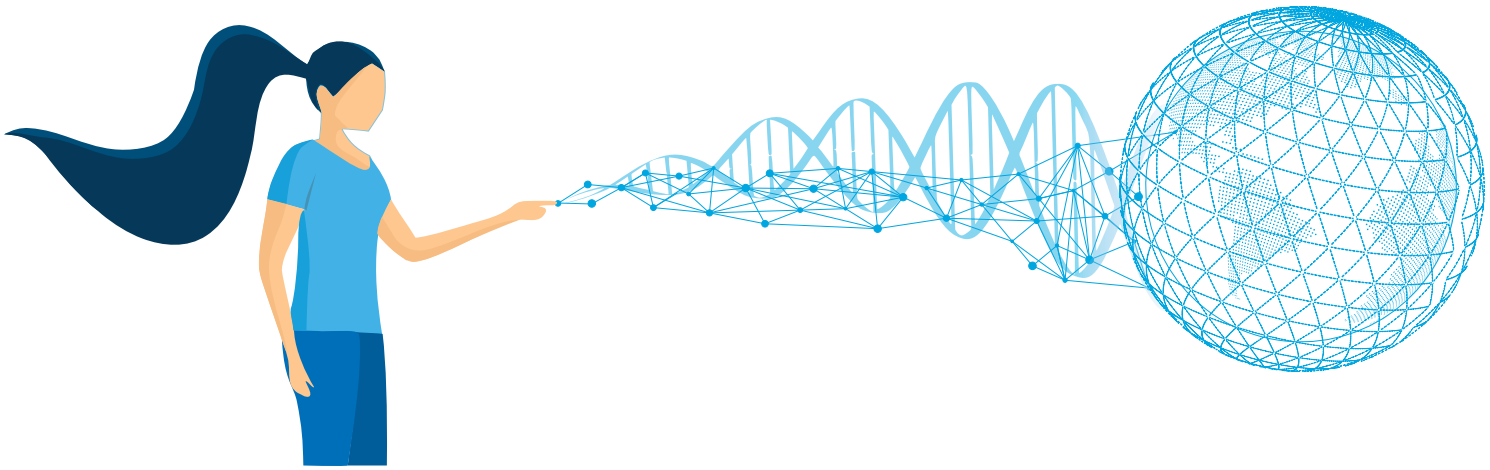
UNIRIS

Be The Only Key

Yellow Paper Saison 1

Un Réseau réellement décentralisé et sans limites

14 Juillet 2019



L'informatique a propulsé l'humanité au coeur d'un nouveau paradoxe. Chaque individu doit désormais disposer d'une identité numérique, clef de toutes les interactions économiques et sociales : s'inscrire sur un réseau, se connecter à une plateforme, payer en ligne, s'abonner à une app, chatter, s'envoyer des photos... Sans identité numérique, tout est verrouillé. Elle est devenue obligatoire pour exister. Or, aucune solution ne permet à ce jour de protéger ou contrôler cette même identité. Il n'existe aucun moyen pour s'identifier de façon absolument sûre et certaine. Il n'existe aucun moyen pour un individu de prouver que c'est bien lui qui se connecte, ni de lui garantir que personne d'autre ne se connectera à sa place - tandis qu'il n'a jamais été aussi facile pour un attaquant de prendre possession de n'importe quel ordinateur, smartphone, tablette (social engineering, exploitation des failles zero days, etc.).

Comment s'assurer qu'un système informatique puisse fonctionner sans avoir besoin de faire confiance à qui que ce soit ? Ou, plus exactement : comment faire pour qu'un système *dont on sait par avance que certains vont essayer de le détourner* puisse garantir la sécurité des données et des informations ? Le principe des Blockchains, et tout particulièrement celle du réseau Bitcoin, offre une solution parfaite à ce problème. Toutefois, malgré sa perfection conceptuelle et la robustesse effective dont elle a fait preuve ces dix dernières années, plusieurs faiblesses sont apparues :

- Pour les utilisateurs, la complexité de sécurisation et sauvegarde des clés,
- Une scalabilité limitée, qui ne permet pas une utilisation planétaire quotidienne,
- Une consommation énergétique trop importante,
- Une professionnalisation des mineurs, conduisant de fait à une forme de recentralisation,
- Une gouvernance fratricide et occultant un acteur essentiel : l'utilisateur.

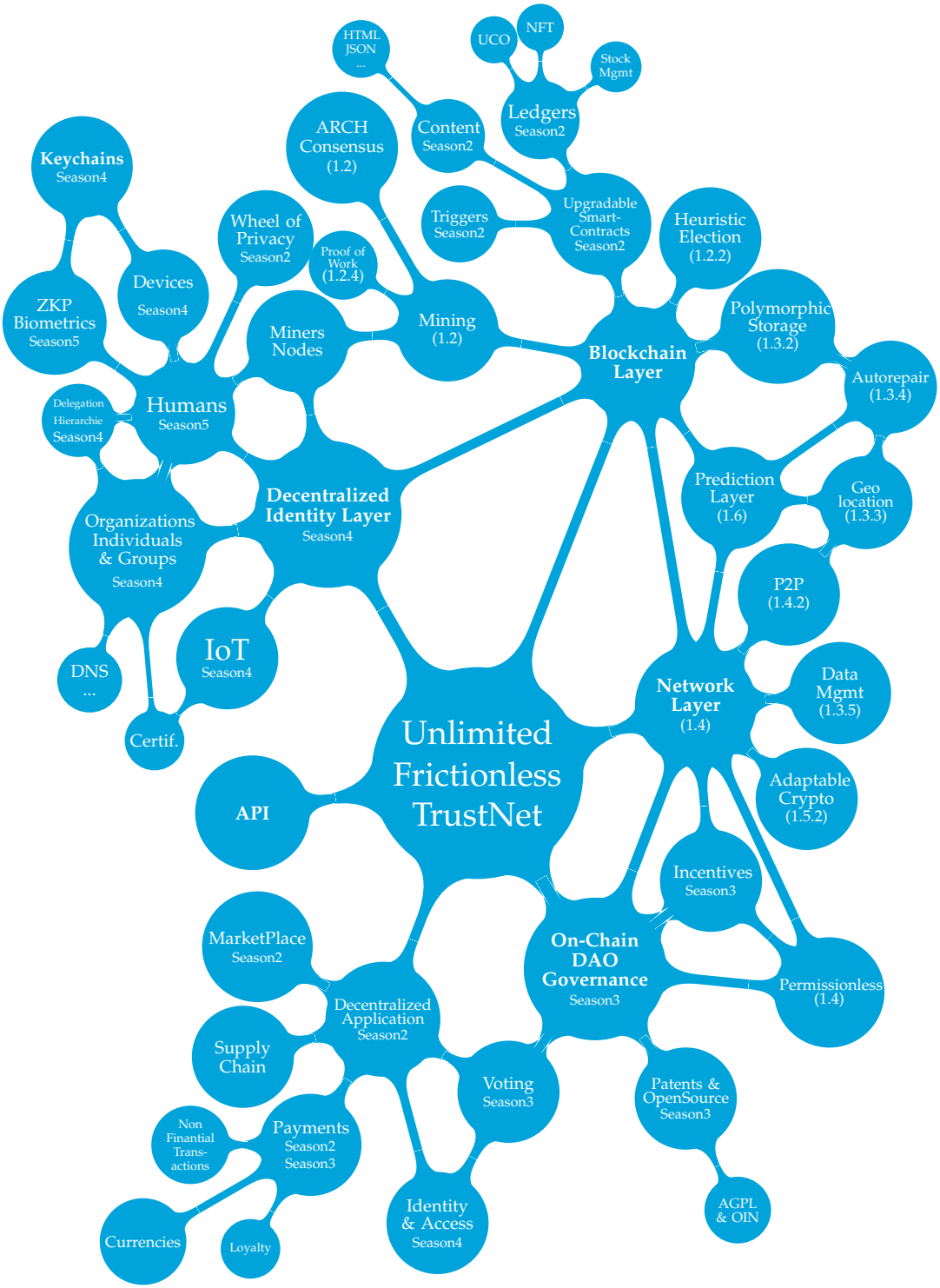
Les Blockchains actuelles ne répondent pas non plus au problème crucial de l'identité : comment faire pour que n'importe qui puisse garantir son identité derrière les transactions et sans pour autant se dévoiler (ouvrir sa porte de maison ou démarrer sa voiture, payer, sauvegarder ses clés Bitcoin, ou même voter, etc.) sans avoir besoin d'un mot de passe ou d'un objet (carte, badge, smartphone) ? Les objets s'égarent, se perdent, se cassent ou se volent, tandis que les mots de passe se notent et se recopient ou s'oublient, et sont d'autant plus difficiles à retenir qu'ils sont sécurisés...

La solution, connue depuis toujours, est d'utiliser le corps lui-même, mais d'une façon jamais envisagée à ce jour. Les solutions actuelles consistent à sauvegarder et comparer des images (reconnaissance faciale, digitale, rétinienne...), ce qui présente des risques de failles similaires à ceux déjà pointés. Face à cela, Uniris apporte une technologie permettant d'utiliser une différence fondamentale entre tous les individus, indépendante de tout dispositif personnel. Sur cette base, des clés cryptographiques universelles sont générées à la volée puis supprimées sans jamais être stockées, tout en permettant aux individus de garder un contrôle total sur cette identité. Ce nouveau type d'accès se fait à travers une blockchain elle aussi repensée, afin de profiter des enseignements du réseau Bitcoin tout en en préservant le meilleur : « une décentralisation capable de prouver que le contrôle n'est pas détenu par certains, mais par tous ».

Dans la suite de ce document et à travers 5 publications successives, nous présentons la solution d'Uniris et comment sa blockchain permet notamment, en contrepoint de chacune des faiblesses des Blockchains actuelles, de développer un réseau disposant d'une identité décentralisée et interopérable – couvrant quasiment tous les usages du quotidien – soutenant le million de transactions par secondes en évitant les failles connues – disposant d'une gouvernance intégrant la communauté dans son ensemble, et, enfin, offrant la perspective d'un dispositif d'interactions Humains/Machines infalsifiable, universel et sans risque.

Le Yellow Paper du réseau Uniris se décompose en 5 saisons :

Saison 1 *Un Réseau réellement décentralisé et sans limites*
Saison 2 *Des Applications Ancrées dans la Vie Réelle*
Saison 3 *Gouvernance et Système Économique taillés pour l'humanité*
Saison 4 *Une identification universelle sans le problème du tiers de confiance*
Saison 5 *L'Humain augmenté*



Saison 1

Un Réseau réellement décentralisé et sans limites

Compte tenu des contraintes matérielles et physiques universelles, des milliards de transactions ne peuvent pas être intégrées dans une seule chaîne de bloc. De la même façon, quelleque soit la méthode de consensus, il est impossible d'assurer un consensus universel sur des milliards de transactions en interrogeant tous les noeuds du réseau. Enfin, compte tenu du fonctionnement des réseaux distribués (P2P) il n'est pas possible de garantir la fraîcheur (consistance) d'une donnée sur un réseau asynchrone, sauf comme dans le cas du réseau Bitcoin en ralentissant significativement le réseau par l'intermédiaire du calcul du nonce d'un bloc dans la preuve-de-travail.

Uniris a résolu ces trois problèmes, de la façon suivante :

Chaînes de transactions au lieu de blocs de transactions chaînées, chaque bloc est réduit en sa version atomique, c'est-à-dire que chaque bloc ne contient qu'une seule transaction - chaque transaction disposera de sa propre chaîne.

Consensus ARCHE est une nouvelle génération de consensus *Atomic Rotating Commitment Heuristic* en français "Validation Atomique par élection heuristique tournante" permettant d'obtenir un Consensus Universel à partir d'une infime partie du réseau.

Système de Réplication Prédictif, Optimisé capable de s'autoréparer au lieu de synchroniser les transactions de façon anarchique sur l'ensemble du réseau, chaque chaîne de transactions sera stockée de façon déterministe et ordonnée sur un ensemble de noeuds - ainsi chaque noeud, de façon autonome, connaitra l'ensemble des noeuds hébergeant une transaction donnée et pourra ainsi soulager le réseau en interrogeant uniquement les noeuds « élus » les plus proches.

Réseau Distribué sans point de saturation basé sur la Multidiffusion Supervisée, le réseau de pair-à-pair utilise un mécanisme d'autodécouverte basé sur les connexions entrantes et le mécanisme des chaînes de transaction du réseau pour maintenir une vision qualifiable et de confiance en générant un minimum de nouvelles transactions sur le réseau.

La Saison 1 aborde tout particulièrement les sujets suivants :

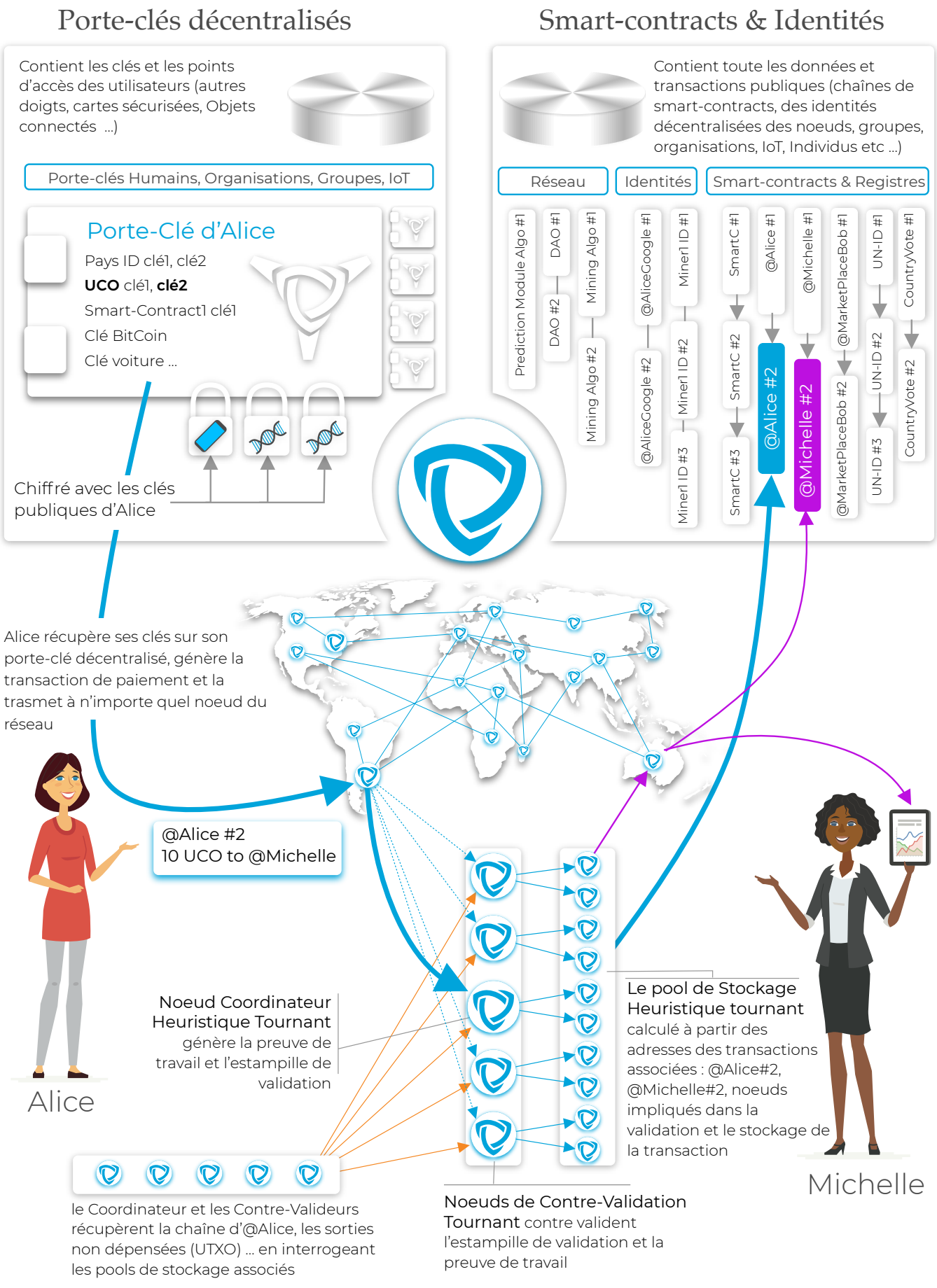


Figure 1.1: Fonctionement Global du réseau Uniris

1.1 L'Eldorado inexploré des Chaînes de Transactions

Comme décrit précédemment, chaque bloc est réduit en sa version atomique, c'est-à-dire que chaque bloc est une transaction disposant de ses propres preuves de validation qui vont lui permettre d'être associée à une chaîne donnée. Chaque transaction dispose également d'une signature supplémentaire *(Figure 1.2)* correspondant à la signature du dispositif à l'origine de la génération de la transaction. Cette signature est utilisée notamment (dans la Preuve de travail (Proof-of-Work) et peut être intégrée en tant que condition nécessaire pour la validation d'une transaction (par exemple dans le cadre d'un vote électronique pour s'assurer de l'identité biométrique d'une personne par l'intermédiaire d'un dispositif reconnu.

