Æternity blockchain – La machine-Oracle sans confiance, décentralisée et purement fonctionnelle

6 Fevrier 2017

v0.1

Zackary Hess
zack@aeternity.com

Yanislav Malahov
yani@aeternity.com

Jack Pettersson
jack@aeternity.com

Résumé —
Depuis l'introduction d'Ethereum en 2014 beaucoup d'intérêt s'est manifesté dans les applications sans confiance décentralisées (contrats intelligents). En conséquence plusieurs applications ont essayé de mettre en place des bases de données réelles au-dessus d'une Blockchain. Nous pensons que stocker l'état de l'application son code sur la chaîne n'est pas efficace pour plusieurs raisons. Avec æternity nous présentons une architecture hautement évolutive d'une Blockchain avec un mécanisme de consensus, également utilisé pour consulter l'oracle. Cela rend celle-ci très efficace, en évitant la superposition de plusieurs mécanismes de consensus sur les uns sur les autres. Des Canaux de l'État sont intégrés afin d'accroître la protection des renseignements personnels et possibilité d'évolution de la Blockchain. La monnaie dans les canaux peut être transférée à l'aide de contrats-intelligents purement fonctionnels qui peuvent accéder aux réponses de l'oracle. En évitant le stockage du code du contrat ou de l'État sur la chaîne, nous sommes en mesure de faire des contrats-intelligents plus facile à analyser et à rendre le processus plus rapide, sans perte de fonctionnalité. Les applications comme les marchés pour les actifs synthétiques et les marchés de prévision peuvent être efficacement mis en œuvre à échelle mondiale.
De nombreuses pièces d'implémentations d'æternity et des preuves de concept sont déjà écrites en Erlang. Le Développement des outils et éléments essentiels tels qu'un portefeuille, un système de nomination et d'identité sont en cours de développement et seront également fournis.

I. Introduction

L'intention de cet article est de donner un aperçu de l'architecture de æternity Blockchain et des applications possibles sur celle-ci. Des documents plus détaillés seront publiés à l'avenir, en particulier pour les mécanismes de consensus et de gouvernance. Cependant, il convient de noter que notre architecture est holistique ; Tous les composants se lient et se synergisent, de manière modulaire. Le reste de cet article est divisé en quatre parties. Tout d'abord, nous allons présenter et discuter des idées théoriques fondamentales qui informent notre architecture. Deuxièmement, nous discuterons des applications ssentielles incluses, d'autres cas d'utilisation

possibles et nous donnerons des cas d'utilisation possible de la plate-forme en tant que développeur. Troisièmement, nous présenterons la mise en œuvre actuelle de la preuve de concept, écrite dans Erlang. Nous concluons avec une discussion, y compris des orientations futures possibles et des comparaisons avec d'autres technologies.

A. Travaux précédent

Les Blockchains, et tout d'abord Bitcoin, a montré une nouvelle façon d'organiser et d'archiver l'échange de valeur sur Internet. Cela a été suivi d'un certain nombre de progrès prometteurs: Ethereum a démontré un moyen d'écrire des contrats intelligents Turing-complet garantis par une architecture Blockchain; Truthcoin a créé des outils pour fabriquer des oracles sur les Blockchains, tandis que Group Gnosis et Augur ont montré comment les rendre plus efficaces; Casey Detrio a montré comment faire des marchés sur les Blockchain; Namecoin a montré comment faire l'équivalent distribué d'un serveur de noms de domaine (DNS); Factom a montré comment une Blockchain qui stocke des hachs peut être utilisée comme preuve d'existence pour toute donnée numérique.

Ces technologies sont très prometteuses quand il s'agit de fournir des services financiers et juridiques de première classe à tout le monde. Jusqu'à présent, ils n'ont pas réussi à se réunir en un tout unifié qui réponde réellement au besoin. Plus précisément, toutes les solutions jusqu'ici ont manqué dans au moins un des aspects suivants: la gouvernance, l'évolutivité, la sécurité des scripts et l'accès bon marché aux données du monde réel. æternity vise à améliorer l'état technique des aspects cite ci-dessus sous tous leurs aspects.

II. La BLOCKCHAIN ÆTERNITY

Nous croyons que le manque d'évolutivité, la sécurité des scripts et l'accès bon marché aux données réelles des «plates-formes de contrats intelligentes» actuelles se résument à trois problèmes essentiels. Tout d'abord, la conception actuelle de l'état rend les contrats intelligents écrits pour la plate-forme difficile à analyser, et la conformité avec l'ordre séquentiel des transactions compliqué et l'évolutivité difficile. Deuxièmement, le coût élevé de l'introduction de données du monde réel dans le système de manière décentralisée, sans confiance et fiable complique ou empêche la réalisation de nombreuses applications prometteuses. Troisièmement, les plates-formes sont limitées dans leurs capacités à se moderniser, afin de s'adapter aux nouvelles connaissances technologiques ou économiques. Nous croyons que chacun de ces trois problèmes présente des solutions claires qui devraient être explorées.

Tout d'abord, des recherches récentes sur la technologie des canaux d'État suggèrent que, pour de nombreux cas d'utilisation, il n'est pas nécessaire de maintenir l'état (des contrats) sur la chaîne. Il est très souvent tout à fait possible de stocker toutes les informations dans les canaux d'État et n'utiliser la Blockchain que pour régler les résultats économiques de l'échange d'informations et les cas de conflit. Cela suggère une approche alternative à l'architecture de chaîne de blocs dans laquelle les contrats intelligents Turing-complet existent dans les canaux d'états mais pas sur la chaîne. Cela augmente l'évolutivité puisque toutes les transactions deviennent indépendantes et peuvent donc être traitées en parallèle.

