pris en compte. Ce sont ici des étapes, des pas. Un pas représente un envoi et/ou réception de message.

-Les messages ne sont pas perdus (hors nœuds en panne) et sont bien reçus.

-Le système est disponible si le récepteur reçoit le message de l'émetteur sans modification de la part des nœuds messagers.

-L'émetteur est toujours sain.

Ainsi, cette modélisation fait voyager un message d'un émetteur à un récepteur. Un observateur vérifie alors qu'un message est bien reçu et qu'il n'ait pas été modifié. À l'aide de la méthode de Monte Carlo, un taux de disponibilité est défini. Par exemple si le taux est de 50% de disponibilité, cela signifie que 50% des systèmes ont bien respecté l'hypothèse de disponibilité. Les autres ont donc soit eu un timeout, soit un message modifié.

Le diagramme de classes de cette modélisation est le suivant :

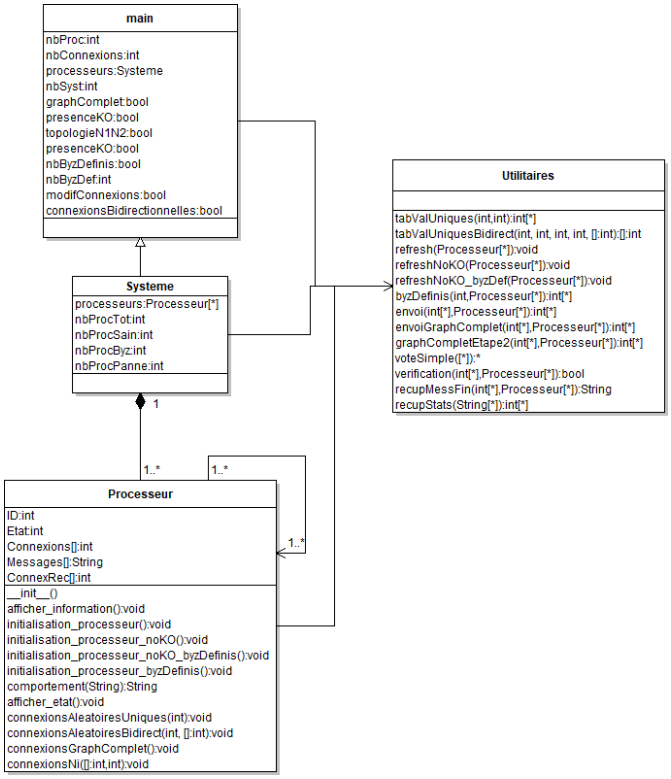

Fig. 8. Diagramme de classes de la modélisation fonctionnelle

Voici un exemple de propagation de message dans le cas d'un graphe complet :

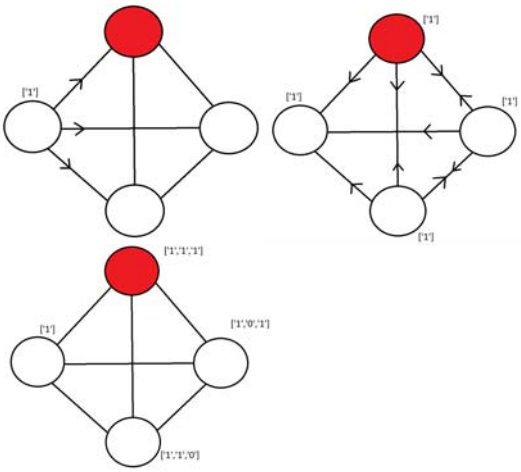

Fig. 9. Propagation d'un message dans un graphe complet en 3 étapes. Broadcast de l'émetteur, broadcast des nœuds messagers, vote final. (Blanc : Sain, Rouge : Byzantin)

Dans le cadre d'un graphe non complet, la propagation se fait par étapes :

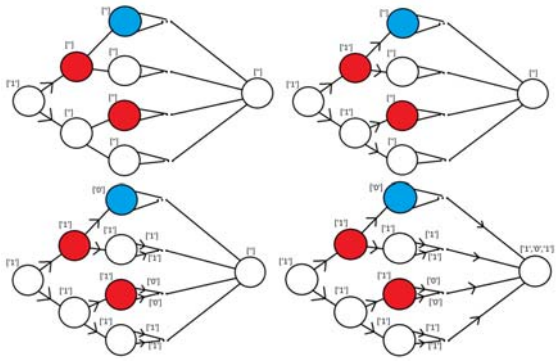

Fig. 10. Propagation d'un message dans un graphe non complet. (Blanc : Sain, Rouge : Byzantin, Bleu : Panne)