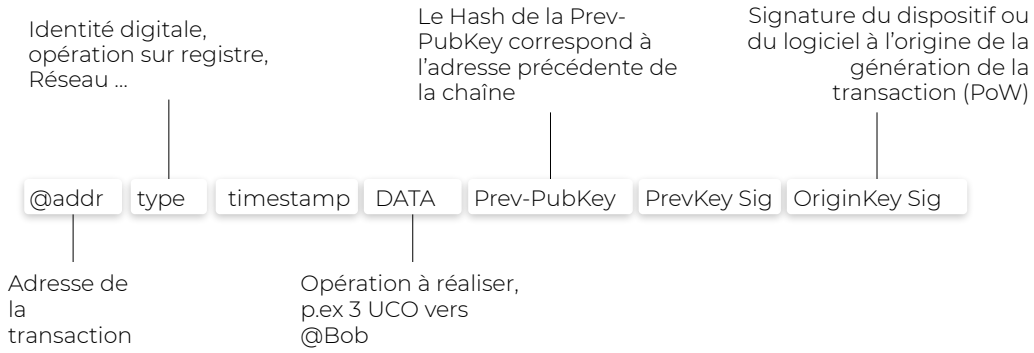


Figure 1.2: Transaction en instance

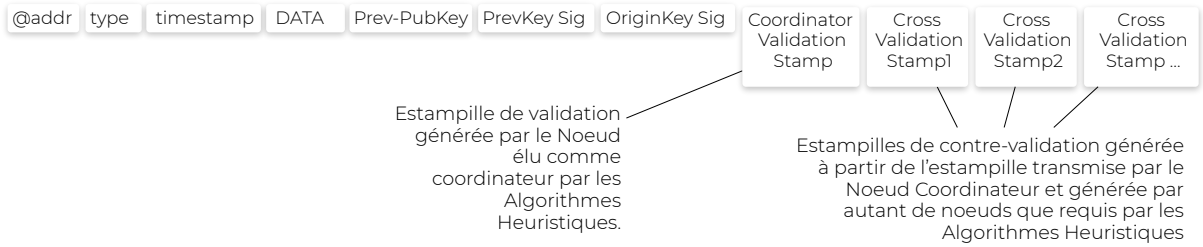


Figure 1.3: Transaction Validée

Les principes des Chaînes de Transactions sont les suivants:

Principe 1. Transaction en instance est une transaction qui ne dispose pas des preuves de validation du réseau. Cette transaction pour être traitée par le réseau doit disposer des informations suivantes (Figure 1.2) :

- `@addr` une adresse non utilisée et qui correspond au hash (empreinte) de la clé publique de la transaction qui lui succédera dans la chaîne.
- `type` un type définissant le rôle fonctionnel de la transaction en instance.
- `timestamp` la date de la génération de la transaction.
- `DATA` une zone de données contenant l'ensemble des opérations à effectuer (opération sur un des registres, tâches programmées, smart-contracts,

opérations sur une identité, etc.)

Prev-PubKey Clé publique associée à la transaction précédente (l'adresse de la transaction précédente étant le hash de cette clé publique)

PrevKey Sig Signature à partir de la clé privée associée à la clé publique mentionnée permettant de prouver la possession de la clé privée, le chaînage des transactions et le contenu de la transaction (adresse, type, timestamp et la zone de données "DATA") .

OriginKey Sig Signature à partir de la clé privée associée au dispositif ou logiciel à partir duquel la transaction en instance a été générée. Cette signature est utilisée pour la preuve de travail des noeuds (1.2.4).

Principe 2. Transaction validée est une transaction en instance complétée des preuves de validation requises par les Algorithmes Heuristiques. Ces preuves de validation sont définies de la façon suivante (Figure 1.3):

Validation Stamp généré par le noeud coordinateur élu et contenant :

Preuves d'intégrité chaînage avec les transactions précédentes

PoW résultat de la preuve de travail

Opérations sur les Registres (Ledgers) contient l'ensemble des opérations sur les registres qui seront prises en compte par le réseau :

- les mouvements de registre mentionnés dans la transaction en instance DATA/Ledgers accompagnés de la signature de l'émetteur.
- la liste des transactions non dépensées (UTXO) de la chaîne de l'émetteur sur l'adresse de la nouvelle transaction.
- les mouvements de registre à destination des noeuds qui ont participé à la validation et à la récupération des données (paiement effectif des noeuds d'accueil, coordinateur, de contre-validation et des noeuds sur lesquels ont été téléchargées les données) DATA/Ledgers/Fee – les frais *(Fee)* pouvant être mentionnés de manière explicite ou non mentionnée (dans le dernier cas, les frais seront calculés par le noeud coordinateur).

Signature cryptographique de l'Estampille de Validation du noeud coordinateur, la clé publique étant déjà mentionnée dans la liste des bénéficiaires des frais.

Counter Validation Stamps pour être considérée comme valide, l'Estampille de Validation *(Validation Stamp)* doit être accompagnée d'autant d'Estampilles de Contre-Validation *(Counter Validation Stamps)* que requis par les Algorithmes Heuristiques. Les Estampilles de Contre-Validation sont les signatures de l'Estampille de Validation par chacun des noeuds de contre-validation (en cas d'incohérence ou de désaccord celle-ci contiendra la liste des incohérences relevées le tout signé par le noeud de contre-validation).

Principe 3. Chaînes de transactions Chaque transaction validée est stockée sous forme d'une chaîne qui ne peut être mise à jour qu'à partir de la dernière transaction validée de la chaîne.

Principe 4. Chaînes de transactions Réseau Les Chaînes de transactions Réseau sont des chaînes identiques aux autres chaînes de transactions, mais utilisent des algorithmes de validation et de réplications différents. Les chaînes de transactions réseau sont répliquées sur l'ensemble des noeuds.

Principe 5. Algorithmes Heuristiques sont l'ensemble des algorithmes, logiciels et fichiers de paramétrages stockés sous forme de chaînes et appartenant aux "Chaînes de transactions Réseau". (Élection des noeuds de validation et de stockage, mise à jour des clés des noeuds, Secrets Partagés, Interpréteurs, Module de Prédiction, Gestion des Chaînes de balisage, etc.)

Principe 6. Principes de Réplication Après la validation d'une transaction, les noeuds de validation seront en charge d'appliquer le processus de réplication suivant:

- L'ensemble des transactions associées à une même chaîne doit être répliqué sur les noeuds de stockage associés à l'adresse de la dernière transaction validée de la chaîne. Les chaînes de transaction associées au fonctionnement du réseau doivent être en intégralité répliquées sur l'ensemble des noeuds (Chaînes des noeuds, des algorithmes, des logiciels décentralisés, etc.).
- Chaque transaction validée doit être répliquée sur les noeuds de stockage associés aux adresses des transactions associées (input/output).
- Chaque adresse de transaction doit être répliquée sur les noeuds de stockage du sous-ensemble d'adresses associé *(Chaînes de balisage)*.

Principe 7. Principes de Synchronisation Chaque noeud dès son arrivée sur le réseau doit en permanence synchroniser les chaînes de transactions pour lesquelles il a été élu comme noeud de stockage à partir des Algorithmes Heuristiques et à partir des adresses des dernières transactions des chaînes, chaque noeud devra également synchroniser les transactions à destination de la chaînes *(notamment les sorties non dépensées / unspent output)* tant que celles-ci ne seront pas intégrées dans une estampille de validation (i.e. tant que celles-ci n'auront pas été consommées dans une nouvelle transaction en tant que sorties dépensées – *Réparation et Reconfiguration Automatique du Réseau*).

Principe 8. Authentification des noeuds Pour participer au réseau, chaque noeud doit être autorisé et authentifié par l'intermédiaire d'une chaîne valide de transactions. Dès lors, dès que sa clé sera périmée ou qu'une information nécessaire au réseau sera modifiée (IP, port, protocole) le noeud devra régénérer une nouvelle transaction sur sa chaîne.

Principe 9. Gestion des Anomalies sur consensus En cas d'incohérence ou de désaccord sur la validité d'une "transaction en cours de validation", n'importe quel noeud élu associé au processus de validation ou de stockage pourra alors soumettre cette "anomalie potentielle" à un pool de validation (via un processus d'élection public) qui devra statuer sur la validité de la transaction et le cas échéant, sur l'origine de l'anomalie (problème réseau, ambigüité de la transaction ou malveillance caractérisée)

Principe 10. Gestion des noeuds "malhonnêtes" Si l'origine de l'anomalie associée à l'invalidité d'une "transaction en cours de validation" est statuée comme "malveillance caractérisée" alors les noeuds associés à la validation de "transaction validée" statuée comme invalide seront bannis du réseau.

Principe 11. Modèle UTXO Une transaction validée ne peut pas changer d'état (stateless - modèle UTXO[1]), ce qui signifie qu'il n'y a pas de réalité en dehors des transactions validées, chaque utilisateur peut ainsi vérifier la validité des transactions précédentes.

[1] Seule une sortie de transaction non dépensée (UTXO) peut être dépensée/utilisée comme entrée d'une nouvelle transaction.

Definition 1. Pour assurer la propriété d'atomicité de la validation d'une transaction en instance avec une marge d'erreur de 10^{-9} et en considérant en permanence que potentiellement 90% du réseau pourrait être "malicieux", le nombre de noeuds "n_v" participants à la validation de la transaction et à la réplication de la chaîne devra toujours satisfaire la propriété suivante :

$$\forall\, n_v \in \mathbb{N},\ \sum_{k=1}^{0.1\times n} \frac{\binom{0.1\times n}{k} \times \binom{0.9\times n}{n_v - k}}{\binom{n}{n_v}} \approx 10^{-9} \tag{1.1}$$

Où:

n_v : est le nombre de noeuds impliqués dans la vérification ;
n : est le nombre total de noeuds du réseau.

Une transaction en instance devra recevoir un accord unanime, positif et concordant de l'ensemble des noeuds n_v pour être considérée comme validée.

Application: *(Définition 1) pour un réseau constitué de 100 000 noeuds (n = 100 000) alors pour qu'une transaction en instance puisse être validée il lui faudra 197 confirmations réparties entre les noeuds de validation et les noeuds de stockage (n_v=197) cf. Figure 1.18*

Par voie de conséquence:

Remarque 1. (Principe 3) L'adresse de n'importe quelle transaction d'une même chaîne pourra être utilisée comme adresse de destination, il n'est pas nécessaire de préciser la dernière transaction de la chaîne. *(Les noeuds répliqueront automatiquement la transaction sur le pool de stockage associé à la dernière transaction de la chaîne).*

Remarque 2. (Principe 3) En termes de sécurité, une fois la clé publique révélée, elle est considérée comme expirée, seul le hash de la clé publique de la transaction suivante @addr est annoncé, ce qui permet de garder la clé publique secrète jusqu'à la prochaine transaction sur cette même chaîne (corrigeant ainsi le problème de l'utilisation répétée d'une même clé ECDSA) .

Remarque 3. (Principe 3 – 6) Lors de la mise à jour d'une chaîne de transaction par l'intermédiaire d'une nouvelle transaction validée en fin de chaîne. Les noeuds qui étaient en charge du stockage de la chaîne avant l'arrivée de la transaction pourront alors arrêter de synchroniser les données de cette chaîne, les nouveaux noeuds de stockage de référence étant désormais les noeuds de stockage calculés à partir de l'adresse de cette dernière transaction (cf. 1.4.1 – 1.10).

Remarque 4. (Principes 3 – 6) En termes d'ordonnancement, cela signifie que toutes les sorties d'une chaîne donnée doivent être sérialisées ("Alice vers {Bob, Tom, etc.}", puis "Alice vers {Michelle}") mais qu'un nombre illimité de transactions peuvent être à destination d'une même transaction (Bob vers {Alice} au même instant que Tom vers {Alice}, Michelle vers {Alice}, etc.) – cela signifie qu'une nouvelle transaction sur une chaîne de transaction spécifique doit attendre la fin de la validation de la précédente (1 à 3 secondes), mais qu'un nombre quasiment illimité de transactions/seconde peuvent être dans le même temps à destination de cette même chaîne de transaction.

Remarque 5. (Principe 2 – 11) La liste des sorties non dépensées (unspent outputs) n'a pas besoin d'être précisée par l'émetteur de la transaction, l'ensemble des sorties non dépensées ou non utilisées sont réintégrées directement dans la dernière transaction de la chaîne de l'émetteur par le noeud coordinateur et par l'intermédiaire de l'Estampille de Validation – permettant ainsi aux noeuds de retrouver directement l'ensemble des sorties non dépensées sur les noeuds de stockage de la dernière transaction et à l'utilisateur d'éviter la coûteuse phase de recherche des sorties non dépensées.

Remarque 6. (Principe 11) Comme pour le protocole du réseau Bitcoin, le réseau Uniris utilise pour des raisons de sécurité et d'irrepudiabilité des transactions, le principe du modèle UTXO, ce qui signifie qu'il n'y a pas de réalité en dehors des transactions validées. Ainsi et contrairement aux smart-contracts tels que ceux fonctionnant à l'intérieur de machines virtuelles (Ethereum, EOS, etc.) les smart-contracts Uniris ne peuvent pas changer d'état une fois validés sans pour autant en réduire le nombre de cas d'usage (cf. Saison 2 : Smart-Contracts)

Remarque 7. (Principes 3,4 et 8) La figure 1.4 ci-dessus représente quatre types de transactions chaînées (une chaîne associée à l'identité d'un noeud du réseau, une chaîne associée à l'identité d'une personne, une chaîne associée à un portefeuille UCO (Uniris Coin) et une chaîne de contrat intelligent). La première transaction de la chaîne (genesis) est largement utilisée comme référence de la chaîne et la dernière transaction est la seule pouvant être mise à jour par une nouvelle transaction (l'adresse "@node0-3" annonçant le hash de la clé publique qui permettra de vérifier la possession de la clé privée, autorisant ainsi la mise à jour de la chaîne).

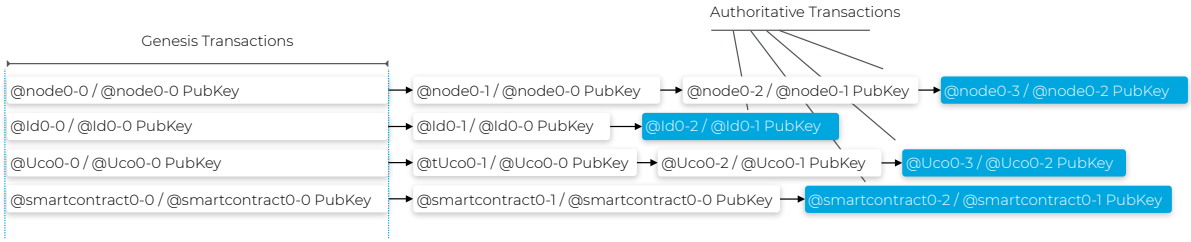
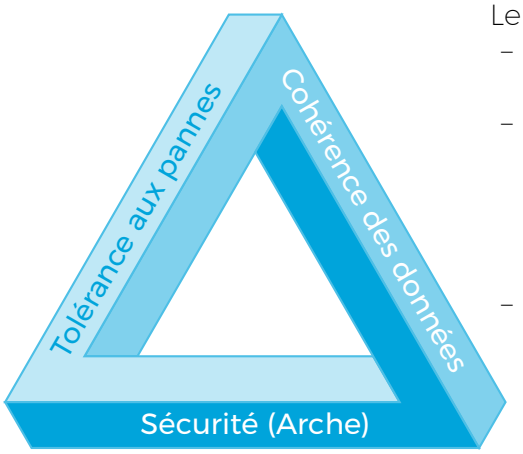


Figure 1.4: Exemples de chaînes de transactions

1.2 L'Arche : Consensus sans compromis permettant le million de txn/sec



Le réseau Uniris s'appuie sur trois propriétés :

- Sécurité : basée sur le consensus ARCHE, chaque transaction est validée de façon atomique.
- Cohérence des données : basée sur les algorithmes heuristiques de stockage qui permettent de garantir l'accès à la dernière écriture mais aussi une disponibilité maximum des données.
- Tolérance aux pannes : à partir des algorithmes heuristiques de validation qui permettent aux noeuds de travailler de façon autonome même en cas de désastre réseau.

Le **Consensus ARCHE** est une nouvelle génération de Consensus « Atomic Rotating Commitment Heuristic (ARCH ou ARCHE) » en français « Validation Atomique obtenue par élection heuristique tournante » - en décomposant chacune des notions :

Validation Atomique *(Atomic Commitment)* est la forme de Consensus « absolue » qui implique 100% de réponses concordantes et positives ou le refus de la validation de la transaction.

Heuristique est l'ensemble des algorithmes, des logiciels et des paramètres qui gèrent l'intégralité du réseau permettant par exemple au réseau d'élire de façon décentralisée et coordonnée les noeuds en charge de la validation et du stockage des chaînes de transactions.

Rotating le réseau étant entièrement distribué (aucun rôle central ou privilégié), les noeuds élus pour chaque opération changent en permanence de sorte qu'aucun noeud ne peut prédire avant l'arrivée de la transaction quels noeuds seront élus.

Le réseau Uniris s'appuie sur les propriétés des lois de répartition hypergéométrique qui à partir d'une élection imprédictible et d'un consensus formel permet d'obtenir avec certitude (99,9999999%) la même réponse en interrogeant 197 noeuds qu'en en interrogeant 100 000 (*Résilience du réseau & détection des opérations malveillantes*). En d'autres termes, cette loi mathématique permet d'obtenir un consensus formel global à partir d'une infime partie des noeuds - cette propriété entre ainsi dans la notion *Heuristique* largement utilisée sur l'ensemble du réseau. Le risque sur la disponibilité associée est assuré par une gestion stricte des noeuds perturbateurs – *Principes 8, 10 et 11*, qui après investigation sur l'origine du désaccord bannira tout noeud perturbateur.