En outre, cela signifie que les contrats n'écrivent jamais à l'état partagé, en simplifiant considérablement leurs tests et leurs vérifications. Nous croyons que cette conception souligne que l'usage des Blockchains serait plus efficace dans la logique financière plutôt que le stockage de données ; Il existe des solutions de stockage décentralisées qui complètent parfaitement les Blockchains plus efficacement.

Deuxièmement, des applications telles que Augur ont tenté de mettre les données du monde réel sur la Blockchain de manière décentralisée - dans le processus essentiellement la construction d'un mécanisme de consensus dans les contrats intelligents, au lieu d'utiliser le mécanisme de consensus de la Blockchain sous-jacente. Cela entraîne des inefficacités mais n'augmente pas la sécurité. La conclusion naturelle en est de généraliser le mécanisme de consensus de la Blockchain afin qu'il puisse fournir des informations non seulement sur le prochain état interne, mais aussi sur l'état du monde extérieur. On pourrait ainsi argumenter que le mécanisme de consensus de la Blockchain détermine le résultat de l'exécution de la théorie de la complexité qui s'appuie sur une machine-Oracle: une machine théorique plus puissante qu'une machine Turing car elle répond à certaines questions qui ne peuvent pas nécessairement être calculées Comme «Qui a gagné le match de football X?».

Troisièmement, il semble naturel que le mécanisme de consensus puisse également être utilisé pour déterminer les paramètres du système. Cela lui permet de s'adapter à l'évolution des conditions externes, ainsi que d'adopter de nouvelles recherches et des développements récents sur le terrain.

Si dessous, on présente æternity plus en détail, en commençant par un bref aperçu des comptes, de la monnaie d'accès, des noms et la structure des blocs. Ceci est suivi d'une explication de notre approche des canaux d'État et des contrats intelligents, puis une discussion sur la façon dont le mécanisme de consensus de la blockchain peut être utilisé à la fois pour créer un mécanisme oracle efficace et pour régir le système. Enfin, nous discutons de l'évolutivité à partir de plusieurs angles différents.

A. Monnaie, Comptes et blocks

En dépit d'être « sans-étas» du point de vue du développeur du contrat, la Blockchain d'æternity surveille plusieurs composants d'état prédéfinis. Nous allons maintenant expliquer cela, ainsi que le contenu de chaque bloc. Pour simplifier, cette section suppose que chaque nœud suit la trace de l'ensemble de la Blockchain. Les optimisations possibles sont décrites dans la section II-E.

A.1) Monnaie d'accès, æon: Utiliser la Blockchain n'est pas gratuit, mais il nécessite que l'utilisateur paie une monnaie appelée æon. L'æon est utilisé comme paiement pour les ressources que l'on consomme sur la plate-forme, ainsi que sur la base des applications financières mises en œuvre sur la plate-forme.

La distribution d'æon dans le bloc de genèse sera déterminée par un contrat intelligent hébergé sur Ethereum. De nouveaux æons seront créés par exploitation minière. Toutes les taxes sur le système sont payées par des æon, et tous les contrats intelligents s'établissent en fonction de l'æon.

A.2) Comptes : chaque compte a une adresse et une balance en æon et un nonce qui augmente la hauteur de cette dernière à la mise à jour de chaque transaction. Chaque compte doit

également payer un petit supplément pour le temps qu'il est ouvert. Les coûts liés à la création et aux transactions empêchent les spams et le gonflement de la chaine. La récompense pour la suppression des comptes favorise la récupération de l'espace sur la chaine.

***A.3)* Système de nom:** De nombreux systèmes de Blockchain souffrent d'adresses illisibles pour leurs utilisateurs. Dans le domaine du travail d'Aaron Swartz et de la Namecoin, æternity dispose d'un système de nom à la fois décentralisé et sécurisé, tout en soutenant des noms lisibles. L'état de la Blockchain comprend une cartographie à partir de chaînes uniques et lisible aux tableaux d'octets à taille fixe. Ces noms peuvent être utilisés pour indiquer des points tels que les adresses de compte sur æternity, ou des hachs d'Arbres Merkle.

***A.4)* Contenu du bloc**: Chaque bloc contient les composants suivants:

- Le hash du bloc précédent.
- Un arbre Merkle de transactions. Un arbre de comptes
- Merkle.
- Un arbre de noms Merkle.
- Un arbre Merkle de chaînes ouvertes.
- Un arbre Merkle d'oracles qui n'ont pas répondu à leurs questions respectives.
- Un arbre Merkle d'oracles ayant répondu à leurs questions.
- Un motif Merkle de Merkle.
- L'entropie actuelle dans le générateur de nombres aléatoires.

Le hachage du bloc précédent est nécessaire pour maintenir une commande de la Blockchain. L'arbre des transactions contient toutes les transactions incluses dans le bloc actuel.

À l'exception de l'arbre de vote par consensus, tous les arbres sont entièrement sous consensus : si un arbre est changé d'un bloc à l'autre, ce changement doit être dû à une transaction dans l'arbre de transaction du nouveau bloc et à une preuve de Merkle La mise à jour doit être incluse dans l'arbre de preuve du bloc. Le but des trois arbres restants devrait être clair dans les sections suivantes.