Dans le cas du système étudié :

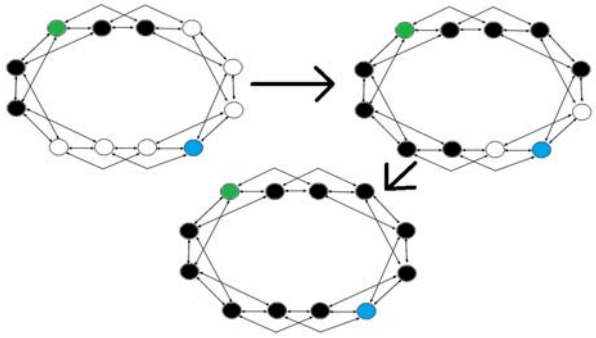

Fig. 11. Propagation d'un message dans le cadre du système étudié en topologie i+1, i+2. (Blanc : Sain reçu aucun message, Vert : Emetteur, Bleu : Récepteur, Noir : Sain message reçu)

Enfin, si une séquence possède un comportement byzantin, le système sauvegarde les paramètres du système afin d'observer le scénario créant ce comportement.

Dans cette modélisation, le vieillissement et l'apparition d'évènements aléatoires (DCC ou DBCC) ne sont pas implémentés. À chaque séquence, chaque nœud à une chance de changer d'état ou rester dans le même état.

2.3 Liaison fonctionnel et dysfonctionnel

La liaison entre les modèles a été effectuée afin de pouvoir observer l'apparition de DCC et de vieillissement. Le but de cette modélisation est d'utiliser le modèle dysfonctionnel comme support. Pour ce faire, un modèle fonctionnel a été ajouté en package au projet et peut être appelé par la partie dysfonctionnelle. Des fonctions permettant de récupérer les états des composants pour en obtenir un état global du nœud ont aussi été ajoutés.

Les composants des nœuds évoluent au cours du temps et sont observés. Si un composant change d'état, une capture des paramètres du système est faite. Les paramètres sont ensuite traduits puis utilisés pour lancer une simulation fonctionnelle. La simulation fonctionnelle vérifiera son hypothèse de disponibilité comme expliqué dans la partie *2.2*. Afin d'obtenir un taux de détection de disponibilité, tous les émetteurs sains du système sont testés avec des récepteurs les plus éloignés dans le système. L'objectif est d'avoir le plus de chance d'observer un comportement byzantin. La valeur de seuil de ce taux a été choisie empiriquement pour représenter la non-détection de comportements byzantins mais aussi l'apparition d'indisponibilité ponctuelle (action de maintenant, perte de paquet, timeout) n'impliquant pas d'indisponibilité définitive du système.

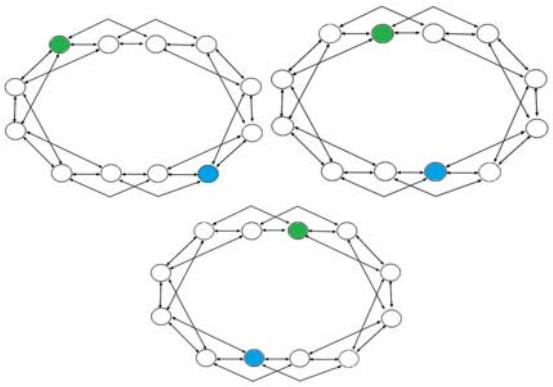

Fig. 12. Rotation de l'émetteur (Vert) et du récepteur (Bleu) dans le système.

Enfin, si le taux de détection de disponibilité est inférieur à 95%, la simulation considère le système comme indisponible. Un signal est ensuite envoyé à l'observateur « Functional » pour signifier que le système est indisponible d'après le modèle fonctionnel.

Il faut prendre en compte que la traduction des données se fait sur l'état global d'un nœud par rapport à l'état de ses composants. Le passage à un niveau plus haut de modélisation fait perdre de l'information sur les composants touchés. Les composants « Byzantin » et « Client » peuvent être tous les deux byzantins avec des comportements différents. C'est pour cela qu'une prévalence sur la force des comportements a été définie en partie *1.2*.

Voici le diagramme de séquence de cette modélisation :

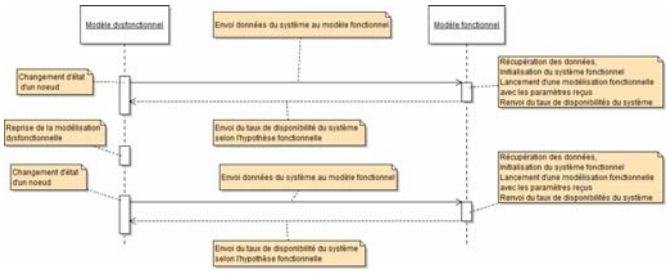

Fig. 12. Diagramme de séquence de la liaison du modèle dysfonctionnel avec le modèle fonctionnel.