Dans la suite de cette section seront définis :

1.2.1 Algorithmes Heuristiques

Les Algorithmes Heuristiques sont indifféremment constitués de logiciels (interpréteurs, librairies, etc.) d'algorithmes et de fichier de paramétrage *(Principe 5)*.

Pour fournir un réseau de confiance et entièrement décentralisé se pose le problème de la confiance sur les couches logicielles et algorithmiques du réseau. En effet, comment parvenir à une validation atomique d'une transaction si les noeuds n'utilisent pas les mêmes algorithmes ni les mêmes interpréteurs et par conséquent comment différencier un noeud malveillant d'un noeud simplement obsolète. De la même façon, il n'est pas possible d'assurer la réalité d'une gouvernance décentralisée si les noeuds sont autonomes pour choisir la couche logicielle/algorithmique qu'ils vont utiliser.

Pour résoudre ces problèmes, les Algorithmes Heuristiques et les Logiciels sont stockés de façon décentralisée sous forme de chaînes de transactions ou plus précisément sous forme de chaînes de smart-contracts. La gouvernance et le fonctionnement des smart-contracts associés étant intégralement paramétrables : auto-déclenchables, protégés par Contraintes Récursives Héritées *(Saison 2)* et sous le contrôle de la communauté *(Saison 3)*.

L'exemple de la figure 1.5 ci-dessous représente un module stocké avec son code source dans la zone DATA/Content et contenant des Contraintes Récursives Héritée.

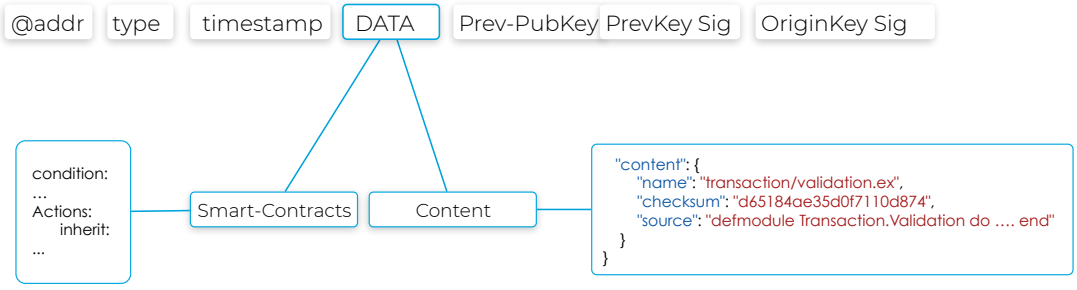


Figure 1.5: Gestion de versions décentralisée des logiciels et Algorithmes Heuristiques

1.2.2 Élection Globale, Imprédictible et Reproductible

Pour réaliser une élection imprédictible, globale, mais exécutée localement, vérifiable et reproductible des noeuds en charge de validation d'une transaction, les Algorithmes Heuristiques d'élection des noeuds de validation se basent sur :

- sur un élément imprédictible: le hash du contenu de la transaction,
- sur un élément connu uniquement des nœuds autorisés: le *Daily Nonce* inclus dans les Secrets Partagés des noeuds et renouvelé quotidiennement *(Figure 1.19)*,
- sur un élément difficilement prédictible des noeuds : la dernière clé publique de la chaîne des noeuds : *Last Node Pubkey*,
- et sur le calcul des clés tournantes de validations : *Validation Rotating Keys*.

Le calcul des clés tournantes *(Validation Rotating Keys)* fonctionne de la façon suivante:

$$\text{RotatingKey}_{NodePubkey} = \text{Hash}(\textit{Last Node Pubkey, Daily Nonce, hash(transaction content)}) \quad (1.2)$$

La figure 1.6, ci-dessous, représente de façon schématique le fonctionnement des algorithmes pour obtenir une liste filtrée des noeuds de validation (Pool de validation):

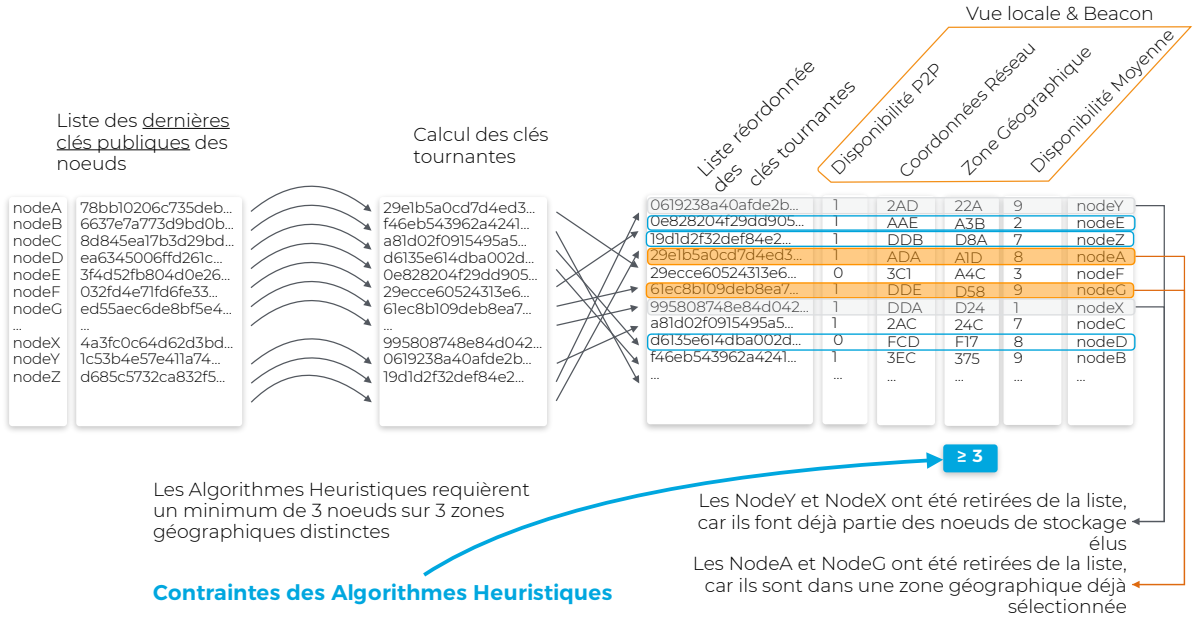


Figure 1.6: Election des Noeuds de Validation (Mining)

L'objectif du calcul des clés tournantes est de fournir une liste ordonnée imprédictible et reproductible de la liste des noeuds autorisés. L'ordonnancement ainsi obtenu permet à chacun des noeuds de retrouver de façon autonome et partagée la liste des noeuds qui seront en charge de la validation de la transaction. Cette liste est ensuite filtrée en fonction des contraintes des Algorithmes Heuristiques à partir des vues consolidées des noeuds constituées de:

- La vue du réseau locale du noeud *(Figure 1.17)*
- Mise à jour des Chaînes réseau répliquées sur l'ensemble des noeuds *(Principe 5)*
- Des informations fournies par les chaînes de balisage *(Figure 1.15)*
- Des contraintes ajoutées par le module de prédiction *(Figure 1.21)*

Cette validation étant ponctuelle, les critères prioritaires pris en compte sont la disponibilité du noeud au moment de l'élection, la zone géographique, le nombre de noeuds à élire et la liste des noeuds de stockage qui participeront dans une deuxième phase à la validation de la transaction (1.3.2).

1.2.3 Processus de validation d'une transaction

Le schéma ci-dessous représente de manière simplifiée le processus de validation d'une transaction de transfert de UCO (Uniris Coin):

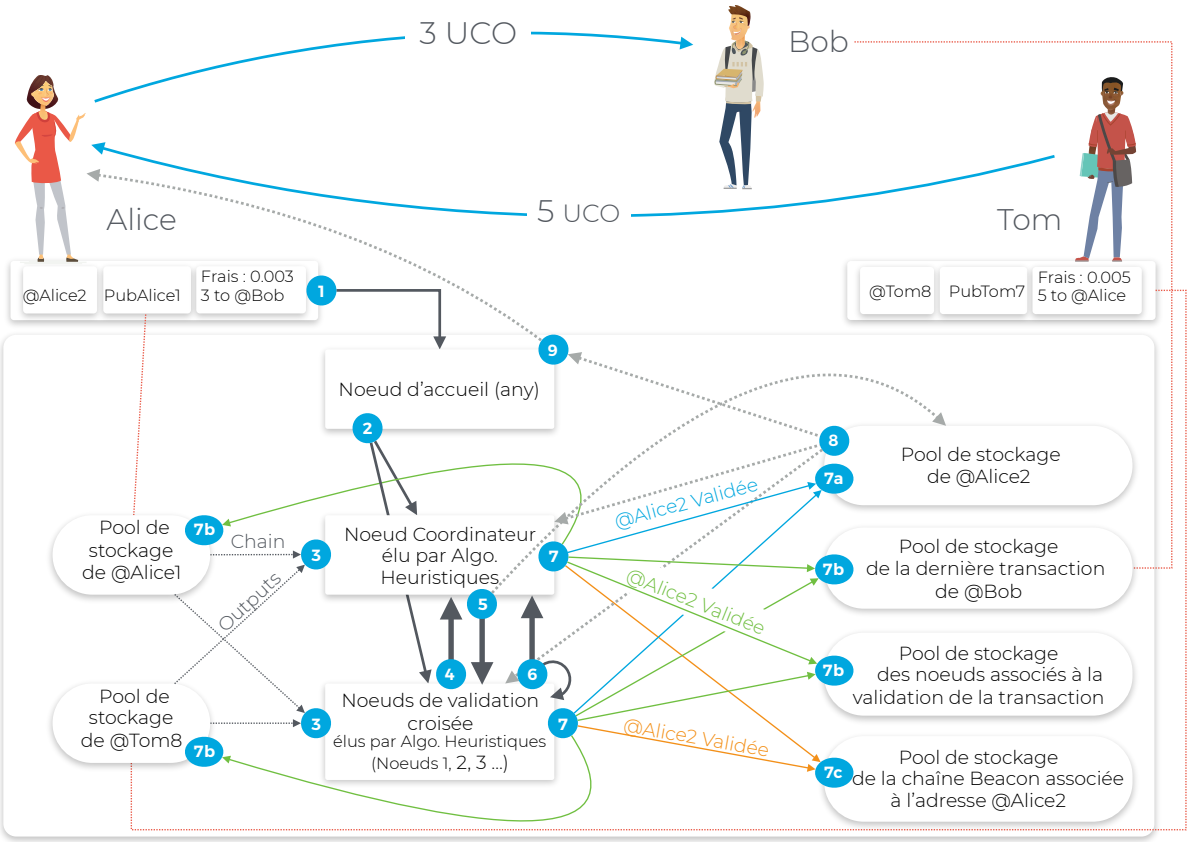


Figure 1.7: Processus de validation d'une transaction

1 Alice transmet la transaction @Alice2 à un n'importe quel nœud du réseau (nœud d'accueil) via la liste des "Seeds" (1.4.3)

2 le nœud d'accueil calcule les nœuds de validation (coordinateur et contre-validation à l'aide des Algorithmes Heuristiques (1.2.2), puis transmet la transaction aux différents nœuds élus pour qu'ils puissent commencer le travail préliminaire de validation.

3 les noeuds élus (coordinateurs et noeuds de contre-validation) récupèrent l'ensemble des transactions nécessaires à la validation sur les noeuds de stockage les plus proches (1.3.3) : la chaîne complète sur les noeuds de stockage de la transaction @Alice1 (1.3.2), et toutes les transactions liées aux entrées non dépensées (UTXO) @Tom8 ... sur les pools de stockage respectifs.

4 une fois, le contexte de la transaction reconstitué, les noeuds de contre-validation transmettent au noeud coordinateur la liste des noeuds de stockage utilisés pour récupérer les données et leurs vues sur la disponibilité des noeuds de validation et de stockage.

5 le noeud Coordinateur, après avoir : reconstitué le contexte de la transaction (chaîne, entrées, sortie), calculé la preuve de travail (PoW), calculé l'arbre de réplication, calculé les opérations sur les registres et signé l'estampille de validation *(Validation Stamp – Principe 2)* la transmet aux noeuds de contre-validation.

6 le contenu de l'Estampille de Validation vérifié, chaque noeud de contre-validation transmettra une "Estampille de Contre-Validation" au coordinateur et aux autres noeuds de contre-validation (Principe 2).

7 l'ensemble des Estampilles de Contre-Validation reçues chaque noeud de validation pourra alors démarrer le processus de réplication de la transaction validée, tel que définit par le noeud Coordinateur:

- 7a le pool de stockage de la chaîne @Alice2 va reconstituer le contexte de la transaction, si tout est conforme l'intégrer dans sa base locale et 8 notifier les noeuds de validation et d'accueil de la finalisation de la réplication.
- 7b les pools de stockage associés à @Bob et aux noeuds impliqués vont vérifier la conformité de la transaction et l'intégrer dans leurs bases.
- 7c le pool de stockage associé à la Chaîne Beacon (1.4.1) va vérifier la conformité de la transaction et intégrer le timestamp et les adresses concernées @Alice1, @Alice2, @Bob dans la chaîne Beacon.

9 le noeud d'accueil notifiera l'utilisateur de la progression de la validation de la transaction.

1.2.4 La Preuve de Travail

Le réseau Bitcoin et les réseaux basés sur le principe de la preuve de travail (PoW) permettent d'assurer dans le même temps une élection imprédictible et pseudo-aléatoire des noeuds en charge de la validation des blocs (minage), mais aussi d'assurer que le réseau reste synchrone, c'est-à-dire que chaque noeud va avoir le temps de récupérer les blocs précédents, grâce au temps nécessaire au calcul de la preuve de travail (10min).

Sur le réseau Uniris:

- L'élection des noeuds de validation imprédictible et pseudo-aléatoire est assurée par les Algorithmes Heuristiques de validation *(Figure 1.6)*.
- La synchronisation des données est assurée par les Algorithmes Heuristique de réplication des données qui permettent en tout temps de connaître de façon ordonnée les noeuds en charge du stockage d'une chaîne de transaction donnée *(Figure 1.10)*.
- Le mécanisme de preuve de travail est néanmoins conservé pour assurer l'authentification des dispositifs à l'origine d'une transaction permettant par exemple d'interdire toute transaction même en cas de vol de clé *(Saison 4)*, mais aussi de permettre d'imposer sur la validation d'une transaction un niveau de sécurité minimum (un utilisateur pourra consulter son solde sur un smartphone spécifique, mais ne pourra réaliser de transaction que sur un autre, lors d'un vote l'organisateur pourra s'assurer de l'identité réelle biométrique d'un votant par l'identification cryptographique associée à un dispositif particulier)

La *Preuve de Travail* consiste à trouver la clé publique associée à la signature du périphérique à l'origine de la transaction OriginKey Sig. Cela implique une recherche sur un ensemble fini de clés publiques garantissant que le périphérique est autorisé de façon certaine à générer la transaction et que, dans le même temps, les noeuds capables de trouver cette clé sont bien autorisés et connaissent les Secrets Partagés des Noeuds *(Figure 1.19)* – dans le cas contraire, la recherche par force brute pourrait prendre un temps considérable.

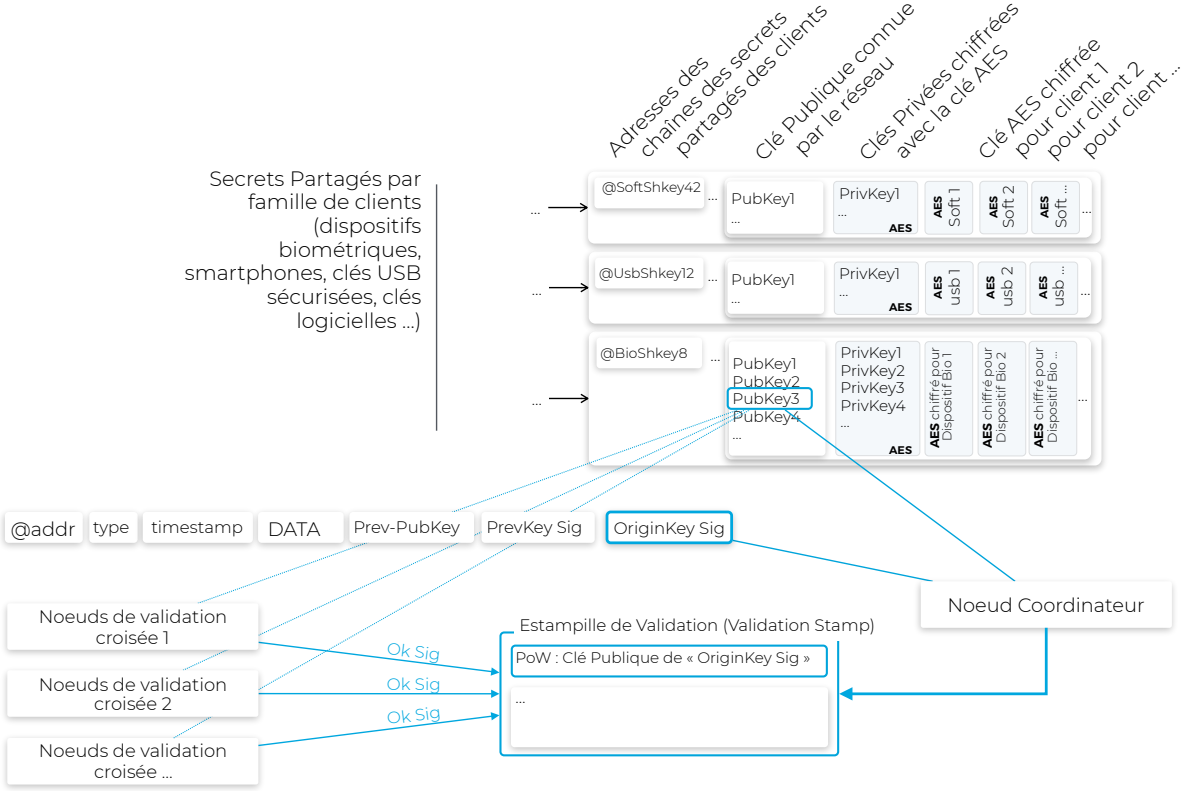


Figure 1.8: Preuve de Travail

Ce mécanisme permet de prendre en charge un nombre non limité de dispositifs tout en garantissant la confidentialité de l'utilisateur par rapport au dispositif utilisé, la clé privée utilisée par un dispositif étant la dernière clé partagée entre les appareils de la même famille (clés logicielles, clés de smartphone, clés de cartes à puce, etc.). Tout comme les noeuds du réseau, chaque dispositif disposera de sa propre chaîne de transaction lui permettant de mettre à jour ses clés.