B. Chaînes d'Etat

L'un des développements les plus intéressants de l'espace Blockchain est récemment celui des canaux d'états. Ils fonctionnent selon le principe fondamental selon lequel, dans la plupart des cas, seules les personnes concernées par une transaction doivent en connaître. Essentiellement, les parties opérationnelles créent un certain état sur une Blockchain, par ex. Un contrat Ethereum ou un multi-signature Bitcoin. Ils envoient simplement des mises à jour signées à cet état entre elles. Le point clé est que l'un d'entre eux pourrait utiliser ceux-ci pour mettre à jour l'état sur la Blockchain, mais dans la plupart des cas, ils ne le font pas. Cela permet aux transactions de se dérouler aussi rapidement que les informations peuvent être transmises et traitées par les parties, au lieu d'avoir à attendre que la transaction ait été validée et potentiellement finalisée par le mécanisme de consensus de la Blockchain.

Sur æternity, la seule mise à jour de l'état qui peut être réglée sur la Blockchain est un transfert d'æon, et le seul æon pouvant être transféré est celui que les parties transactionnelles ont déjà déposées dans le canal. Cela rend tous les canaux indépendants l'un de l'autre, ce qui bénéficie que toutes les transactions liées aux canaux peuvent être traitées en parallèle,

```
1 macro Gold f870e8f615b386aad5b953fe089 ;
2
3 Gold oracle
4 if 0 1000 else 0 0 end
5 0
```

Fig. 1. Un contrat simple encodant un pari sur le prix de l'or. La langue utilisée est la Chalang, qui sera présentée dans la section IV-A.

améliorant ainsi grandement le débit des transactions.

La Blockchain n'est utilisée que pour régler le résultat final ou pour résoudre les conflits qui se posent, à peu près analogues au système judiciaire. Cependant, comme le comportement de la Blockchain sera prévisible, il n'y a pas de gain pour contester le résultat recherché d'un canal d'état; Les acteurs malveillants sont incités à se comporter correctement et à régler l'état final sur la Blockchain. Tous pris ensemble, cela augmente la vitesse et le volume de la transaction de plusieurs ordres de grandeur, ainsi que la confidentialité.

B.1) Contracts intelligents: Malgré que le seul état qui peut être réglé sur la chaîne est un transfert d'æon, æternity dispose encore d'une machine virtuelle Turing complète qui peut exécuter des "contrats intelligents". Les contrats sur æternity sont strictement des accords qui distribuent des fonds selon certaines règles, ce qui contraste fortement avec les contrats-entité de par exemple, Ethereum. Les deux des différences pratiques les plus remarquables sont que, par défaut, seules les parties impliquées connaissent un contrat donné, et seules celles ayant un canal ouvert entre-elles peuvent créer un contrat valide. Si les parties acceptent le contrat, elles le signent et conservent des copies pour référence ultérieure. Il n'est soumis à la Blockchain que si son résultat est contesté. Même dans ce cas-là le code n'est jamais enregistré dans la transaction soumise, et jamais dans aucun autre état. Si cela se produit, la Blockchain distribue les æons selon le contrat et ferme le canal.

À titre d'exemple, fig. 1 montre un contrat très simple qui code un pari sur le prix de l'or à un certain moment. Sur la ligne 1, la macro d'or sauvgarde l'identifiant de l'oracle en question, ce qui rendra vrai si le prix de l'or est inférieur à 38 $ / g le 1er décembre 2016. L'ensemble du contrat est affiché aux lignes 2 à 4 : nous Appuyez d'abord sur l'identifiant de l'oracle d'or sur la pile et appelez-le en utilisant oracle, ce qui laissera la réponse de l'oracle en haut de la pile. Nous utilisons cela pour faire une ramification conditionnelle : si l'oracle retourne vrai, nous poussons 0 et 1000 vers la pile, ce qui indique que 0 æon devrait être brûlé et que 1000 æon devrait passer au premier participant dans le canal. Sinon, nous poussons 0 et 0, le deuxième 0 indiquant que l'autre participant reçoit tout l'éon dans le canal. Enfin, nous poussons 0, qui est considéré comme le nonce de cet état de chaîne. Dans l'utilisation réelle, la nonce serait générée lors du déploiement.

Une chose importante à noter est que les contrats sur æternity ne conservent aucun état propre. Tous les états sont maintenus par les parties à la transaction et soumis comme contribution à l'exécution. Chaque contrat est essentiellement une fonction pure qui prend une certaine entrée et donne un nouvel état de la chaîne en sortie. Les avantages de

```
1 : hashlock
2 swap
3 hash
4 == ;
```

Fig. 2. Un hashlock simple.

```
1 macro Commitment a9d7e8023f80ac8928334 ;
2
3 Commitment hashlock call if
4 0 100 else 0 50 end
5 1
```

Fig. 3. Utiliser le hashlock pour envoyer æons à travers un intermédiaire sans.

l'utilisation de fonctions pures dans le développement de logiciels en général et dans le développement des applications financières en particulier ont été largement documentés dans les milieux universitaires et l'industrie depuis des décennies

a) Interaction contractuelle et contrats multi-étapes:

Même si tous les contrats sont « sans-états » et s'effectuent indépendamment l'un de l'autre, l'interaction avec le contrat et l'épanouissement peuvent encore être obtenus grâce au hashlocking. Un simple hashlock est montré dans la fig. 2. À la ligne 1, nous définissons une fonction appelée hashlock qui s'attend à ce que la pile contienne un hash h et un secret s. Il les échange sur la ligne 2, afin de hacher le secret sur la ligne 3, avant d'appeler l'opérateur d'égalité sur hash (v) et h sur la ligne 4. Cela renvoie vrai si le secret est une préimage du hash. Cette fonction peut être utilisée pour prédire l'exécution de branches de code dans différents contrats sur l'existence de la même valeur secrète.