La simulation utilise les mêmes observateurs que la modélisation dysfonctionnelle en ajoutant l'observateur « Functional ». Le résultat de la simulation est constitué de plusieurs courbes de disponibilités. Celles-ci sont, comme pour la partie dysfonctionnelle, des moyennes à l'issue de l'utilisation de la méthode de Monte Carlo.

Exemple de résultat d'une simulation :

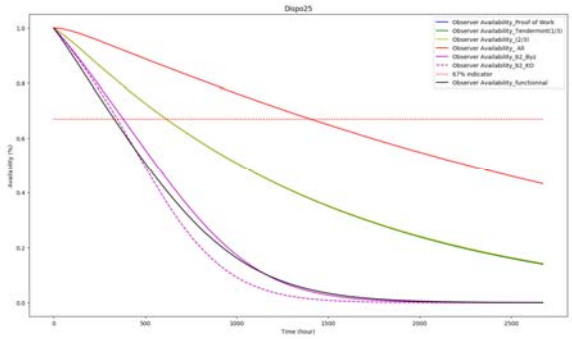

Fig. 13. Courbes moyennes de disponibilité (100000 séquences) selon différentes hypothèses (Rouge trait plein : All ; Vert : Tendermint ; Jaune : 2/3 ; Mauve trait plein : B2Byz ; Mauve Trait pointillés : B2KO ; Noir : Functional ; Rouge pointillés : indicateur 67%)

3. PLAN D'EXPERIENCE

Une fois le modèle construit, l'objectif est d'observer l'impact de différents paramètres sur la disponibilité du système étudié. Pour ce faire, un plan d'expérience a été monté en s'inspirant des travaux de (Del Bianco, 2016). Pour cette étude, les différentes valeurs absolues de taux de défaillance et probabilités ont été définies arbitrairement mais en conservant un sens en relatif. L'objectif est d'observer l'impact des défaillances byzantines sur un système et déterminer quels sont les paramètres importants.

Les paramètres modifiables sont les suivants : Fréquence d'apparition des DCC / DBCC ; Probabilité d'état byzantin issu d'une DBCC ; Probabilité de propagation byzantine ; Taux de messages byzantins ; Taux de défaillance standard et byzantin (vieillissement causant une défaillance byzantine) ; Connectivité initiale du système ; Nombre de nœuds initiaux dans le système.

À l'issu de ces modifications, le temps pour atteindre le seuil de 67% de disponibilité sera observé. Si une différence est observée cela signifiera l'impact du paramètre modifié.

3.1 Paramètres par défaut

Table 3. Paramètres généraux par défaut

Paramètre	Valeur
Durée d'observation	2700 heures
Séquences	100 000
Taux de défaillance standard Linky	1.10^{-5}
Taux de défaillance standard nœud	1.10^{-4}
Taux de défaillance Byzantin	1.10^{-5}
Connectivité initiale	4
Nœuds initiaux	25

Table 4. Paramètres par défaut des comportements

	DBNC	DBCL	DBNCL	DBC
Attribut	1	2	3	4
Probabilité de propagation	0	0.855	0.931	0.001
Probabilité de message byzantin	0.8	0.25	0.8	0.25

Table 5. Paramètres par défaut des probabilités d'état à l'issu d'une DCC / DBCC

	OK	KO	Byz
DCC	0.875	0.125	0
DBC et DBCL	0.823	0.035	0.142
DBNC et DBNCL	0.889	0.035	0.076

Table 6. DCC et DBCC dans la modélisation

Type	Nom	Fréquence	Comportement
DCC	Lightning	1/8760	DBNC
DBCC	Control Attack	1/17520	DBNCL
DCC	Blaze	1/8760	X
DCC	Malice	1/8760	X
DCC	Lightning Linky	1/8760	X
DCC	Hardfork	1/4380	X
DBCC	Update Error	1/17520	DBNCL
DBCC	Usurpation Byzantin	1/8760	DBC
DBCC	Usurpation Client	1/8760	DBC
DBCC	Corruption Byzantin	1/8760	DBCL
DBCC	Corruption Client	1/8760	DBCL

3.2 Modifications apportées

Pour ce plan d'expérience, une méthode de bifurcation séquentielle a été utilisée. Pour chaque paramètre, 2 modifications sont effectuées. Une modification supérieure et une modification inférieure. Les modifications effectuées sont les suivantes :