Le mécanisme de renouvellement des clés partagées des dispositifs étant réalisé de la même manière que le renouvèlement des clés partagées des noeuds *(Figure 1.19)* le réseau pourra bannir un dispositif particulier (la clé partagée des dispositifs ne sera plus chiffrée pour le dispositif banni).

1.2.5 Protection contre la Double et les Multiples Dépenses

Le problème de la double dépense *(Double Spending)* est un aspect essentiel des réseaux décentralisés, car au-delà du problème financier, il pose le problème de la synchronisation des opérations à partir de noeuds autonomes ne disposant d'aucun serveur central capable d'ordonnancer les opérations. Chaque noeud doit donc pouvoir localement assurer la cohérence du réseau et de façon autonome.

Pour résoudre le problème de la désynchronisation du réseau (cas non frauduleux), le réseau Uniris utilise:

- Le mécanisme des chaînes de transactions *(Principe 3)* qui permet de sérialiser la génération d'une nouvelle transaction sur une chaîne – la génération d'une transaction est donc toujours synchrone par rapport à sa chaîne. D'autant plus pour les chaînes utilisant le mécanisme de dérivation de clés, pour lesquelles un fork ne pourrait techniquement pas être stocké et devra donc être résolu avant son stockage.
- les Algorithmes Heuristiques de stockage permettant en tout temps de connaître localement, l'ordre de réplication et donc l'ordre de fraîcheur de chaque transaction à partir de son adresse *(Figure 1.10)*. Ce mécanisme utilisé aussi bien pour les chaînes de transaction que pour le stockage des algorithmes et des logiciels *(Figure 1.5)* permet de garantir que chaque noeud dispose, de façon autonome, des moyens nécessaires lui permettant d'utiliser des données synchrones.
- Enfin, le mécanisme de validation *(Figure 1.7)* utilise des notifications croisées (le pool de validation récupère avant et notifie après validation, les pools de stockage associés à la chaîne et aux sorties non dépensées/utilisées (unspent Outputs) permettant ainsi d'ordonnancer et de notifier les pools de validation en cas de double demande.

Pour résoudre le problème de la double dépense frauduleuse (tentative de fraude organisée par le pool de validation), le réseau Uniris se base :

- Sur la propriété de la distribution hypergéométrique *(Figure 1.18)* permettant de garantir que même avec 90% de noeuds malhonnêtes, le risque qu'un noeud honnête ne puisse pas détecter une opération frauduleuse est seulement de 10^{-9} soit une chance sur un milliard *(Définition 1)*.
- Si un cas de fraude est détecté par n'importe quel noeud de réplication *(Principe 9)*, un processus d'investigation public et décentralisé sera alors initialisé pour statuer sur le caractère frauduleux de la transaction validée. À l'issue de cette investigation, les noeuds considérés comme malicieux seront bannis du réseau *(Principe 10)*.

Au-delà, du problème de la double dépense, le mécanisme des chaînes de transactions évitent également le problème des *multiples paiements simultanés* - ainsi si un élément perturbe le processus de paiement - un utilisateur peut envoyer un nombre illimité de tentatives (en utilisant la même adresse de transaction précédente) et pourra être certain qu'une seule transaction (paiement) sera effectuée.

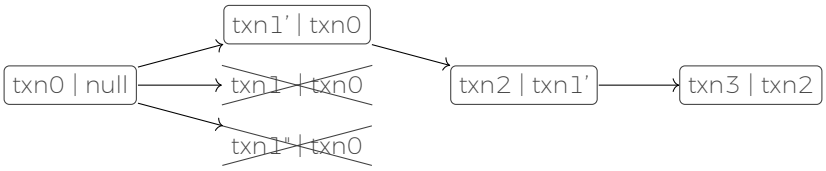


Figure 1.9: Gestion de la double dépense

1.3 Une Capacité de Stockage Décuplée et Geo-Sécurisée

Quelle que soit la technologie de réseau distribué, les défis sont les suivants :

- Comment trouver une donnée spécifique sans interroger tout le réseau ?
- Comment s'assurer que la donnée est bien la plus à jour ?
- Comment garantir que les données ne seront pas perdues ou corrompues ?
- Comment s'assurer de la cohérence des données sur un réseau fragmenté ?
- Comment éviter une attaque si les nœuds en charge de données sont connus ?
- Comment fournir un réseau illimité sans créer une surcharge du réseau global ?

Pour répondre à ces contraintes, les couches d'autodécouverte et de synchronisation des données ont été complètement repensées pour permettre de garantir la disponibilité des données quelque soit la catastrophe naturelle ou réseau, de donner une priorité en termes de fraîcheur des données, de permettre de récupérer les données par le meilleur chemin réseau.

Dans la suite de cette section seront définis tour à tour:

1.3.1 Nombre de réplicas par Chaîne

La couche de stockage distribué est un maillon essentiel pour permettre au réseau de toujours stocker une quantité plus importante de données. Dans le réseau Uniris, les données sont stockées de façon fragmentée (sharding), c'est-à-dire qu'aucun des noeuds ne contient l'ensemble des données et que le réseau va donc pouvoir stocker une quantité linéairement plus importante de données à mesure que le nombre de noeuds va augmenter.

À l'exception des Chaînes de transaction du réseau qui sont stockées sur l'ensemble des noeuds *(Principe 4)*, la règle du nombre de réplicas par chaîne est prévue pour s'affiner dans le temps par l'intermédiaire du module de prédiction *(Figure 1.21)*. En première approximation le nombre de réplications par transaction devrait répondre à la formule suivante :

$$\forall\, n_r \in \mathbb{N},\ \sum_{k=1}^{n} \textit{disponibilité}(k) > 2^{(\log_{10} n + 5)} \tag{1.3}$$

Où:

n_r : Nombre de réplicas nécessaires;
n : Nombre total de noeuds du réseau;
disponibilité : Disponibilité moyenne d'un noeud sur le réseau.

Paradigme du réseau illimité: Le nombre de réplicas par chaîne de transactions (à partir de l'adresse de la dernière transaction des chaînes) sera basé :

1. En première implémentation sur la formule (1.3)
2. Pour ensuite réduire le nombre de réplicas à la distribution hypergéométrique ≈ 200
3. Pour enfin ne garder après trois transactions validées que les condensats de chaînes (en ne conservant que les preuves cryptographiques et les mouvements sur les registres pour les parties les plus anciennes). Le passage vers les condensats permet de gagner un facteur minimum de 10 sur la taille des chaînes stockées.

Le résultat en termes de capacité de stockage utile et en considérant que les noeuds disposent de 512Go de stockage utile donne les résultats suivants :

	1000 noeuds	10 000	100 000	1 000 000	Nombre de transactions			
1	2 To	10 To	50 To	250 To	6.10^9	32.10^9	161.10^9	806.10^9
2	2,56 To	25,6 To	256 To	2,56 Po	8.10^9	82.10^9	825.10^9	8.10^{12}
3	2,56 To	25,6 To	256 To	2,56 Po	82.10^9	8.10^{11}	8.10^{12}	82.10^{12}

Sans fragmentation des données (sharding) la taille maximale admissible (capacité utile) est limitée à la taille du plus petit disque des noeuds, à titre d'exemple, avec 512Go:

- le réseau Bitcoin peut stocker 830 millions de transactions (431 millions pour 266Go).
- le réseau Ethereum peut stocker 122 millions de transactions (489 millions ≈ 2 To).

Pour un nombre de noeuds équivalents (100 000), le réseau Uniris permet de stocker au minimum (Étape 1) et au maximum (Étape 3):

- entre 193,9 et 9638 fois le nombre de transactions du réseau Bitcoin.
- et entre 1319 et 65 573 fois le nombre de transaction du réseau Ethereum.

1.3.2 Élection des noeuds de stockage

Contrairement à l'élection ponctuelle des noeuds de validation, le calcul des noeuds de stockage associés à une chaîne de transaction est réalisé à chaque modification du réseau (déconnexion ou arrivée d'un noeud) et à chaque nouvelle transaction sur le réseau. Ce calcul effectué en quelques millisecondes permet à chacun des noeuds de façon autonome de savoir s'il doit ou non télécharger une chaîne de transaction ou s'il n'a plus besoin de la stocker *(Figure 1.13)*.

Pour réaliser cette élection, l'Algorithme Heuristique Rotatif de stockage est basé:

- sur l'adresse de la transaction (les chaînes de transactions étant stockées sur l'adresse associée à la dernière transaction de la chaîne – *Principe 8*).
- sur un élément stable connu uniquement des nœuds autorisés: le *Storage Nonce* inclu dans les Secrets Partagés des noeuds *(Figure 1.19)*.
- sur la première clé publique (genesis) de la chaîne de chacun des noeuds : *Genesis Node Pubkey*, la première clé publique des noeuds étant stable, les noeuds de stockage d'une chaîne de transactions donnée resteront constants sur un réseau constant.
- et sur le calcul de clés tournantes de stockage.

Le calcul des clés tournantes de stockage *(Storage Rotating Keys)* fonctionne de la façon suivante:

$$\text{Rotating}_{Pubkey} = \text{Hash}(\textit{Genesis Node Pubkey, Storage Nonce, hash(transaction address)}) \quad (1.4)$$

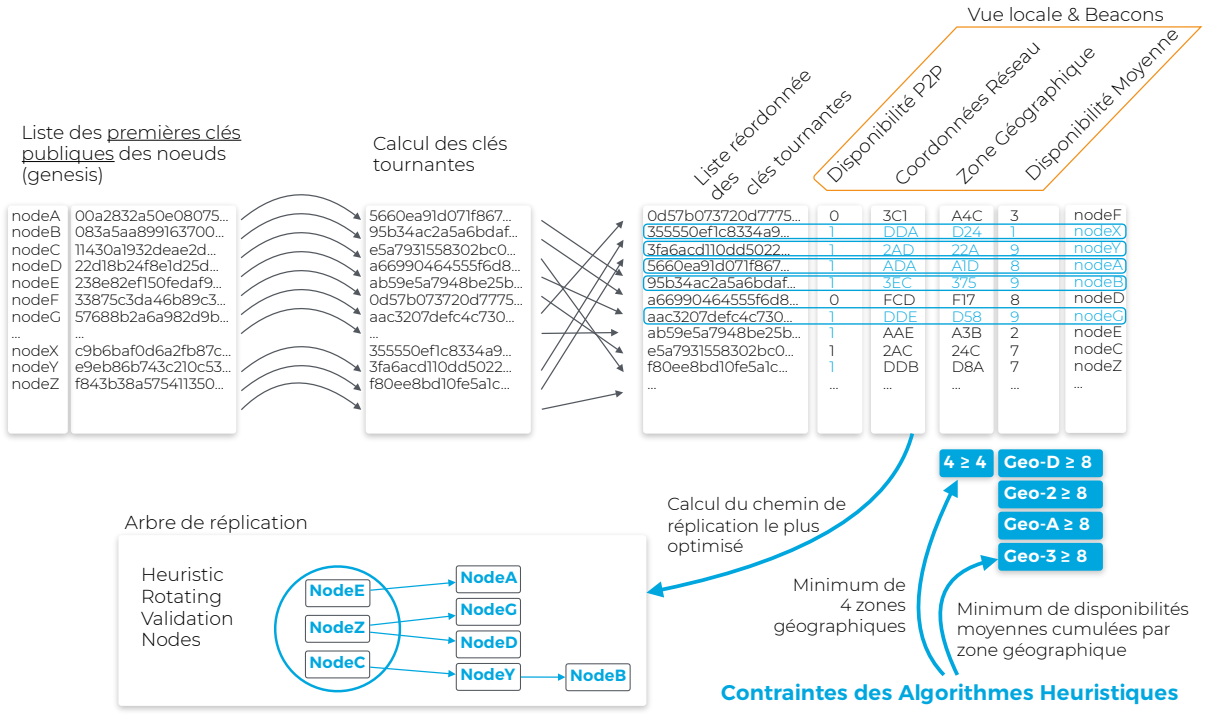


Figure 1.10: Processus d'élection des noeuds de stockage

La figure 1.10 ci-dessus représente de façon schématique les algorithmes mis en oeuvre lors du calcul des noeuds de stockage (Pool de stockage). L'objectif du calcul des clés tournantes de stockage est de permettre aux noeuds d'obtenir de façon autonome et commune une liste ordonnée et reproductible des noeuds de stockage à partir de l'adresse d'une transaction. Pour garantir une disponibilité et une sécurité maximum des données, cette liste est ensuite consolidée par les contraintes imposées par les Algo-

rithmes Heuristiques de stockage à partir des vues consolidées des noeuds constituée de:

- La vue du réseau locale du noeud *(Figure 1.17)*
- Mise à jour des Chaînes réseau répliquées sur l'ensemble des noeuds *(Principe 5)*
- Des informations fournies par les chaînes de balisage *(Figure 1.15)*
- Des contraintes ajoutées par le module de prédiction *(Figure 1.21)*

Les critères prioritaires pris en compte sont :

- La position géographique : permettant une continuité de service même en cas de catastrophes naturelles sur une ou plusieurs zones géographiques. Pour permettre de fiabiliser la position géographique d'un noeud, celle-ci est contextualisée par les coordonnées réseau calculées de façon globale par le mécanisme des chaînes de balisage *(Figure 1.16)*, mais également au moment de renouveler les clés associées aux noeuds.
- La disponibilité moyenne des noeuds dans une zone donnée : plutôt que de modifier la liste des noeuds élus, comme dans le cas de l'élection des noeuds de validation, le poids de l'élection d'un noeud de stockage sera pondéré par sa disponibilité, le but recherché sera donc la disponibilité cumulée (chaque zone géographique nécessitant un minimum de disponibilité cumulée D = 1+9 > 8 ...).
- Le risque identifié par le module de prédiction sur un ou plusieurs groupes de noeuds pour modifier la pondération de la disponibilité des noeuds.

La réalité d'une position géographique n'indiquant pas nécessairement la proximité du point de vue du réseau (latence, débit), l'arbre de réplication est calculé de façon globale à partir des vues locales et qualifiées des noeuds, au moment de la génération des estampilles de validation *Coordinator Validation & Counter Validation Stamps* permettant ainsi de répartir le travail de réplication à partir des chemins les plus optimisés.

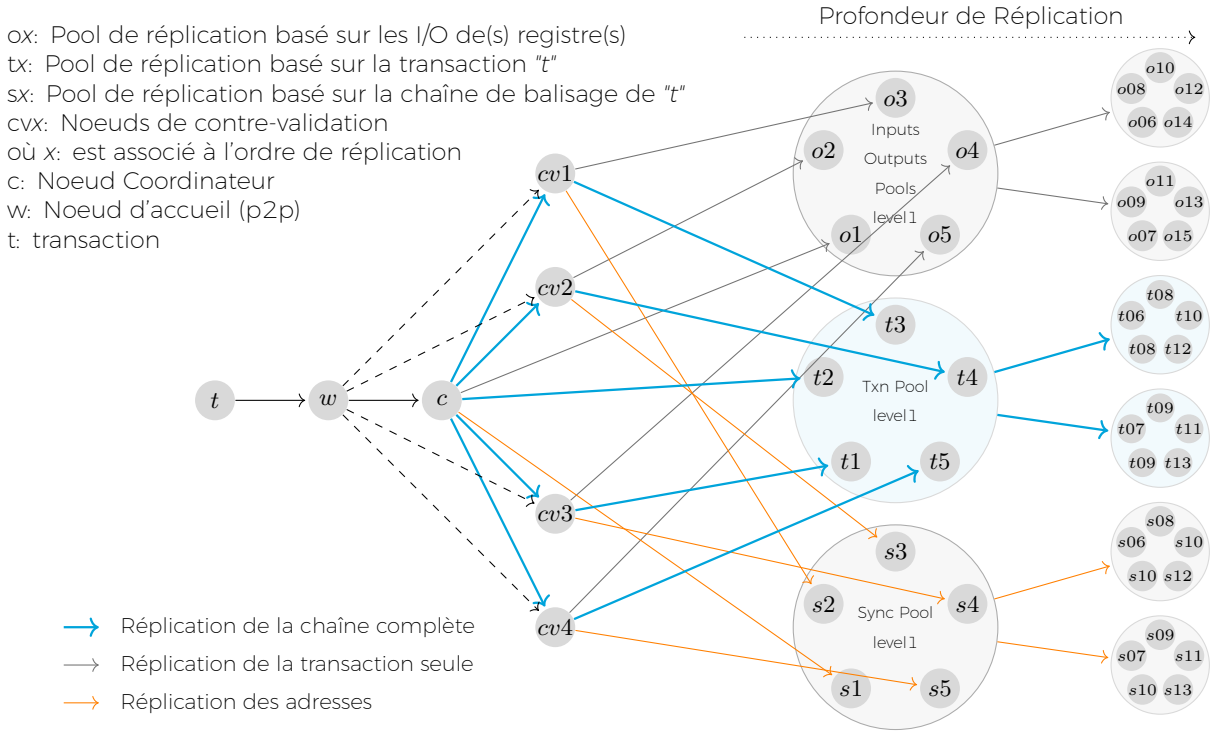


Figure 1.11: Vue fonctionnelle du processus de réplication d'une transaction validée

Comme représenté sur la figure 1.11 ci-dessus, une fois la transaction validée, les noeuds de validations démarreront le processus de réplication à partir de l'arbre de

réplication définit. Pour éviter les répliques manquées, chaque nœud attendra la confirmation du niveau de réplication suivant dont il est en charge avant d'arrêter son processus de réplication. De la même façon, si un nœud à l'intérieur de cette réplication ne confirme pas l'acquittement, le nœud au niveau supérieur prendra en charge cette réplication manquée pour assurer la continuité de la réplication.