Comme exemple d'utilisation simple, les hashlocks permettent aux utilisateurs qui ne partagent pas un canal d'état de s'evoyer mutuellement æon, tant qu'il existe un chemin de canaux entre eux a travers une tierce partie. Par exemple, si Alice et Bob ont une chaîne et Bob et Carol ont une chaîne, Alice et Carol peuvent transiter à travers Bob. Ils le font en créant deux copies du contrat figurant à la fig. 3, un pour chaque canal. L'engagement sur la ligne 1 est le hash d'un secret que Alice choisit. Sur la ligne 3, nous la poussons vers la pile et appelons la fonction hashlock. Quelle branche du si cela s'effectue dépend de la valeur de retour de hashlock. Une fois que ces contrats ont été signés par toutes les parties, Alice révèle le secret, ce qui permet à Bob et Carol de l'utiliser pour réclamer leur æon.

Le hachage peut également être utilisé pour, par exemple, Jouer à des jeux multi-joueurs dans les chaînes, comme le montre la fig. 4. Tout le monde fait une chaîne avec le gestionnaire de jeu, qui publie le même contrat à chaque chaîne. Disons que nous sommes dans l'état de jeu 32,

2 Il convient de noter que, puisque les contrats peuvent lire les réponses des oracles et certains paramètres de l'environnement, ce ne sont pas des fonctions complètement pures. Cependant, les réponses d'oracle ne changent jamais une fois qu'elles ont été fournies et peuvent être considérées comme étant dues à la richesse informatique de la machine à oracle, plutôt que d'être une impureté.
Les paramètres de l'environnement sont considérés comme un «mal nécessaire» et seront idéalement compartimentés de manière appropriée par des langages de haut niveau.

```
1 macro Commitment a9d7e8023f80ac8928334 ;
2
3 Commitment hashlock call
4 if State33 else State32 end
5 call
```

Fig. 4. Un exemple simplifié d'utilisation du hashlock pour jouer à un jeu multijoueur dans les canaux.

défini par la fonction Etas 32 (State32), et nous voulons sans faille la mise à jour simultanée de tous les canaux à l'état 33. Lorsque le responsable du jeu révèle le secret, cela fait que tous les canaux soient mis à jour simultanément.

b) Exécution mesurée L'exécution du contrat est mesurée d'une manière similaire à celle du « gaz » d'Ethereum, mais æternity utilise deux ressources différentes pour son comptage, une pour le temps et une pour l'espace. Les deux sont payés pour utiliser æon par la partie qui demande l'exécution.
Cela pourrait être considéré comme indésirable, car c'est probablement un autre parti qui cause le besoin de la Blockchain pour résoudre le conflit en premier lieu. Cependant, tant que tout l'argent dans la chaîne n'est pas utilisé pour le pari, cela peut être annulé efficacement dans le code du contrat, car il a la possibilité de redistribuer des fonds d'une partie à l'autre. En fait, il est généralement préférable d'éviter d'utiliser tous les fonds dans un canal pour traiter, car il empêche la partie perdante de coopérer lors de la fermeture du canal.

B.2) Example:
Mettons toutes ces idées à terre. En pratique, si Alice et Bob veulent effectuer une transaction en utilisant un canal d'état sur æternity, ils devront suivent la procédure suivante :

1) Alice et Bob signent une transaction qui précise combien d'argent chacun dépose dans le canal et le publie dans la Blockchain.

2) Une fois que la Blockchain a ouvert le canal, ils peuvent à la fois créer de nouveaux canaux d'états, les envoyer entre eux et les signer. Les canaux d'états peuvent être soit une nouvelle distribution des fonds dans la chaîne, soit un contrat qui détermine une nouvelle distribution. Chacun de ces états de chaîne a une identité croissante et sont signés par les deux parties, donc, si un différend ou conflit survient, le dernier état valide peut être soumis à la Blockchain qui l'applique.

3) Le canal peut être fermé de deux façons différentes :
   a) Si Alice et Bob décident qu'ils ont fini de négocier et de s'entendre sur leurs soldes finaux, ils signent une transaction en indiquant ceci et le soumettent à la Blockchain, qui fermera la chaîne et redistribuera l'argent dans le canal en conséquence.

   b) Si Alice refuse de signer une transaction de clôture pour quelque raison que ce soit, Bob peut soumettre le dernier état que les deux ont signé et demander de fermer le canal à l'aide de cet état. Cela commence un compte à rebours. Si Alice croit que Bob est malhonnête, elle a la possibilité de publier un état avec une nonce plus élevée que

les deux ont signé.
Avant le début du compte à rebours. Si elle le fait, la chaîne se ferme immédiatement. Sinon, il se ferme lorsque le compte à rebours est terminé.

C. Mécanisme de consensus

æternity utilise un mécanisme hybride de preuve-de-travail (PoW) et de preuve-de-participation (PoS) pour le consensus. L'ordre des blocs sera déterminé par la preuve de travail. Certaines variables du système seront déterminées par un système de marché de prédiction en chaîne, ce qui permet aux utilisateurs de participer et de faire connaître leurs connaissances. Pour l'algorithme PoW, nous proposons actuellement une variante du Cuckoo Cycle de John Tromp, qui est lié à la mémoire, et est également une « preuve de travail indirectement utile », car il nécessite moins d'électricité pour fonctionner, mais a un autre facteur limitant, celui de la disponibilité de latence de la mémoire. Cela rend également possible de mener la validation avec de petits appareils électroniques comme le téléphone intelligent.

Tromp écrit son travail :
"[Cuckoo Cycle] est un mécanisme de preuve-de-participation à la mémoire vérifiable instantanément qui est unique en étant dominé par la latence plutôt que par le calcul. En ce sens, le Cuckoo Cycle est une forme d'extraction ASIC où les puces DRAM servent à l'application de la lecture et de l'écriture aléatoire de milliards de bits. Lorsque même les téléphones en re-charge du jour au lendemain peuvent faire l'objet d'une mine PoW sans grandes perte d'ordre de grandeur de l'efficacité, et ceci pas avec une mentalité de rentabilité mais avec celle d' un jeu de loterie. Ainsi, le paysage du matériel minier verra une vaste expansion, profitant de l'adoption et de la décentralisation.