Table 7. Modifications apportées aux différents paramètres

Fréquence d'apparition des DBCC * 10	Fréquence d'apparition des DBCC / 10
Fréquence d'apparition des DCC * 10	Fréquence d'apparition des DCC / 10
P(Byz|DBC/DBCL) = 0.276	P(Byz|DBC/DBCL) = 0.069
P(Byz|DBNC/DBNCL) = 0.276	P(Byz|DBNC/DBNCL) = 0.0386
Probabilité propagation DBCL = 0.95	Probabilité propagation DBCL = 0.5
Probabilité propagation DBNCL = 0.98	Probabilité propagation DBNCL = 0.5
Probabilité propagation DBC = 0.01	Probabilité propagation DBC = 0.0005
Taux de messages byzantins DBNC/DBNCL = 0.95	Taux de messages byzantins DBNC/DBNCL = 0.5
Taux de messages byzantins DBCL/DBC = 0.5	Taux de messages byzantins DBCL/DBC = 0.125
Taux de défaillance byzantine = 1.10^{-6}	Taux de défaillance byzantine = 1.10^{-4}
Taux de défaillance standard = 1.10^{-3}	Taux de défaillance standard = 1.10^{-5}
Connectivité initiale = 8	Connectivité initiale = 2
Nœuds initiaux = 35	Nœuds initiaux = 15

4. RESULTATS

À l'issue de ce plan d'expérience, les valeurs des temps d'atteinte du seuil des 67% ont été relevés et comparés aux résultats de la simulation avec les paramètres par défaut. Par convention les grandeurs présentant des différences supérieures à 1% indiquent un impact sur la disponibilité du système (Del Bianco, 2016).

En s'intéressant aux données recueillis par la modélisation fonctionnelle, il a été vu que sur le système étudié, la localisation des défaillances byzantines avait un fort impact sur la disponibilité. D'après les grandeurs de notre système, il suffit théoriquement de 2 nœuds byzantins pour rendre le système byzantin. La connectivité étant de 4, si m est supérieur ou égal à la connectivité divisée par 2, le système peut présenter des comportements byzantins.

En s'intéressant aux données recueillis par la modélisation liant les deux modèles, voici quelques exemples de courbes issues de ces expériences :

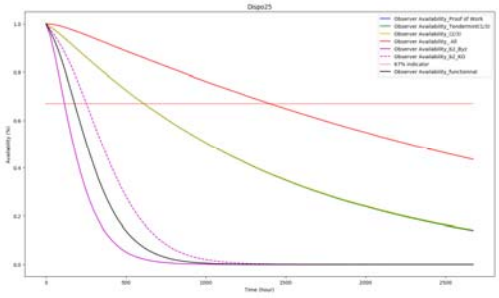

Fig. 14. Courbes moyennes de disponibilité avec modification de la connectivité initiale (b = 2)

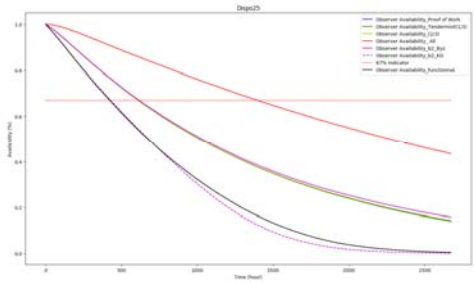

Fig. 15. Courbes moyennes de disponibilité avec modification de la connectivité initiale (b = 8)

Pour l'exploitation des données, il a été défini ces critères d'impact :

Table 8. Paramètres à l'impact fort sur la disponibilité

Fréquence d'apparition des DCC	Fréquence d'apparition des DBCC	Taux de défaillance standard
Taux de défaillance byzantine	Connectivité initiale	

Paramètre à l'impact modéré sur la disponibilité du système : - Nombre de nœuds initiaux

Table 9. Paramètres à l'impact faible sur la disponibilité

Probabilité d'état byzantin à l'issue d'une DBCC	Probabilité de propagation de l'état byzantin
Taux de messages byzantins	

5. CONCLUSIONS

Cette étude, avec les paramètres définis lors des simulations, a permis d'observer que les défaillances byzantines peuvent avoir un impact significatif sur la disponibilité d'un système. Cet impact est augmenté si des pannes indépendantes s'ajoutent. Il faut donc assurer un niveau minimum de fiabilité, en particulier du HW, pour assurer une disponibilité du système, même s'il est résilient. Dans cette étude, le terme de Défaillance Byzantine de Cause Commune a été créé afin

de mieux définir l'origine de ces défaillances. De plus, des comportements byzantins ont été donnés afin de montrer que les conséquences de ces défaillances peuvent être très variées.