Parce que chaque nœud a une confiance limitée dans les autres *(Risque de bannissement – Principe 10)*, la réplication se fera toujours à partir de la transaction validée complète *(Transaction en instance + Estampilles de validation + Estampilles de contre-validation)* permettant ainsi à chaque noeud de vérifier la validité de la transaction avant de la stocker.

Techniquement seule cette transaction validée sera transférée aux différents noeuds de stockage. En accord avec le *Principe 6* les noeuds de stockage devront réaliser les tâches suivantes :

- Les noeuds du pool de stockage "t*" de la transaction "t" auront pour charge de récupérer et de conserver les autres transactions de la chaîne et de vérifier de façon exhaustive la validité de la transaction et de sa chaîne associée. Après vérification, les noeuds du pool de stockage "t*" stockeront la chaîne complète dans leurs bases de données locales.
- Les noeuds des pools de stockage associés aux sorties non dépensées et aux destinataires stockeront directement la transaction après avoir vérifié sa validité de façon unitaire.
- les noeuds de la chaîne de balisage associée *(Formule 1.6)* stockeront la précédente adresse de la chaîne, l'adresse de la transaction, les adresses de destination et le timestamp pour les intégrer dans la chaîne de balisage associée.

1.3.3 Coordonnées Géographiques vs. Coordonnées Réseau

Le réseau Uniris se base sur deux types de coordonnées:

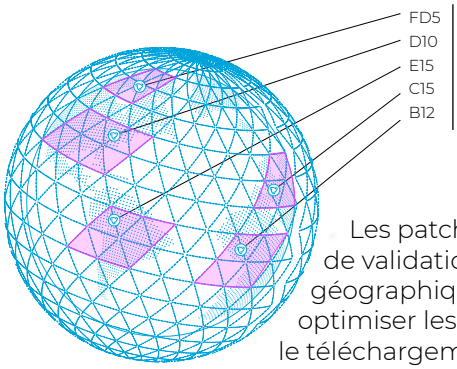
- les coordonnées géographiques: issues en partie de l'adresse IP publique du noeud.
- les coordonnées réseau: calculées à partir d'algorithmes modifiés proches de ceux publiés dans *Vivaldi: À Decentralized Network Coordinate System* [1] et directement intégrés dans la chaîne de balisage *(Beacons Chains – Figure 1.16)*.

Pour garantir la haute disponibilité et la consistance des données, la connaissance de la localisation réelle d'un noeud devient donc un critère essentiel, notamment pour éviter les pertes de données en cas de catastrophes naturelles. Pour ce faire, le réseau décompose en deux parties distinctes les coordonnées des noeuds:

- Les Coordonnées Réseau "Vivaldi" sont calculées de façon globale et quotidienne à travers les chaînes de balisage *(Beacons Chains – Figure 1.16)* et permettent, cycle après cycle, d'amélioré la précision des coordonnées réseaux des noeuds. La position d'un noeud est calculée à partir de sa latence (le temps minimum pour répondre à une transaction n'étant pas modifiable) et de son débit. Cette vue est utilisée pour choisir le meilleur chemin de réplication *(via l'arbre de réplication – Figures 1.10 et 1.11)*.
- Les Coordonnées Géographiques contextualisées d'un noeud : la contextualisation des coordonnées géographiques d'un noeud est réalisée au moment de la mise à jour de sa chaîne et par les noeuds en charge de la validation de cette mise à jour. Cette contextualisation est réalisée à partir des coordonnées déduites (GeoIP) ou annoncées et est pondérée par les calculs des Chaînes de balisage *(Figure 1.15)*. La mise à jour de ces données étant liée au renouvellement de clés des noeuds, elles seront ainsi renouvelées automatiquement de façon hebdomadaire et à chaque changement d'adresse IP.

L'objectif de ces contrôles supplémentaires n'est pas de mettre en défaut les noeuds, car il n'y a pas de punitions associées, mais d'assurer la meilleure répartition et terme de géographie et de disponibilité des réplications (1.3.2) à partir de données qualifiées pour assurer, in fine, une disponibilité maximum des données.

Comme représenté dans la figure ci-dessous, ces coordonnées réseaux et géographiques sont regroupées par zones *(patches)* permettant un calcul simplifié pour les noeuds (nombre de zones, arbres de réplication...). Les zones sont définies dans un multiplet de 12 bits représentant un arbre (ex: A5F).



Pour améliorer la confiance dans la position géographique des noeuds, la géolocalisation est vérifiée par recoupement à l'aide d'une version adaptée des algorithmes de Vivaldi.

Les patchs (zones) sont utilisés par les Algorithmes Heuristiques de validation et de stockage pour assurer une parfaite réplication géographique (disponibilité/sécurité des données) mais aussi pour optimiser les requêtes réseau. Lors du processus de validation/replication le téléchargement sera effectué à partir des noeuds les plus proches permettant ainsi d'éviter des transferts qui traverserai la planète alors même que la donnée serait déjà sur un noeud à proximité réseau.

Figure 1.12: Coordonnées Géographiques vs Coordonnées Réseau

1.3.4 Réparation et Reconfiguration Automatique du Réseau

Pour maintenir une capacité réseau illimitée, une infime quantité d'informations peut être répliquée sur tous les nœuds, et seulement celles qui ne nécessitent qu'une faible écriture (taille ou fréquence). Pour assurer la réparation et la reconfiguration automatiques des arbres de réplication, les nœuds utilisent les mêmes informations et formules que celles utilisées dans l'élection des noeuds de stockage *(Figure 1.10)*. Ce mécanisme permet à n'importe quel noeud de calculer en quelques millisecondes l'arbre de réplication associé à une adresse et donc de savoir s'il doit synchroniser une chaîne ou non *(Principe 6)*.

Ce processus est exécuté de façon autonome par chacun des noeuds à 3 moments spécifiques :

- Après une modification sur les Algorithmes Heuristiques (élection et contraintes sur les noeuds de stockage, mise à jour du module de prédiction – Figure 1.21)
- Après un changement au niveau des noeuds du réseau (disparition ou arrivée d'un nouveau noeud sur le réseau – Figure 1.14)
- Quotidiennement, après resynchronisation des données locales avec les dernières Chaînes de balisage *(Figure 1.15)* en reconstruisant l'historique des chaînes de transaction (nouvelle transaction sur une chaîne, etc.)

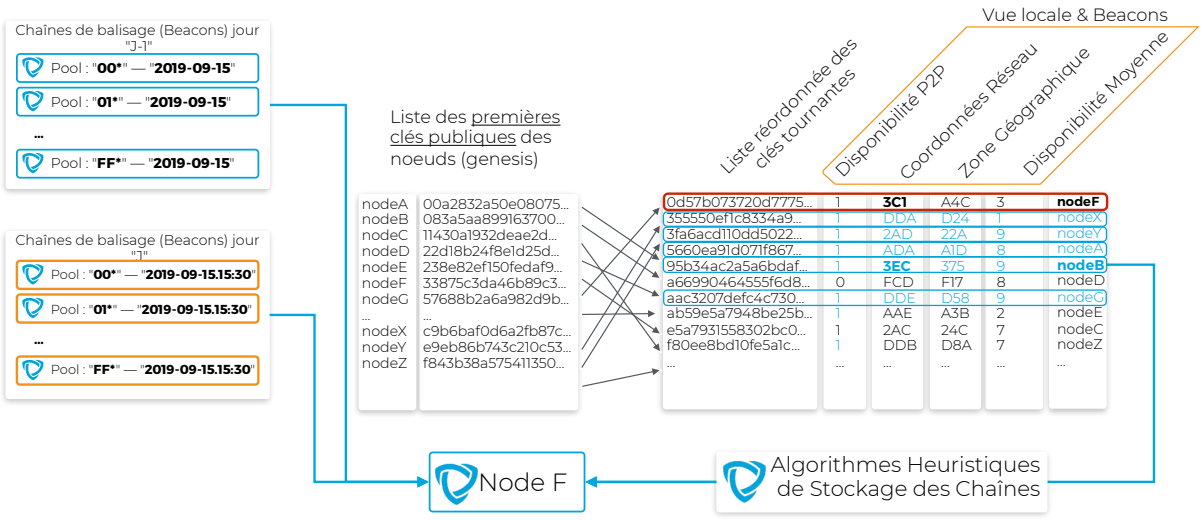


Figure 1.13: Mécanisme d'auto réparation du réseau

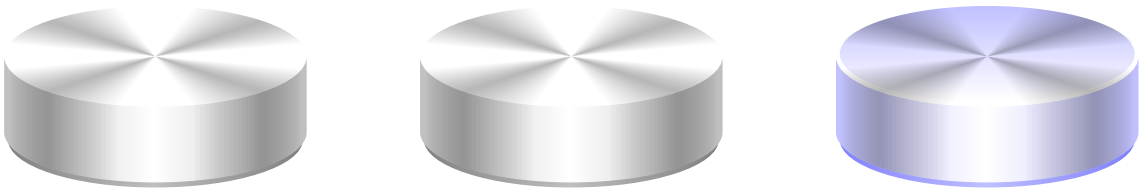
Dans l'exemple de la figure 1.13 ci-dessus, la NodeF a été déconnectée pendant 1 jour et commence le processus d'amorçage en téléchargeant les chaînes de balisage du jour "j-1" et la liste des transactions chaînées de la journée sur les pools de stockage associés *(Figure 1.15)*. Une fois le contexte téléchargé, la NodeF va reconstruire sa vue globale et historisée des chaînes de transactions (transactions, nœuds, mises à jour des algorithmes, etc.). En se basant sur cette vue et sur toutes les transactions qui n'ont pas déjà été téléchargées les jours précédents, la NodeF exécutera les Algorithmes Heuristiques de stockage pour connaitre sa position en tant que réplica par rapport aux exigences de chaque transaction. Une fois cette liste reconstituée la NodeF commencera à télécharger les transactions manquantes en se basant sur la vue du réseau grâce aux chaînes de balisage *(Beacons Chains)*, la NodeF choisira le NodeB pour télécharger la transaction, les coordonnées réseau de la NodeB étant les plus proches de la sienne.

1.3.5 Organisation des Données et des Registres Décentralisés

Pour assurer le meilleur équilibre entre disponibilité des données, intégrité globale des chaînes de transactions et des registres (ledgers), chaque type de données a une profondeur de réplication spécifique à l'intérieur du réseau. Les données souvent mises à jour ne pouvent pas être synchronisées partout sans générer de goulots d'étranglement réseau, très peu d'informations sont répliquées sur tous les nœuds, comme les chaînes de transaction du réseau *(Principe 4)* :

- les Chaînes de transactions des noeuds autorisés, utilisées pour les élections des nœuds de validation et réplication, mais également pour connaître leurs IP.
- les Chaînes de transactions des Algorithmes Heuristiques et des logiciels *(Figure 1.5)*.
- les Chaînes des Secrets Partagés des Nœuds et des périphériques *(Figure 1.19)*.
- la Chaîne du module de prédiction *(Figure 1.21)*.

La stratégie de profondeur de réplication est définie par les Algorithmes Heuristiques et exécutée directement par le logiciel décentralisé. Ce qui signifie que le réseau est autonome pour gérer les réplications sans utiliser d'autres mécanismes externes. En utilisant ces algorithmes, chaque nœud sera également capable de resynchroniser de manière autonome les données manquantes dont il est responsable (des nœuds nouveaux ou déconnectés changeront la position des autres nœuds dans l'ordre de réplication *(Figure 1.13)*.



Smart-Contracts &
Decentralized IDentity

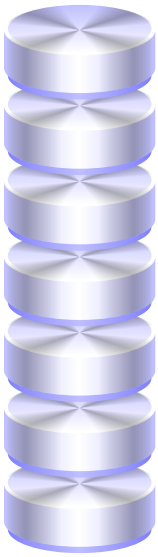
KeyChains

Local In-Memory DBs

Deux bases de données décentralisées constituent les références du réseau Uniris:

- La première : *Public Smart-Contracts and Decentralized Identity Ledgers* contient toutes les chaînes de transactions des utilisateurs (smart-contracts, mouvement sur les registres & les identités, etc.). Cette base de données est le cœur de la solution, car elle contient les chaînes des Algorithmes Heuristiques *(Pincipe 5)* et toutes les Chaînes Réseau (Chaînes des nœuds autorisés, clés partagées...) et par conséquent les règles de gouvernance.
- La seconde base, la *Keychains* gère les portefeuilles décentralisés contenant les clés privées des utilisateurs, groupes, organisations et des objets connectés. Les données sont gérées à l'aide d'algorithmes similaires au stockage des chaînes de transactions. Aussi, et pour garantir la sécurité du contenu des portefeuilles, l'accès est restreint à la preuve de possession de la clé privée (signature) - le fonctionnement de la Keychains est décrit en *(Saison 4)*.

Enfin, les nœuds utilisent des bases de données locales en mémoire, ces bases de données sont conçues pour pouvoir être reconstruites à chaque démarrage d'un noeud et permettent d'optimiser les calculs à réaliser par chacun des noeuds (vue réseau, liste des noeuds, vérifications des dépenses/contraintes, déclencheurs, etc.). La figure ci-dessous représente les principales bases de données locales en mémoire :



UCO Ledger	Optimized database focused on hosted UCO spent/unspent and conditions
NFT Ledger	Optimized database focused on Non-Financial Transactions (Emitter, spent/unspent & conditions)
MarketPlace Ledger	Optimized database focused on the stock management regarding MarketPlace hosted transactions
Triggers Ledger	Optimized database focused on smart-contract triggers for each hosted transaction
Network Ledger	Optimized database focused on Network requirements (Nodes, Sharedkeys, Prediction, etc.)
Pending&KO Ledger	Database listing all pending & KO transactions (to wait smart-contract countersignature ...)
P2P Ledger	Network discovery database (IP, availability, rating, geolocation, network coordinates,etc.)

Afin d'assurer la meilleure organisation possible des données, mais aussi d'utiliser le moins de mémoire possible, ces registres fonctionnent à l'aide d'un moteur de base de données NoSQL de type "orientée colonne" particulièrement adaptée au filtrage par champs, utilisés notamment pour les calculs associés aux Algorithmes Heuristiques.

1.4 Un Réseau Distribué Ouvert, Optimisé et Structuré

Il existe dans les réseaux distribués deux modes de communication : le mode Gossip dont les propriétés sont définies par la *connaissance des voisins sortants* c'est-à-dire que chaque noeud du réseau va découvrir les propriétés des autres noeuds en les interrogeant un à un généralement de façon aléatoire et le mode Broadcast [2] dont les propriétés sont définies par la *connaissance des voisins entrants* utilisent les connexions entrantes. Le réseau Uniris est un réseau hybride utilisant la Multidiffusion Supervisé *(Supervised Multicast)* se rapprochant plus des propriétés des réseaux de type "broadcast" et réunissant les propriétés suivantes :

Multidiffusion Supervisé *(Supervized Multicast)* la multidiffusion supervisée intervient dans trois processus du réseau :

- Processus de réplication des transactions : en capitalisant sur les informations des connexions entrantes et sortantes lors du processus de réplication *(Figure 1.17)*.
- Par le processus de mise à jour des *Chaînes de Transactions Réseau* (Principe 6) qui permet, par exemple, lors de la mise à jour de l'adresse IP d'un noeud de propager l'information sur l'ensemble du réseau par l'intermédiaire de la mise à jour de la chaîne associée au noeud.
- Par le processus décentralisé des chaînes de balisage *(Beacons Chains – Figure 1.16)* qui, toutes les 10min va prendre un instantané et tous les jours une synthèse de l'état de chacun des noeuds pour maintenir en permanence une vision globale et qualifiée du réseau.

Structuré et Authentifié chaque noeud connaît à tout moment la liste des noeuds autorisés à participer au réseau par l'intermédiaire des chaînes de transactions associées aux noeuds. Chaque connexion est authentifiée par la dernière clé publique de chacun des noeuds *(Principe 8)*.

Rémunéré chaque noeud est rémunéré en fonction de sa contribution au réseau *(Système d'Incitation)*, aussi bien pour les phases de validation que pour les phases de mise à disposition de l'information : un noeud n'est pas rémunéré pour répliquer une transaction, mais il le sera lorsqu'il mettra la transaction à disposition du réseau.

Prédictif et Adaptatif pour compenser le caractère imprédictible des noeuds sur un réseau constitué de noeuds quelconques, le réseau dispose d'un mécanisme de prédiction *(Figure 1.21)* qui lui permet d'apprendre et d'apporter des contre-mesures sur les anomalies détectées.