Prévisualisation : le mécanisme de consensus a un rôle quelque peu non-standard dans æternity. En plus d'accepter de nouveaux blocs pour la Blockchain, il accepte également les réponses aux questions d'oracle et les valeurs des paramètres du système. En particulier, le mécanisme de consensus peut se modifier. Cependant, il convient de noter que ce n'est pas tout à fait improductif. Par exemple, si un simple mécanisme de preuve de travail était utilisé, il serait plutôt peu coûteux de soudoyer les mineurs pour corrompre l'oracle. Par conséquent, æternity va utiliser un nouvel algorithme hybride de preuve de travail et de preuve de participation, en tirant parti des avantages des deux. Indépendamment de cela, PoW va être utilisé pour émettre de nouveaux jetons d'æon.
N.B.: À l'origine, æternity avait l'intention d'être une chaîne PoS a 100%. Maintenant, nous ne pensons plus qu'un système PoS 100% décentralisé soit possible.

C.1) Oracles: Une caractéristique cruciale pour la plupart des contrats, qu'ils soient codés en texte ou en code, est la capacité de se référer à des valeurs de l'environnement, telles que les prix de marchandises différentes ou si un certain événement s'est produit ou non. Un système de contrat intelligent sans cette capacité est essentiellement un système fermé et sans doute n'est pas très utile. Il s'agit d'un fait généralement accepté et il existe déjà plusieurs projets qui tentent d'amener des données externes dans la Blockchain de manière décentralisée [8]. Toutefois, pour décider si un fait fourni est vrai ou non, cela nécessite essentiellement (dans ces autres systèmes) la mise en place d'un nouveau mécanisme de consensus en dessus du mécanisme du consensus original. Exécuter deux mécanismes de consensus l'un en dessus de l'autre est aussi coûteux que de les faire fonctionner séparément. En outre, cela n'augmente en aucuns cas la sécurité, car le moins sécurisé peut toujours être attaqué et falsifié de façon à produire des valeurs "fausses". Ainsi, nous proposons de confondre les deux mécanismes de consensus en un seul (mais double), en réutilisant essentiellement le mécanisme que nous utilisons pour réaliser le consensus sur l'état du système, afin de nous mettre aussi d'accord sur l'état du monde extérieur.

Cela fonctionne de la façon suivante. Tout porteur peut lancer une oracle en s'engageant à répondre à une question par un oui ou un non. Lorsque cela se fait, ils doivent également spécifier le délai pendant lequel la question doit être répondue, qui peut commencer maintenant ou quelque temps dans le futur. L'utilisateur qui lance l'oracle est tenu de déposer une certaine somme d'æons proportionnellement à la durée du délai. Ce dépôt lui sera retourné s'il fournit une réponse qui est acceptée comme vérité, sinon le dépôt est consumé. La Blockchain génère un identifiant unique pour l'oracle qui peut être utilisé pour récupérer la réponse une fois qu'elle est disponible.
Une fois que le temps est venu pour répondre à la question, l'utilisateur qui a lancé l'oracle peut fournir une réponse gratuite. Une fois que le lanceur d'oracle a fourni sa réponse et jusqu'à qu'a un certain délais, tout autre utilisateur peut soumettre des contre-revendications en déposant le même montant d'æons. Si aucune contre-revendications n'a été soumise avant la fin du délai, la réponse fournie par l'utilisateur qui a lancé l'oracle est acceptée comme vérité et le dépôt est retourné. Si des contre-revendications sont soumises, le mécanisme de consensus pour les blocs sera utilisé pour répondre à l'oracle. Ceci est plus cher, mais comme nous savons que nous pouvons prendre au moins l'un des deux dépôts de sécurité, nous pouvons utiliser ce système.

D. Gouvernance

La gouvernance des systèmes à base des Blockchain a eu de gros problème dans le passé. Chaque fois qu'une mise à jour du système doit être effectuée, cela nécessite une fourchette-sévère (Hard fork), ce qui entraîne généralement de grandes discussions entre tous les détenteurs de valeur. Même les choses simples, comme la correction d'une variable arbitraire dans le code source, comme nous l'avons vu avec le débat sur la capacité des blocs de Bitcoin, semblent être très difficiles dans un système où les motivations des utilisateurs ne sont pas alignées avec les décideurs et où Il n'y a pas de chemin de mise à jour clair. Nous avons également vu des décisions de gouvernance plus compliquées, comme la fixation d'un seul bug de contrat intelligent dans "The DAO", a nécessité une intervention rapide des développeurs d'Ethereum.
Le problème principal de ces systèmes est facilement identifiable : le processus décisionnel d'un protocole de qualité ou de changement n'est pas bien défini et manque de transparence. Le système de gouvernance d'æternity fait partie du consensus. Il utilise les marchés de prédiction pour fonctionner de manière aussi efficace et transparente que possible.
En outre, le mécanisme de consensus est défini par un certain nombre de variables qui déterminent la manière dont le système fonctionne et qui sont légèrement mis à jour par chaque nouveau bloc. Par exemple, le coût des transactions ou des questions à l'oracle ou les modifications de paramètres fondamentaux comme le temps de création des blocks.