Il faut rester critique face à ces résultats. Les grandeurs sont arbitraires et ne représentent pas des données réelles (très peu de données). De plus, la question du comportement du taux de défaillance byzantine se pose. D'après les exemples montrés dans une étude précédente (Hequet, 2019), le taux de défaillance byzantine se comporte comme le taux de défaillance standard. Il y a plus de chance de l'apparition de défaillance byzantine en début et en fin de vie.

6. PERSPECTIVES

Des suites pourraient être étudiées après ces travaux. Les DCC orientées (Deleuze et al., 2016) pourraient avoir un impact intéressant sur l'occurrence des défaillances byzantines. Ces DCC s'appliquant à des nœuds pouvant être proches spatialement ou bien connectés entre eux d'autres manières pourraient s'appliquer à un cadre byzantin.

De plus avec les modèles actuels, il pourrait être intéressant d'étudier d'autres types d'architecture et protocole pour étudier leur impact sur la disponibilité.

Enfin, une autre étude pourrait être intéressante. Le système étudié est un système physique, les connexions sont physiques, faisant que le graphe en découlant ne peut être complet (problématiques spatiales et financières). Or, dans le cadre des blockchains, les graphes sont considérés comme complets grâce au routage. On peut donc avoir des différences mais aussi des liens entre les graphes dits « réels » représentants le support physique et les graphes dits « virtuels » représentant les connexions grâce au routage. Dans la littérature en général, il est admis que les réseaux distribués blockchains ont assez de connexions pour les simplifier en graphes complets. Mais dans le cadre de petites blockchains, ces hypothèses sont plus difficiles à respecter. Si la connectivité est trop basse, alors la condition de 1/3 de byzantins tombe et comme dans le cas étudié, la tolérance byzantine chute. Il serait intéressant d'étudier, dans le cadre de la sûreté mais aussi de la sécurité, le lien entre un nœud physique et les nœuds virtuels. : Si un nœud physique devient byzantin, combien de nœuds virtuels supportés par ce nœud physique deviendront byzantins ?

REFERENCES

(Bracha, 1987) - Bracha, G., 1987. An O(log n) expected rounds randomized byzantine generals protocol. Journal of the ACM 34, 910–920. https://doi.org/10.1145/31846.42229

(Buchman et al., 2018) - Buchman, E., Kwon, J., Milosevic, Z., 2018. The latest gossip on BFT consensus. arXiv:1807.04938 [cs].

(Del Bianco, 2016) – Del Bianco, M., 2016. Modélisation et évaluation des Défaillances de Cause Commune de systèmes programmés par Réseaux de Petri colorés, temporisés, hiérarchiques.

(Deleuze et al., 2016) - Deleuze, G., Brinzei, N., Hodicq, T., 2016. Représentation de défaillances de cause commune orientées dans un système distribué par réseaux de pétri colorés. Presented at the 20e congrès de maîtrise des risques et de sûreté de fonctionnement, Saint-Malo.

(Dolev, 1982) - Dolev, D., 1982. The Byzantine Generals Strike Again.

(Hequet, 2019) - Hequet, G., 2019. Modélisation des défaillances byzantines et évaluation de leur impact sur la disponibilité d'un système. Article bibliographique universitaire.

(Lamport et al., 1982) - Lamport, L., Shostak, R., Pease, M., 1982. The Byzantine Generals Problem. ACM Transactions on Programming Languages and Systems 4, 382–401. https://doi.org/10.1145/357172.357176

(LOI2017-227, 2017) - LOI n° 2017-227 du 24 février 2017, 2017. DEVR1623346L.

(Martin, 2018) - Martin, V., 2018. Modélisation et évaluation des performances d'un système d'autoconsomation collective à base de blockchain avec PyCATSHOO par des Automates Stochastiques Hybrides (Rapport de stage). EDF R&D PERICLES.

(Miller et al., 2016) - 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 - CCS'16. Presented at the the 2016 ACM SIGSAC Conference, ACM Press, Vienna, Austria, pp. 31–42. https://doi.org/10.1145/2976749.2978399

(Nakamoto, 2008) - Nakamoto, S., 2008. Bitcoin: A Peer-to-Peer Electronic Cash System.

(Pialot, 2017) - PIALOT, D., 2017. EDF se prépare à conjuguer l'autoconsommation au pluriel [WWW Document]. www.latribune.fr. 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).

(Rabin, 1983) - 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

(Sota et al., 2015) - 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