Sans Permission *(Permissionless)* tout noeud peut participer au réseau à partir du moment où il dispose d'un *Hardware Security Module - HSM*[1] et qu'ils n'intègrent pas d'élément lié à un précédent bannissement. Le droit d'être un réplica est ouvert à tous, mais le droit de valider (mining) est soumis à des prérequis connus de façon publique *(Algorithmes Heuristiques)* basés par exemple sur la localisation géographique ou sur la prise en compte de la rentabilité des noeuds existants pour ne pas mettre à mal l'intérêt de participer au réseau *(Saison 3)*.

Le schéma de la figure 1.14 ci-dessous représente les mécanismes d'amorçage de réplication et d'autodécouverte du réseau :

[1] Cryptoprocesseur permettant la génération, le séquestre sécurisé des clés cryptographiques et le calcul sans divulgation d'algorithmes cryptographiques (1.5.2)

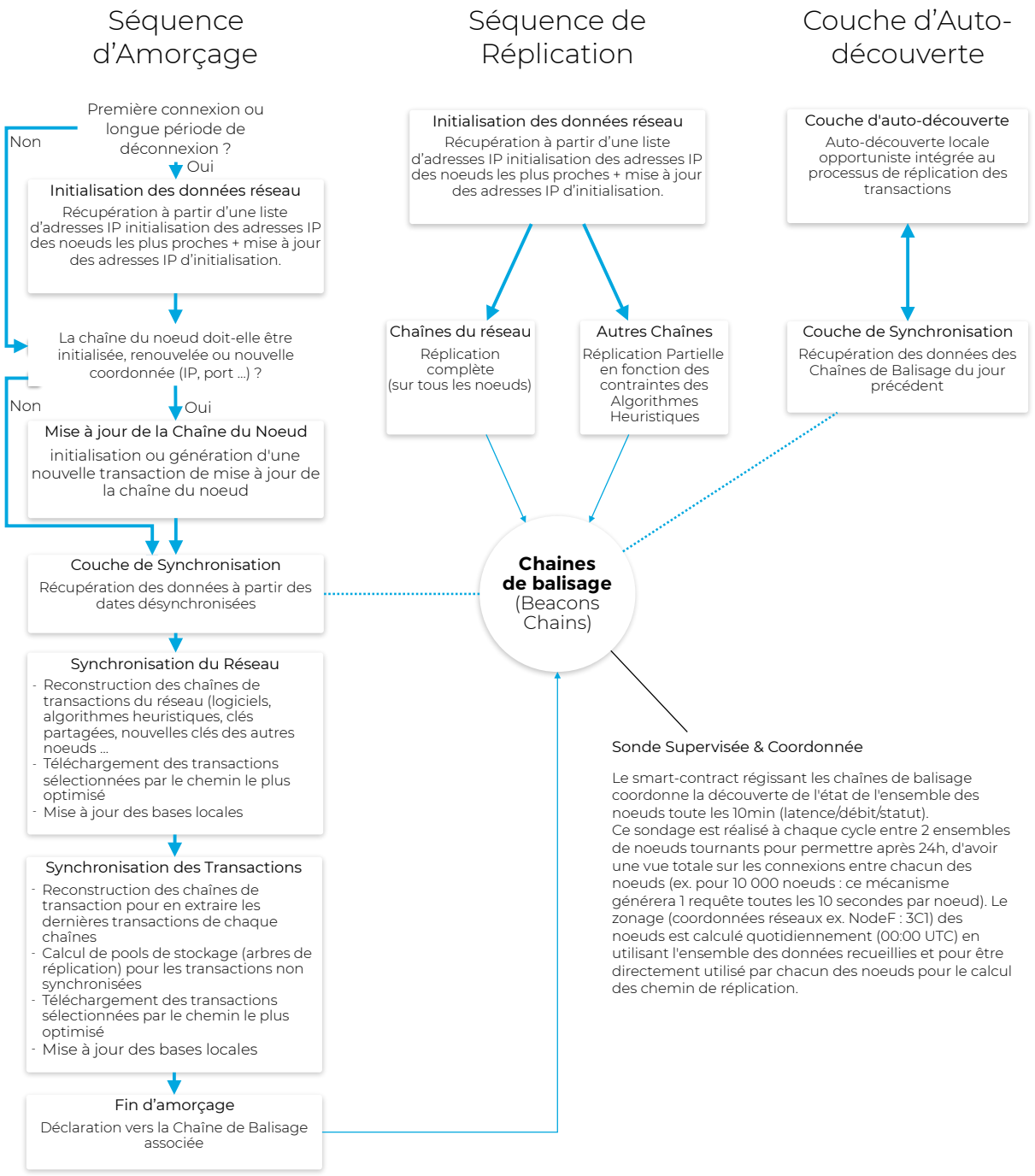


Figure 1.14: Amorçage, Réplication & Autodécouverte

Dans la suite de cette section seront définies :

1.4.1 Chaînes de Balisage garantes de la synchronisation globale

Calcul des adresses des Chaînes de Balisage (Beacons Chains) Aucun noeud n'ayant la capacité physique de connaître l'état de chaque transaction dans un réseau illimité, le réseau Uniris utilise un jeu de chaînes de transactions spécifiques contenant chacune un sous-ensemble des adresses des dernières transactions (00*, 01* ... FF*) pour une date donnée. L'adresse de la transaction est calculée directement à partir de la date et du sous-ensemble d'adresse (par l'intermédiaire d'une fonction de dérivation et d'un seed) de façon similaire au calcul suivant:

$$\text{PrivateKey}_{(subset\ +\ date)} = \text{Hash}(\text{subset + date, Storage Nonce}) \tag{1.5}$$

$$\text{Address}_{(subset\ +\ date)} = \text{Hash}(\text{PubKey}(\text{PrivateKey}_{(subset\ +\ date)})) \tag{1.6}$$

Example: Full day for 00 subset on 2019, April 18th

$$\text{Address}_{(00\ +\ 2019.04.18)} = \text{Hash}(\text{PubKey}(\text{PrivateKey}_{(00\ +\ 2019.04.18)})) \tag{1.7}$$

Example: Current day (every 10min) for 00 subset on 2019, April 18th at 2:40

$$\text{Address}_{(00\ +\ 2019.04.18\ \ 2:40)} = \text{Hash}(\text{PubKey}(\text{PrivateKey}_{(00\ +\ 2019.04.18\ 2:40)})) \tag{1.8}$$

Cette clé de dérivation est donc connue par l'ensemble des noeuds dès leur arrivée sur le réseau. Ce mécanisme permet ainsi à tout moment aux noeuds de retrouver la chaîne de transaction contenant l'ensemble des transactions d'une date donnée.

Suivi des Transactions et Horodatage

La transmission de l'adresse de la transaction validée au sous-ensemble de stockage associé (figures 1.10 et 1.11) est effectuée de façon conjointe et coordonnée entre les nœuds de validation. Pour garantir, la confidentialité sur le chaînage des transactions, l'adresse de la transaction précédente est chiffrée avec la clé partagée des noeuds *(Figure 1.19)*, permettant ainsi aux noeuds de mettre à jour leur table de réplication en reconstruisant localement l'historique des transactions *(Figure 1.13)*, mais sans pour autant divulguer les transactions associées à une même chaîne.

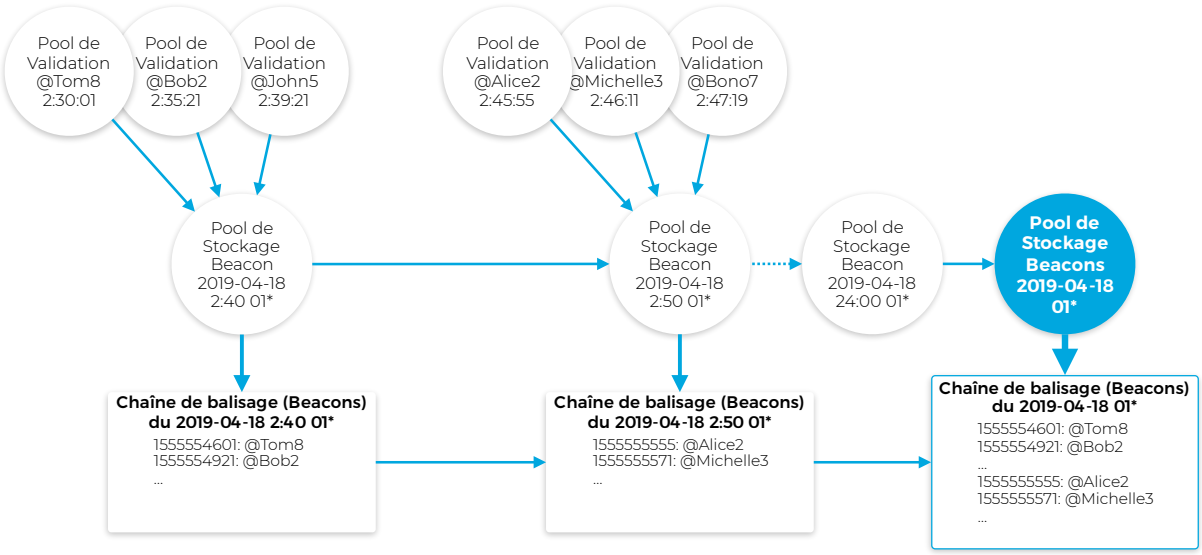


Figure 1.15: Chaînes de balisage (Beacons Chains) Suivi et Horodatage des transactions

La figure 1.15 représente le mécanisme de génération des chaînes des sous-ensembles toutes les 10 minutes ainsi que la génération de la dernière transaction qui sera la synthèse de toutes les informations recueillies de la journée pour un sous-ensemble donné. Même si l'ensemble des chaînes sont reliées de façon croisée par la signature

des précédentes transactions, chaque jour dispose de sa propre chaîne de transaction et donc de pools de stockage différents, permettant ainsi d'équilibrer le stockage sur chaque noeud. La liste des transactions est stockée compressée dans la zone DATA → Content *(Saison 2)*

État et Coordonnées Réseau des noeuds Les chaînes de balisage contiennent également l'état du réseau pour le sous-ensemble des noeuds dont la première clé publique (genesis) appartient au même sous-ensemble (01*). Pour chaque nouvelle transaction *"beacon"* sur cette chaîne (ie. "01*" le "2019-04-18" à "00:20") les noeuds de stockage associés auront pour tâche de :

- Vérifier l'état des noeuds appartenant à ce sous-ensemble et de stocker le résultat obtenu par consensus, sous forme binaire (la liste des noeuds étant connue: p.ex 111010110 pour le statut des 9 premiers noeuds de la liste ordonnée).
- Donner la liste des noeuds ayant participé à cette vérification (toujours sous-forme binaire à partir de la liste ordonnée).
- Et entre autres informations, la latence et le débit entre {chacun des noeuds du pool de stockage} et {chacun des noeuds du sous-ensemble (01*)}. Cette information est utilisée pour calculer les coordonnées réseau (1.3.3) de chacun des noeuds en fin de journée dans une transaction de synthèse qui sera téléchargée par l'ensemble des noeuds. Lors de ce calcul, comme représenté dans la figure 1.16, chacun des pools de stockage de "fin de journée" va récupérer toutes les chaînes de balisage de chacun des sous-ensembles pour calculer de façon globale le positionnement réseau de chacun des noeuds et intégrer cette liste (complète et calculée localement) dans la transaction de synthèse.

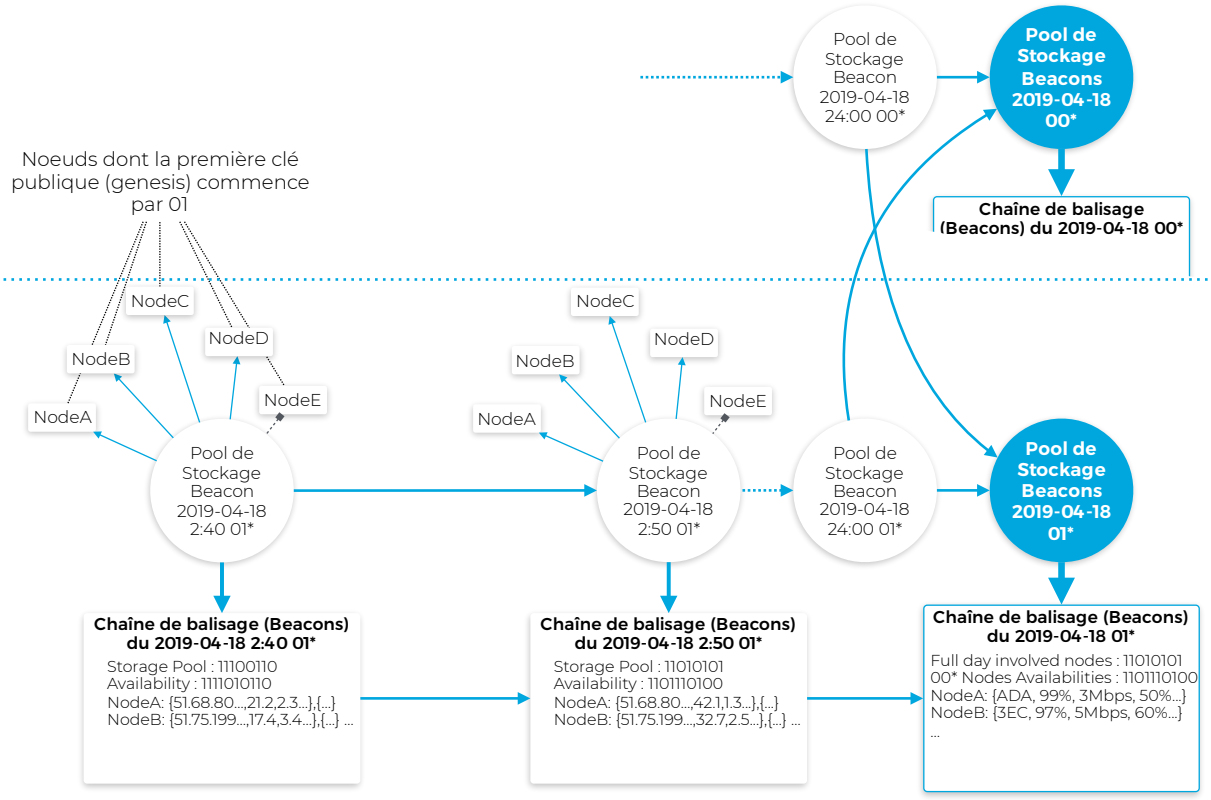


Figure 1.16: Chaînes de balisage : Statut des noeuds et coordonnées réseau

Paradigme du réseau illimité: Le découpage des sous-ensembles sera ajusté par les Algorithmes Heuristiques en fonction du nombre de transactions par seconde supporté par le système. À titre d'exemple, un sous-ensemble créé à partir du premier octet ("00*") générera une requête toutes les 2,56 secondes sur chacun des noeuds du pool pour une charge globale de 100 req/sec sur le réseau, de la même façon un sous-ensemble créé à partir des trois premiers octets ("000000*") générera une requête toutes les 5,59 secondes sur chacun des noeuds du pool pour une charge globale de 3 000 000 req/sec.

1.4.2 Couche d'Auto-Découverte du Réseau

Comme décrit dans les figures 1.14 et 1.17, l'opération d'auto-découverte "semi-passive" ou "opportuniste" va permettre à chacun des noeuds de se construire une vue locale sur l'état des noeuds voisins sans pour autant générer de nouvelles transactions. Les données réseau (IPs, protocoles, latence, débit ...) pourront être calculées directement et les autres données (utilisation des disques, utilisation moyenne du CPU/mémoire) seront transmises explicitement par les noeuds émetteurs de la transaction. Cette vue locale (cf. figure 1.17) sera ensuite confrontée aux vues des autres noeuds lors de la mise à jour des chaînes de balisage *(Figure 1.16)*.

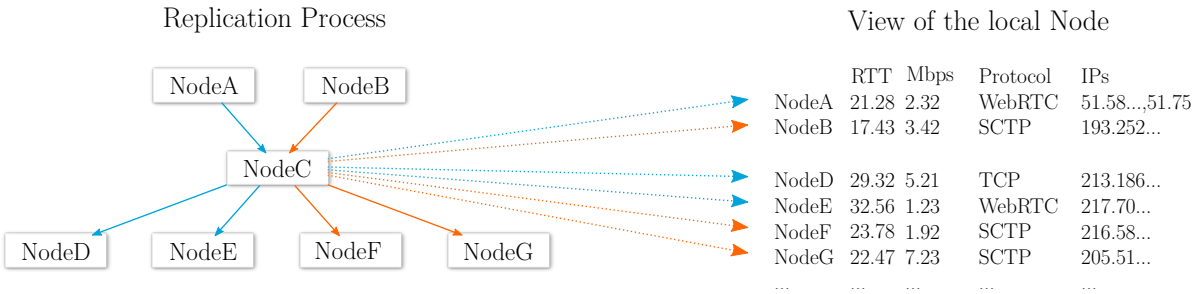


Figure 1.17: Découverte lors de la phase de réplication

1.4.3 Couche d'Amorçage du Réseau

Lors de la première connexion d'un noeud ou d'un client au réseau, le noeud ou le client devra contacter un des "Seed" pré-renseigné (liste d'adresses IP stables de noeuds ayant une disponibilité supérieure) qui lui retournera la liste mise à jour des "Seeds" et la liste des noeuds disponibles les plus proches (via *Coordonnées Géographiques vs. Coordonnées Réseau*).
Dans le cas de la première connexion d'un noeud, celui-ci générera une transaction d'initialisation de sa chaîne à destination d'un des noeuds les plus proches pour que celle-ci puisse ensuite être validée et complétée par les informations nécessaires pour participer au réseau (Secrets Partagés des noeuds, etc.).