En ayant des marchés de prédiction sur les variables qui Définissent le protocole, les utilisateurs peuvent apprendre à améliorer efficacement celui-ci. En ayant des marchés de prédictions sur les fourchettes potentielles, nous pouvons aider la communauté à s'entendre sur la version du code à utiliser. Chaque utilisateur choisit pour lui-même quelle métrique il cherche à optimiser, mais une stratégie par défaut simple serait de maximiser la valeur de ses exploitations.

E.Évolutivité

E.1) Arbres Sharding: L'architecture présentée jusqu'ici est hautement évolutive. Il est possible d'exécuter la Blockchain même si chaque utilisateur ne fait que suivre la partie de l'état de la blockchain qui le concerne et ignore les données de tous les autres. Au moins une copie de l'état est nécessaire pour que les nouveaux utilisateurs soient certains du sous-état dont ils se préoccupent, mais nous pouvons réduire ces données de manière arbitraire sur de nombreux nœuds afin que la charge de chaque nœud soit arbitrairement faible. Les Arbres Merkle sont utilisés pour prouver qu'un sous-secteur fait partie de l'état [11]. Il est facile d'imaginer un scénario où certains nœuds se spécialisent dans le suivi des arbres et sont ainsi payés pour les inserts et les recherches.

E.2) Clients légers : les clients légers ne téléchargent pas les blocs entiers. D'abord, l'utilisateur donne à ses clients un hash dans l'histoire de la fourchette qu'ils préfèrent, une technique aussi connue sous le nom de faible subjectivité [12]. Ensuite, le client télécharge des fourches qui incluent un bloc avec ce hash. Le client ne télécharge que les entêtes des blocs. Ces entêtes là sont beaucoup plus petites que les blocs pleins ; Très peu de transactions sont traitées. Pour simplifier, nous n'avons pas mentionné les en-têtes des blocs lors de la discussion de la structure des blocs dans la section II-A.4, mais ils contiennent les éléments suivants :
• Le hash du bloc précédent.
• Le hash de racine de tous les arbres d'état.

E.3) Canaux d'Etat et parallélisme : Les chaînes d'État ont un débit immense et la plupart des transactions à l'intérieur ne sont jamais exécutées ou même enregistrées sur la Blockchain. En outre, les canaux n'écrivent pas sur un état partagé sur la chaîne, de sorte que toutes les transactions qui enregistrent réellement sur la chaîne de bloc peuvent être traitées en parallèle. Étant donné que la plupart des matériels de consommation vendus aujourd'hui ont au moins quatre noyaux de traitement, cela a un effet immédiat que le débit des transactions est multiplié par environ un facteur de 4.
En outre, le fait qu'il n'y ait jamais d'interaction simultanée complexe suggère que le résumé de cette architecture de blockchain devrait être relativement simple. Étant donné que les fragments de Blockchain sont encore assez expérimentaux, nous avons délibérément choisi de ne pas poursuivre de techniques de déchiquetage dans la conception initiale d'æternity. Cependant, si cela change à l'avenir, æternity devrait être l'une des Blockchain les plus faciles à compresser.

E.4) Transactions par seconde et exigence de mémoire donnée : les variables qui définissent le protocole sont constamment mises à jour par consensus. À partir de leurs valeurs par défaut initiales**,** nous pouvons calculer le taux de défaut initial de Transactions par seconde.

```
1 Notez qu'il s'agit d'un brouillon qui sera probablement
2 changé.
3
4 Nous définissons les variables suivantes pour les calculs
  suivants:
5
6 B = block\_size in bytes
7 F = blocks\_till\_finality
8 R = time\_till\_finality in seconds
9 T = transaction size in bytes
10 transactions per second = B * F / (T * R)
11 B = 1000000 bytes = 1 megabyte per
12 block
13 F = 24*60*2 blocks per day
14 R / F = 30 seconds per block
15 R = 24*3600 seconds per day
16 T = 1000 bytes per transaction
17
18 1000000 * 24*60*2 / 1000 / 24*3600
19 = 1000000 / 1000 / 30
20 = ca. 32 transactions per second (fast
21 enough to sign up every human within 8
   years)
```

Pour faire fonctionner un nœud, nous devons conserver une copie de tous les blocks depuis la genèse, et nous devons pouvoir enregistrer 100 fois plus d'informations, au cas où il y a une attaque. L'estimation de cette finalité est de 2 jours, alors il y aurait 5760 jusqu'à des blocs de finalité. Donc, l'exigence de mémoire est de 5760 * un mégaoctet
* 100 = 576000 méga-octets = 576 gigaoctets. 5,76 gigaoctets pour stocker les blocs.

III. Applications

La nature « sans-états » des contrats intelligents d'æternity permet de construire facilement les applications suivantes sur la Blockchain d'æternity. Il est particulièrement adapté aux cas d'utilisation à fort volume.

A. Exigences de la Blockchain

Les exigences de la Blockchain sont des primitives nécessaires comme l'æon, les portefeuilles, les noms et les concepts connexes. Ils modulent les composants réutilisables qui peuvent être utilisés comme fondations d'application et peuvent être améliorés.

A.1) Identités: chaque compte aura un numéro d'identification unique associé. Les utilisateurs peuvent enregistrer aussi des noms uniques et à la racine Merkle d'une structure de données. La structure de données peut contenir son identifiant unique ainsi que d'autres informations sur son compte. Nous visons à utiliser le format JSON de Schema.org pour représenter des personnes ou des entreprises [13].

A.2) Portefeuille: un portefeuille est un logiciel qui est utilisé pour interagir avec æternity. Un portefeuille gère les clés privées pour l'æon et crée et signe des transactions. On peut utiliser le portefeuille pour envoyer des transactions dans les canaux et utiliser des applications dans les réseaux de canaux.