Un noeud déjà initialisé, mais dont la chaîne serait périmée ou une des informations réseau modifiée (IP, port, protocol ...) devra également commencer la séquence par la mise à jour de sa chaîne.

Une fois la chaîne du noeud initialisée ou mise à jour, le noeud pourra demander la liste des noeuds du réseau à n'importe quel autre noeud. Cette liste contient pour chaque noeud du réseau : son identifiant, sa première (genesis) et sa dernière clé publique, sa (ou ses) {adresse IP, port, protocole, meilleur débit, meilleure latence, zone géographique, coordonnées réseau} et ses données système (utilisation du disque et disponibilité moyenne CPU/RAM). Cette vue est construite par chacun des noeuds à partir des données qualifiées des chaînes de balisage du jour précédent et est maintenue à jour grâce au processus de réplication des informations réseau (qui sont

toutes répliquées sur l'ensemble du réseau).

Une fois les informations minimales reconstituées, le noeud pourra alors resynchroniser l'ensemble de ces données de la façon suivante :

1. Calcul des adresses des chaînes de balisage *(Formule 1.6)* à partir des dates où les informations n'ont pas été synchronisées et récupération des transactions associées.
2. Reconstruction de l'historique des chaînes de transaction du réseau (chaînes des noeuds, des algorithmes Heuristiques, etc.), récupération des chaînes associées et mise à jour des bases locales.
3. Reconstruction de l'historique des autres chaînes de transaction, calcul des arbres de réplication, identification des transactions à télécharger (Figure 1.13) et récupération des transactions par les chemins le plus optimisés (via coordonnées réseau 1.3.3).
4. Enfin, lorsque l'ensemble des données auront été mises à jour, le noeud notifiera la chaîne de balisage associée à sa clé publique "genesis" pour qu'il puisse à nouveau participer au réseau.

Nota: *Pour de permettre aux noeuds de réaliser l'opération de resynchronisation, le pool de validation en charge de la mise à jour de la clé du noeud précisera la date à partir de laquelle le noeud pourra participer au réseau – pour calculer le temps nécessaire à un noeud pour resynchroniser ses données, le module d'apprentissage (Module de Prédiction) s'adapte en fonction de l'état de désynchronisation, des capacités matérielles et réseaux du noeud et des précédentes nofitications sur la chaîne de balisage pour affiner l'estimation.*

1.5 Une Sécurité au-delà des Solutions Connues

Le réseau Uniris a été conçu pour fournir une amélioration significative de la sécurité informatique pour les utilisateurs (identification biométrique sans aucun stockage de clé *(Saison 5)*, mais aussi pour améliorer significativement la sécurité sur les réseaux décentralisés sans pour autant toucher aux fondamentaux.

Dans la suite de cette section seront définis :

Malgré les avantages indéniables des réseaux distribués par rapport aux systèmes centralisés qui reposent uniquement sur la confiance de tiers, il convient de lister néanmoins les vulnérabilités connues de ces réseaux :

Faille des 51% de la Proof-of-Work pour les preuves de travail basées sur la puissance de calcul (HashRate) la faille pour un attaquant consiste a rassembler pendant une période donnée une puissance de calcul équivalente à la puissance du réseau complet. La sécurité du réseau dépend directement de la puissance de calcul de ses noeuds.

Failles de la Proof-of-Stake La Proof-of-Stake au-delà du caractère recentralisé du consensus, les principaux risques sont la prise de contrôle à partir de moyens financiers et pour un attaquant de pouvoir concentrer son attaque sur quelques noeuds du réseau pour en prendre le contrôle.

Failles des de la dBFT la Delegated Byzantine Fault Tolerance n'est pas non plus un réseau entièrement décentralisé (mixant généralement des mécanismes de proof-of-Stake ou des mécanismes de preuve de participation déléguée) communément ce type de consensus nécessite de regrouper (n-1)/3 ($\approx$ 66%) des noeuds pour réussir une attaque.

Failles du "Split-Brain" le split-brain peut intervenir de façon non frauduleuse, par exemple après la coupure d'un câble sous-marin rendant une moitié du réseau inaccessible à l'autre. Dans ce cas de figure, deux versions de chaînes pourraient exister pendant la durée de la coupure.

Failles DDoS la faille DDoS, consiste à surcharger de transactions tout ou partie des noeuds rendant alors une partie du réseau incapable de participer aux consensus.

Failles des attaques Sybil l'attaque Sybil (dédoublement de personnalité) consiste à surcharger le réseau de fausses informations à partir d'identités multiples ou à partir d'une identité démultipliée, cette attaque est d'autant plus efficace sur des réseaux non structurés utilisant les informations p2p des noeuds voisins pour créer leurs vues.

Failles des smart-contracts (vm) la faille la plus connue est la faille "TheDAO" apparue en mai 2016 et qui a eu pour conséquence la création de deux chaînes de blocs distinctes sur le réseau Ethereum. Cette faille a mis en exergue le problème du code exécuté en aveugle par des machines virtuelles sur des noeuds. À ce titre, d'autres Blockchain, proposent par exemple une méthode de programmation impérative des smart-contracts permettant d'améliorer le déterminisme sur le résultat en sortie, mais le consensus est généralement basé sur de la Proof-of-Stake ou de la dBFT.

Failles des Wallets les wallets (gestionnaire de portefeuilles) peuvent se résumer à trois principales familles:

Wallets Externalisés généralement fournies par des plateformes d'échanges, elles posent le problème de la centralisation des clés et sont donc des cibles de choix pour un attaquant (Zaif, Coincheck), les attaques à partir de failles "zéro day" étant bien plus aisées que de prendre le contrôle de dizaines de milliers de mineurs.

Wallet Applicatives apportent un vrai confort pour l'utilisateur qui peut utiliser en tous lieux ses clés cryptographiques, mais qui posent les problèmes de la sécurité de la couche système, de la sécurité des autres applications installées, mais aussi de la façon dont elles ont été développées (ex. Bitcoin-Qt qui posait le problème de la prédictibilité des clés générées)

Hardware Wallets les portefeuilles physiques apportent le meilleur moyen de sécuriser les clés cryptographiques, mais posent le problème, par ailleurs généralisé, de la perte du support.

Les vulnérabilités listées couvrent quatre domaines, sur les lesquels le réseau Uniris fournit les contre-mesures suivantes :

Couche Consensus là où les meilleurs consensus offrent une résistance aux attaques de l'ordre de 66% de noeuds malhonnêtes, le réseau Uniris grâce au Consenus ARCHE permet d'atteindre un risque de 10^{-9} (1/1 000 000 000) inférieur aux standards aéronautiques ou nucléaires avec un réseau constitué de plus de 90% de noeuds malhonnêtes *(Figure 1.18)*. Le consensus ARCHE étant basé sur la Validation Atomique ou le consensus formel obtenu par élection Heuristique tournante, en d'autres termes : un accord total sur la validation d'une transaction des noeuds élus à partir d'une élection aléatoire et tournante.

Couches Réseau là où la majorité des solutions Blockchain reposent sur Gossip, le réseau Uniris se base sur la Multidiffusion Supervisée *(Supervized Multicast – Figure 1.14)* permettant d'éviter les flux réseau inutiles tout en apportant une vision complète et partagée de l'état des noeuds. Cette capacité permet ainsi de détecter tout problème de type Split-Brain et d'apporter les solutions de contremesure adaptées. L'ensemble des flux étant authentifiés au niveau des clients (clés partagées des dispositifs et logiciels émetteurs) ou des noeuds (chaînes de noeuds et Secrets Partagés) toute attaque de type DDoS est rapidement identifiée et écartée.

Couche Smart-Contracts le réseau Bitcoin s'appuie sur les entrées non dépensées (unspent outputs UTXO) permettant à chacun de pouvoir vérifier la cohérence entre les différentes entrées sur le registre et les sorties. Chaque transaction étant signée cryptographiquement il est impossible de créer une fausse transaction sur un Wallet dont la clé privée est inconnue. Le réseau Uniris s'appuie sur cette même propriété : seules des transactions validées font autorité, il n'est, par exemple, pas possible de faire fonctionner une base de données à l'intérieur d'un smart-contract. Par contre, la totalité des smart-contracts actuels fonctionneront au moins aussi bien sur le réseau Uniris que sur d'autres réseaux *(par exemple dans un smart-contract d'e-commerce, le smart-contract émis par un marchant pourra définir des stocks, des prix et des interactions avec ses clients en utilisant une vue calculée en permanence par les noeuds en charge du stockage du smart-contract et en fonction des transactions validées à destination de ce même smart-contract – (Saison 2)*. Le fonctionnement "UTXO" ne donne pas d'état à l'intérieur d'un smart-contract mais permet de le calculer (dans l'exemple ci-dessus le marchant ne peut pas directement interroger un smart-contract sur l'état des commandes, mais peut vérifier l'état prouvé des commandes par l'intermédiaire des transactions validées).

Couche Wallet bien qu'abordé en dernier point, le but premier du réseau Uniris a été

de développer une méthode d'identification biométrique permettant à un utilisateur de récupérer ses clés cryptographiques directement à partir de son corps et sans jamais avoir à stocker ces données biométriques ou dérivées ou que se soit, en dehors du corps lui-même *(Saison 5)*. Associé au réseau Uniris, le mécanisme utilisé permet même de dévoiler ses clés privées sans qu'un attaquant puisse générer de transaction. L'objectif étant de supprimer les freins à l'adoption associés à la gestion des clés cryptographiques et à leur sécurisation pour n'importe quel humain.

1.5.1 Résilience du réseau & détection des opérations malveillantes

Les propriétés énoncées de l'ARCHE nécessitant 100% de réponses positives et concordantes, le réseau permettant d'authentifier chacun des noeuds et la multidiffusion supervisée permettant de fiabiliser la connaissance de la disponibilité des noeuds permet au consensus de répondre à la loi de distribution hypergéométrique (1.9). La distribution hypergéométrique permet de décrire la probabilité de k succès (détection d'opération frauduleuse) pour n tirages (vérifications) sans remise avec un nombre total finit de noeuds N et en considérant un nombre N1 de nœuds malveillants (90%) :

$$\mathbb{P}[X = k] = \sum_{k=1}^{p} \frac{\binom{N_1}{k} \times \binom{N-N_1}{n-k}}{\binom{N}{n}} \tag{1.9}$$

Cette loi permet ainsi de calculer la chance de détecter une opération malveillante en connaissant le nombre de validations (vérification). L'exemple ci-dessous permet de calculer que pour un total 100 noeuds, dont 90 malveillants, 42 vérifications permet d'obtenir 99,7% de chance de détecter une opération frauduleuse (d'avoir au moins un noeud honnête : $1 \leqslant k \leqslant 10$)

$$\mathbb{P}[1 \leqslant X \leqslant 10] = \sum_{k=1}^{10} \frac{\binom{10}{k} \times \binom{90}{42-k}}{\binom{100}{42}} \approx 99.7\% \tag{1.10}$$

De la même façon et sur ce même tirage, 84 vérifications vont permettre d'obtenir 99,999999999% de chance de détecter une opération frauduleuse soit un risque de ne pas la détecter de 1.2×10^{-9} *(soit au-delà des standards du risque acceptable pour l'aviation ou le nucléaire)* :

$$\mathbb{P}[1 \leqslant X \leqslant 10] = \sum_{k=1}^{10} \frac{\binom{10}{k} \times \binom{90}{84-k}}{\binom{100}{84}} \approx 99,9999999\% \tag{1.11}$$

La distribution hypergéométrique devient réellement intéressante quand on commence à augmenter le nombre total de noeuds du réseau (toujours pour 90% de noeuds malhonnêtes). La figure 1.18 ci-dessous montre le nombre de vérifications nécessaires (nombre de cycles) en fonction du nombre de noeuds pour obtenir un risque de l'ordre 10^{-9}. Ainsi pour 100 noeuds il faudra 84 vérifications (84%), pour 1000 noeuds : 178 (soit 7,8%), pour 10000 noeuds : 195 (soit 1,9%) et pour 100000 noeuds : 197 (soit 0,2%)

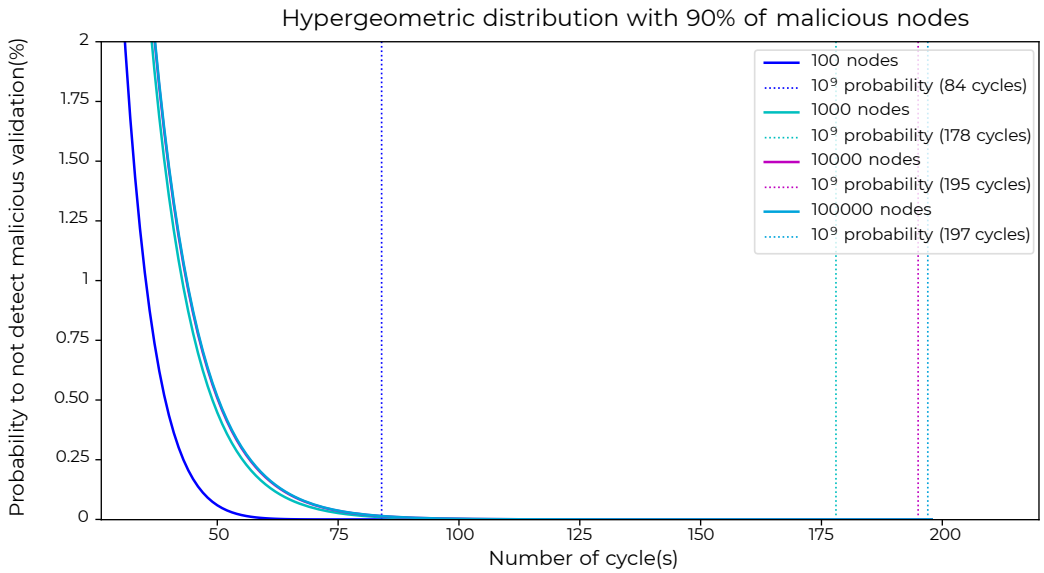


Figure 1.18: Répartition Hypergéométrique pour 100, 1000, 10000 et 100000 noeuds

$$1 - \left[\lim_{N \to +\infty} \mathbb{P}[X = k]\right] = 1 - \left[\lim_{N \to +\infty} \sum_{k=1}^{p} \frac{\binom{0.1*N}{k} \times \binom{0.9*N}{n-k}}{\binom{N}{n}}\right] \approx 10^{-9} \Rightarrow n \approx 200 \qquad (1.12)$$

Cela signifie que même avec 90% de noeuds malveillants, quel que soit le nombre de noeuds à l'intérieur du réseau, un contrôle avec une partie infime du réseau (inférieur à 200 noeuds) permet d'assurer la propriété d'atomicité des transactions du réseau.

1.5.2 Une cryptographie évolutive et résistante aux ordinateurs quantiques

Pour permettre une rétrocompatibilité cryptographique et faire évoluer le réseau au fur et à mesure des progrès de la recherche en cryptographie, mais aussi pour donner la possibilité aux personnes, organisations ou pays de choisir un algorithme cryptographique spécifique, la clé publique sera versionnée sur un octet indiquant l'algorithme utilisé. Les algorithmes supportés étant listés dans une chaîne de smart-contracts spécifiques.
Combiné au mécanisme de non-divulgation des clés publiques *(Remarque 2)* et combiné à la nécessité de connaître simultanément plusieurs clés privées temporaires (1.2) pour être autorisé à générer une transaction, la tâche d'un ordinateur quantique potentiellement capable de "casser" des clés privées devrait être considérablement complexifiée.
Sauf problème de compatibilité matérielle (HSM, etc.), les signatures EdDSA, Curve25519 et AES256 seront utilisées par défaut sur le réseau.

1.5.3 Séparation des Pouvoirs pour Renforcer la Sécurité

Le réseau Uniris est basé sur le concept de SoD *(Segregation of Duties - séparation de rôles)*. Pour permettre une séparation formelle des pouvoirs entre les mineurs eux-mêmes et entre les utilisateurs et les mineurs, le réseau Uniris utilise la contrainte technique de connaissance d'un groupe de clés cryptographiques (un groupe connait la clé privée alors qu'un autre n'en connait que la clé publique et inversement).
La connaissance des secrets partagés permet aux nœuds de participer au processus de validation d'une transaction (minage) et donc d'être rémunéré pour ce travail. Techniquement, le secret partagé est une clé dérivée ou étirée, générée à partir d'une fonction de dérivation de clé (KDF: key derivation function) [3] [4] permettant aux noeuds de retrouver la clé partagée à partir d'un secret (Seed), d'une fonction de dérivation et, dans le cas des secrets partagés des noeuds, de la date. Le secret (Seed) et la fonction de dérivation sont chiffrés avec la dernière clé publique connue de chacun des noeuds et renouvelés à chaque fois qu'un noeud est banni du réseau (cf. figure ci-dessous).
Le renouvellement de la clé partagée des noeuds est effectué quotidiennement à 00:00 UTC et est communiquée au reste du réseau dans les minutes qui précèdent.

Utilisant le mécanisme des smart-contracts auto déclenchables *(Saison 2)*, la génération quotidienne d'une nouvelle transaction sur la chaîne des Secrets Partagés (Figure 1.19: @NodeShardKey-3) est est confiée au premier noeud de stockage disponible associé à cette adresse (1.3.2) et est validée de façon imprédictible par les noeuds de validation élus (1.2.2). Pour garantir l'intégrité de cette chaîne même en cas de compromission de la clé privée, celle-ci utilise le mécanisme des contraintes récursives héritées *(Saison 2)* interdisant toute modification de la chaîne indépendamment de la connaissance de la clé privée.

La figure 1.19 ci-dessous représente le renouvellement des *Secrets Partagés* après bannissement d'un ou plusieurs noeuds du réseau:

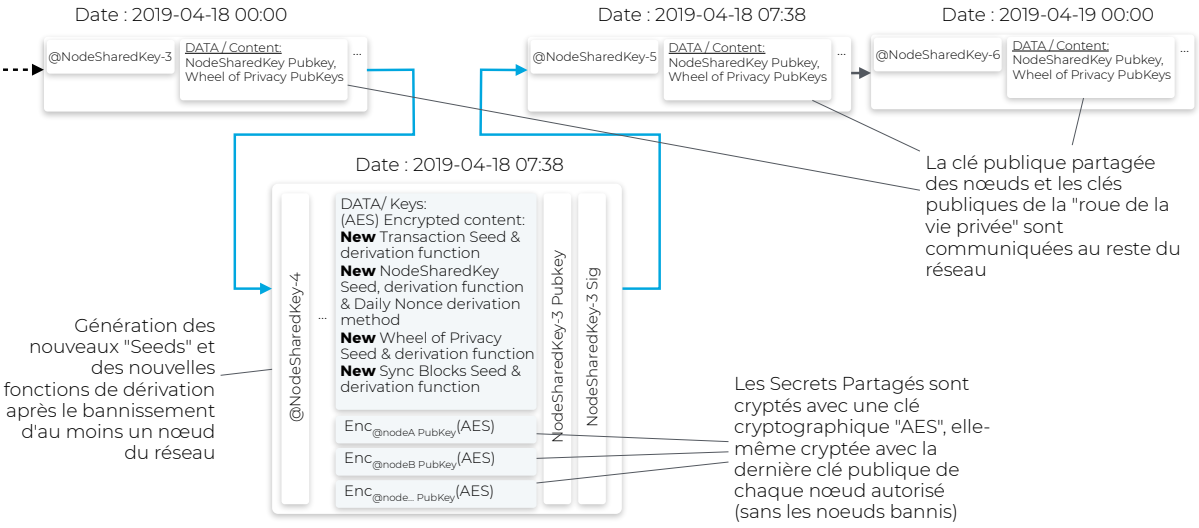


Figure 1.19: Secrets Partagés des Noeuds

1.6 Un Réseau capable de se Reconfigurer pour Anticiper les Catastrophes

Pour permettre à un réseau décentralisé de survivre à des décennies voir des siècles, il doit pouvoir s'adapter aux menaces et réagir en conséquence. Pour ça le réseau Uniris repose de deux algorithmes adaptatifs :

Capacité d'action et de réaction : cette capacité est assurée par les Algorithmes Heuristiques qui gèrent la majorité du comportement du réseau : par exemple en adaptant les contraintes sur le nombre de validations/ réplications en fonction du nombre de noeuds connectés du réseau ou encore en pondérant une zone géographique en fonction de la disponibilité moyenne des noeuds. L'ensemble des comportements est préétabli directement dans les algorithmes.

Capacité de prédiction, d'anticipation et de correction : ce module a la capacité de faire le lien entre des motifs/échantillons (Pattern) et un futur comportement et donc potentiellement de prédire un futur état du réseau. Par exemple de détecter qu'un opérateur dans un pays donné met à jour mensuellement le micrologiciel de ses box et que le réseau sera coupé pour tous les noeuds associés pendant cette période, que le réseau électrique de telle ou telle région du monde est coupé pendant les orages ou encore qu'avant une tentative d'attaque les noeuds en charge de la validation auront tendance à répondre moins rapidement aux autres transactions. L'objectif étant de garantir en tout temps la disponibilité des données, par exemple en changeant le poids de disponibilité d'un des noeuds élu pour la réplication ou en augmentant le nombre de validations nécessaire pour une transaction donnée sans pour autant baisser les contraintes des Algorithmes Heuristiques.

1.6.1 Uniris Oracles Chains

Pour reconstituer le contexte des événements avant l'apparition de l'anomalie, le réseau doit disposer d'un maximum d'informations qualifiées. Pour cela le module de prédiction utilise deux mécanismes décentralisés :

Chaînes de Balisage (Beacons Chains): le premier, liste l'ensemble des états du réseau et des transactions toutes les 10min et le synthétise tous les jours, son fonctionnement est décrit dans la section *Chaînes de Balisage garantes de la synchronisation globale*.

Oracle: l'Oracle *"État du Monde"* fonctionne de la même façon que les Chaînes de balisage *Beacons Chains* à la différence près que les nouvelles transactions sur la chaîne ne sont pas générées toutes les 10min, mais à chaque mise à jour d'une information (par exemple à la diffusion d'un nouveau bulletin météorologique) - l'ensemble des informations seront également synthétisées en fin de journée (00:00 UTC) par une dernière transaction sur la chaîne. Les informations listées à l'intérieur de cette chaîne sont de plusieurs natures : climatiques, financières (cours de la bourse, des cryptomonnaies, dont l'Uniris Coin), sociétale (nombre d'occurrence de mots clés sur les sites d'information ou même lorsque c'est possible les derniers mots les plus utilisés sur les moteurs de recherche). L'ensemble des références (URL, etc.) est listé dans une chaîne de smart-contracts spécifique.

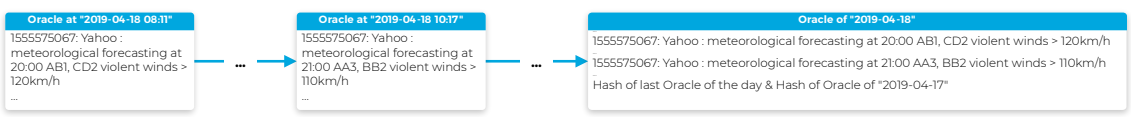


Figure 1.20: Représentation d'une des Chaînes de l'Oracle

L'objectif de ces données étant de reconstituer le contexte avant la période de l'événement pour en détecter les signes précurseurs.

1.6.2 Paramètres du module de Prédiction

Les paramètres du module de prédiction sont hébergés directement sur la chaîne des smart-contracts *Uniris Prediction Module*. Ce smart-contract autonome pourra modifier en permanence sa chaîne, sans néanmoins avoir la possibilité de modifier les contraintes imposées *(Anomalies Triggers, Scope of countermeasures proposals & KPI)* qui seront protégées par des contraintes récursives héritées *(Saison 2)* imposant un vote de la communauté *(Saison 3)*. Le module de prédiction dispose de 5 paramètres principaux :

Déclencheur d'Anomalies: (*Anomalies Triggers*) les anomalies, contrairement aux *Chaînes de Balisage* et aux *Chaînes de l'Oracle* ne donnent pas de contexte d'apprentissage, uniquement les événements et les seuils à surveiller déclencheurs de la phase de détection des patterns (schéma qui semble avoir conduit à l'anomalie). Cette détection d'anomalie est réalisée localement par chacun des noeuds, notamment lors du cycle d'autoréparation du réseau (1.3.4) par exemple lorsqu'une transaction aura un niveau de réplication dangereusement faible après l'indisponibilité d'un grand nombre de noeuds sur une ou plusieurs zones géographiques.

Périmètre des Contre-mesures: *(Scope of countermeasures proposals)* le périmètre des contre-mesures permet aux noeuds de savoir sur quels paramètres les contre-mesures vont pouvoir s'appliquer (p.ex demander une zone de géographique de réplication supplémentaire pour les transactions dont les noeuds de stockage se trouvent sur le passage d'une tempête).

KPI: l'indicateur clé de performance permet de pondérer chacune des propositions de contre-mesure notamment pour assurer un coût minimum au réseau par exemple: le temps de calcul supplémentaire, le nombre de réplicas, le nombre de validations supplémentaires induites, l'anticipation en temps permis et bien sûr le pourcentage de succès de la contre-mesure proposée.

Demandes de contre-mesures: (*Request for countermeasures*) sont les demandes du réseau associées à des anomalies qualifiées (ayant reçu un nombre minimum de noeuds confirmant l'anomalie), les anomalies sans propositions de contre-mesures pertinentes (facteurs d'efficacité KPI) resteront listées dans cette section.

Déclencheurs de Contre-mesures : (*Network Countermeasures triggers*) ces déclencheurs sont utilisés en permanence par les Algorithmes Heuristiques en particulier pour les élections des noeuds de validation et de stockage, mais surtout pendant la phase d'autorépartion (1.3.4) permettant ainsi au réseau d'appliquer des contre-mesures avant même qu'un événement se produise.

1.6.3 Fonctionnement du modèle de prédiction

Le module de prédiction est hébergé sur la chaîne des smart-contracts *Uniris Prediction Module* répliqué sur l'ensemble des noeuds du réseau. Techniquement, le contenu des déclencheurs (Déclencheurs d'anomalies et de contre-mesures) est ajouté à la base locale du registre des déclencheurs *Triggers Ledger* (1.3.5) et sera donc consulté pour chaque opération. Le schéma ci-dessous représente de façon simplifiée le mécanisme permettant de modifier le modèle de prédiction suite à l'apparition d'une anomalie qualifiée.

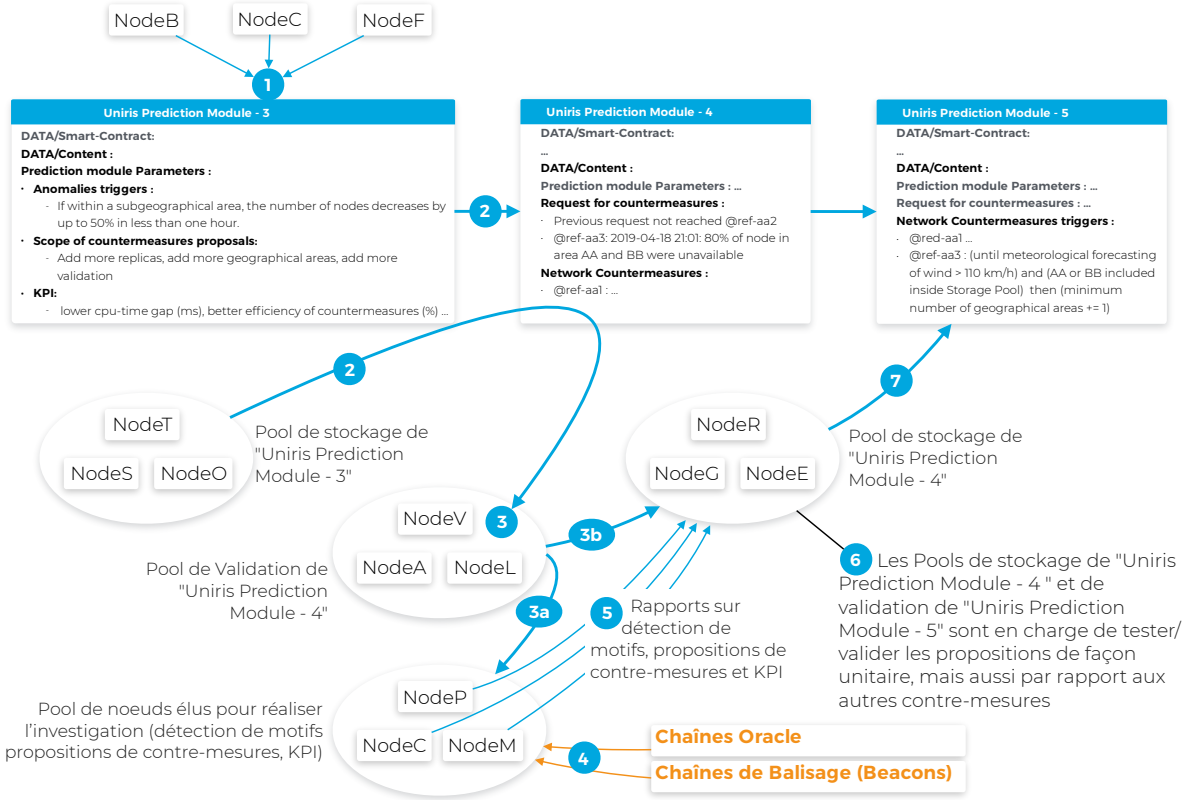


Figure 1.21: Module de Prédiction

1 l'anomalie est détectée par plusieurs noeuds qui vont transmettre une nouvelle transaction à destination de la chaîne de smart-contracts *Uniris Prediction Module*

2 une fois le quorum du nombre de confirmations de noeuds atteint, le pool de stockage en charge du stockage du smart-contract *Uniris Prediction Module - 3* génèrera une mise à jour du smart-contract par l'intermédiaire d'une nouvelle transaction sur la chaîne en ajoutant une nouvelle demande de contre-mesures *Request for countermeasures*.

3 Au moment de la validation de la transaction:

3a le pool de validation va élire les noeuds qui seront en charge de l'investigation (noeuds P, C et M) et intégrer les noeuds élus au processus de réplication.

3b avant de lancer le processus de réplication de cette nouvelle transaction.

4 les noeuds élus vont alors récupérer les données des chaînes de l'Oracle et de Balisage qui ont précédé l'événement pour commencer la recherche des événements (patterns) qui auraient pu prédire l'anomalie. Une fois les différents événements évalués, chacun des noeuds va alors tester de la modification de chacun des paramètres autorisés *Scope of countermeasures proposals* pour en extraire les plus efficaces avant d'en mesurer l'impact à partir des transactions hébergées localement (bac à sable).

5 l'évaluation terminée chacun des noeuds élus pour l'investigation transmettra au pool de stockage du smart-contract *Uniris Prediction Module - 4* les propositions les plus pertinentes (Patterns / Contre-mesures et KPI).

6 une fois le quorum des réponses reçues, le pool de stockage du smart-contract *Uniris Prediction Module - 4* testera en local les différentes propositions et si un consensus est trouvé, généra une mise à jour du smart-contract par l'intermédiaire d'une nouvelle transaction contenant la contre-mesure validée. Cette contre-mesure sera à nouveau testée par le pool de validation de cette nouvelle transaction avant d'être répliquée sur l'ensemble du réseau.

7 une fois la mise à jour de la chaîne propagée, chacun des noeuds du réseau intégrera ce nouveau déclencheur dans la base locale des triggers *Triggers Ledger* (1.3.5) pour être ensuite vérifié lors de chaque processus de validation, de réplication ou d'auto réparation.

1.7 0.1g de sucre : une consommation énergétique divisée par 3,6 milliards

Malgré la résistance des réseaux décentralisés basés sur la preuve de travail permettant de passer de la confiance imposée par un tiers à une confiance prouvée et distribuée. La consommation énergétique de la Preuve-de-travail basée sur le Hashrate (jusqu'à 400 kWh pour la validation d'une seule transaction) est clairement un frein à son adoption massive. Cette consommation ne permet pas non plus, même à moyen terme, d'imaginer une utilisation à grande échelle de cette révolution technologique tant pour sa rentabilité (le coût planché de l'électricité consommée pour un mineur étant de 2$ par transaction) que pour le nombre de centrales électriques qu'il faudrait déployer.

D'autres solutions sont apparues comme la Proof-of-Stake, mais qui posent un problème évident de centralisation et de gouvernance ou encore les DAG (Directed Acyclic Graph), mais qui posent le problème de la disponibilité des données. C'est notamment sur l'extrême rigueur associée à la disponibilité des données et sur l'esprit inclusif, global et décentralisé de la preuve de travail (mais non basée sur le HashRate) que le réseau Uniris se démarque.

Le calcul ci-dessous, bien que théorique, est basé sur les premiers résultats obtenus, les hypothèses prises en compte sont les suivantes :

- Consommation d'un noeud Uniris utilisé à 100% : 15Wh (ex. Intel NUC i3)
- En hypothèse conservative, dix noeuds sont dédiés pendant 10 secondes pour valider et répliquer une transaction (les chiffres actuels étant: inférieur à 1 seconde pour le traitement unitaire d'un noeud pour la validation d'une transaction et de l'ordre de 100ms pour la réplication, le tout multithreadé donc non dédié).
- Le nombre de transactions du réseau Bitcoin est de 93 millions transactions par an avec une consommation énergétique de l'ordre de 38,7 TWh/an.

Ce qui nous donne pour 10 noeuds sur un an : $365.4 \times 24 \times 60 \times 6 = 3\,157\,046$ transactions et donc $93\,000\,000 \div 3\,157\,046 = 29{,}45$ soit $29.45 \times 10 = 295$ noeuds pour couvrir la puissance de minage actuelle du réseau Bitcoin.

Quant à la consommation, on obtient alors $295 \times (365{,}4 \times 24) \times 15 = 38\,805$ kWh/an ($10^{-9} \times 38{,}7$ TWh/an soit 3,6 milliards de fois moins) 0,42W/sec par transaction soit l'équivalent énergétique en joule d'un dixième de gramme de sucre.

Bibliographie

[1] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. *Vivaldi: A Decentralized Network Coordinate System*. SIGCOMM'04, 2004.

[2] Bernadette Charron-Bost and André Schiper. The heard-of model: Computing in distributed systems with benign failures. 01 2007.

[3] Pieter Wuille. Hierarchical deterministic wallets. 2012.

[4] Marek Palatinus, Pavol Rusnak, Aaron Voisine, and Sean Bowe. Mnemonic code for generating deterministic keys. 2013.