A.3) Preuve de l'existence: un type de transaction permet la publication du hash de toute donnée. Les participants au système peuvent utiliser les entêtes pour prouver que les données existent à un certain moment.

B. Applications des Canaux de l'État

Les contrats intelligents dans les chaînes d'État sont parfaits pour les micro-services sur le Web qui requièrent un débit élevé de transactions.

B.1) API payant: la plupart des API existantes aujourd'hui sont publiquement disponible pour n'importe qui, ou bien ils sont sécurisés par un nom d'utilisateur et mot de passe. Les Blockchain permettent un nouveau type d'API, où l'on paie chaque appel à l'API, éventuellement toutes les requêtes HTTP. Payer pour accéder à une API résout ainsi les problèmes de DDoS. Il est aussi plus facile de créer des API de haute qualité qui sont toujours disponibles. Les API qui nécessitent un paiement sont fondamentals pour la création de types d'entreprises encore impossibles maintenant et peuvent jouer un rôle important dans l'émergence de l'économie décentralisée et l'Internet-of-things. Ils créent des motivations pour les propriétaires d'informations afin de rendre les données autrement privées, publiques et accessibles.

B.2) Crowdfunding assurés : Nous pouvons mettre en œuvre des Crowdfunding assurés en utilisant des contrats d'assurance dominante. Ce sont des contrats intelligents qui servent à collecter de l'argent pour un bien public, comme un nouveau pont, une école ou un marché.

Les contrats d'assurance dominante diffèrent de sites comme Kickstarter, en ce sens qu'ils constituent une stratégie dominante pour participer. Si le bien n'est pas financé, tous les participants obtiennent leur investissement plus un intérêt, de sorte qu'ils sont assurés contre la réduction de leur liquidité sans recevoir le bien. À l'aide d'une oracle, nous pouvons nous assurer que le fournisseur du bien ou du service est payé uniquement si le bien ou le service est effectivement fourni.

B.3) Echange atomic inter-chaine :

Les swaps permettent des échanges sans confiance d'æon pour bitcoins ou tout autre crypto-monnaie [14], [15]. Ceux-ci peuvent être implémentés à l'aide d'un hashlock, qui verrouille les transactions sur les deux Blockchain sous la même valeur.

B.4) Réplication de portefeuilles et d'actifs de value stable:

Nous pouvons utiliser des contrats intelligents pour programmer des actifs synthétiques qui suivent le prix qu'un bien réel. Par exemple, nous pourrions créer un jeton qui reste au même prix que l'or. Les dérivés synthétiques sont créés dans des paires égales et opposées. Pour qu'un utilisateur ait un jeton qui se déplace avec le prix de l'or, un autre utilisateur devra avoir un jeton qui se déplacera inversement au prix de l'or. Par exemple, Alice pourrait conclure un contrat avec Bob pour qu'Alice possède 1 gramme d'or. Alors un gramme de valeur d'or d'æon ira à Alice, et l'argent restant va à Bob. Le contrat ayant une date d'expiration à laquelle, le prix de l'or sera mesuré et les fonds redistribués à Alice et Bob en conséquence.

B.5) Contrats d'événement: les contrats d'événements sont payés lorsqu'un événement se produit et ne le sont pas lorsqu'un événement ne se produit pas, selon le rapport de l'oracle. En plus d'être intéressants en eux-mêmes, ceux-ci peuvent être utilisés par plusieurs applications différentes :

a) Assurances: nous pouvons utiliser les contrats d'événement pour implémenter des assurances. Par exemple, les billets coûteux pour les événements musicaux peuvent devenir inutiles si le temps est mauvais. Cependant, si l'acheteur du billet est remboursé si l'oracle décide qu'il a plu le jour de l'événement, son investissement est ainsi protégé

Les agriculteurs s'intéressent souvent au nombre total de pouces de pluie pendant une saison. Nous pouvons les assurer contre le flétrissement de leurs plantes.

b) Dénonciation:

Les contrats d'événements peuvent également être utilisés pour inciter la révélation d'informations sensibles. Par exemple, nous pourrions parier sur l'événement « L'information indiquant que la Société A à utilisé des pesticides illégaux a été diffusée avant le 24 janvier 2017 ». Toute personne ayant accès à de telles informations serait incitée à parier tout d'abord que l'événement se produira et à le publier.

B.6) Marchés de Prédiction:

Un marché de la prédiction fonctionne en permettant aux utilisateurs de parier si un événement futur se produira. Du prix des paris, nous pouvons prédire la probabilité future [3], [8], [16]. Ils sont le moyen le plus précis de mesurer l'avenir à un prix donné. Une fois l'événement arrivé, le marché est réglé à l'aide de l'oracle. Comme indiqué dans la section II-D, nous pouvons par exemple utiliser les marchés de prédiction pour prédire quelles mises à jour du logiciel seront bénéfiques ou nuisibles. Nous pouvons également les utiliser pour estimer le nombre de candidats à une élection, et les mensonges et les promesses non-fondées peuvent être détectés plus facilement.

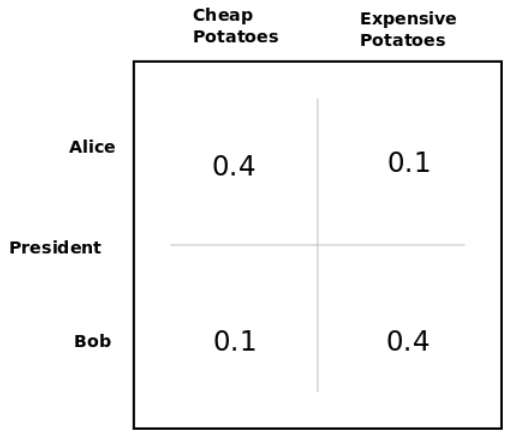

Fig. 5. Marchés de prediction multidimensionnels

a) Marchés de prédiction multidimensionnels:

Les marchés de prédiction nationaux nous permettent de prédire la corrélation entre les événements futurs possibles. Ainsi, par exemple, on pourrait prédire que si Alice est élue leader, le prix des pommes de terre descendra, et que si Bob gagne, le prix augmentera. On pourrait apprendre que si Google utilise le plan A pour les 3 prochains mois, cela gagnera probablement plus d'argent et que s'il utilise le plan B, cela gagnera probablement moins. Ou, comme dans la fig. 5, on peut voir que si Alice serait élue présidente, il y a une forte probabilité que le prix des pommes de terre soit plutôt faible.

B.7) Marché avec négociation de lots à un prix unique:

Là deux approches sont disponibles pour les attaquants qui souhaitent voler des æons à partir d'un marché. Ils peuvent profiter d'un marché divisé dans le temps, ou ils peuvent profiter de la division dans l'espace.

• Si le marché est divisé dans l'espace, l'attaquant arbitre. Il effectue simultanément des métiers dans les deux marchés à la fois afin que son risque s'annule et qu'il gagne un profit.

• Si le marché est divisé dans le temps, alors l'attaquant fait face au marché. Il lit les transactions sur le marché et crée des commandes d'achat et de vente immédiatement avant et après.

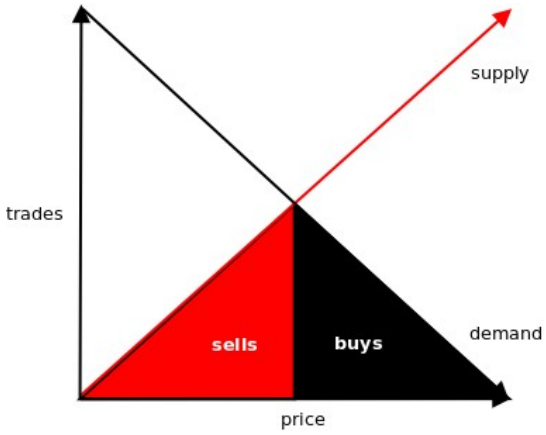

Fig. 6. La ligne noire est la courbe de la demande, la ligne rouge est la courbe d'approvisionnement. Les ventes en rouge sont de la même taille que les achats en rouge. La ligne verticale est le prix choisi par le market maker. Tout le monde disposé à acheter à un prix plus élevé échange à ce prix, et tout le monde prêt à vendre à un prix inférieur négocié à ce prix.

Pour combiner les marchés dans l'espace, chacun devrait utiliser les mêmes créateurs de marché. Pour combiner les marchés dans le temps, nous devons faire des transactions par lots, à un prix unique. Le créateur du marché doit s'engager envers chaque personne sur le prix qu'il a décidé, et si quelqu'un peut trouver des engagements contradictoires avec le créateur du marché, tous ses clients devraient être en mesure d'éliminer tous ses canaux. Si le créateur de marché s'engage à un juste prix, il combinera le même volume d'acheteurs et de vendeurs, comme l'indique le fig. 6 montre. Sinon, il se retrouvera dans une situation similaire à la fig. 7, prenant ainsi un grand risque.

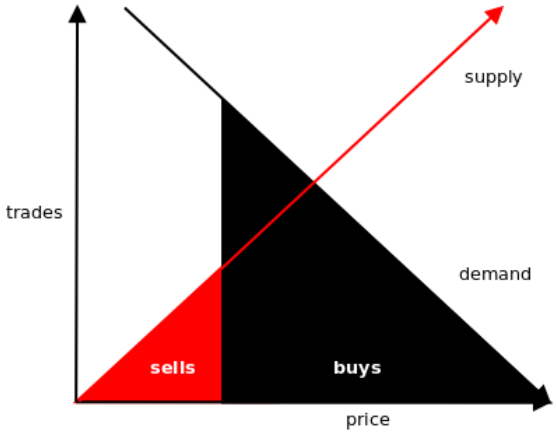

Fig. 7. Le noir est beaucoup plus grand que le rouge. Le créateur de marché vend de nombreuses autres actions qu'il acheté, ce qui pose beaucoup de risques.

IV. LA MISE EN OEUVRE

La plupart des concepts clés ont déjà des implémentations de preuve de concept dans Erlang. Cela comprend la Blockchain elle-même, la langue de contrat et VM, l'oracle et les mécanismes de gouvernance, ainsi qu'une ancienne version du mécanisme de consensus. Nous avons utilisé Erlang / OTP car il est facile d'écrire un code qui peut répondre à de nombreuses demandes en parallèle et ne se bloque pas. Les serveurs les plus performants au monde sont basés sur Erlang. Il a été utilisé pour des applications industrielles depuis 30 ans, se révélant être un produit fiable et stable.

A. Langue de la machine virtuelle et du contrat

La machine virtuelle est basée sur un empilement et est similaire au langage de script « Forth and Bitcoin », bien qu'en comparaison avec celle-ci, elle est plus riche. La VM prend en charge les fonctions au lieu de gotos, ce qui rend sa sémantique relativement simple à analyser. Une liste des opcodes de la VM peut être trouvée sur notre Github3.
De plus, il existe une langue quatrième de niveau supérieur appelée Chalang, qui compile au bytecode pour la VM. Elle prend en charge les macros et les noms de variables, mais conserve le modèle d'exécution basé sur l'empilement [17]. Des exemples de code Chalang peuvent également être trouvés sur notre Github4.

B. Adoption par intégration web

Le Web est la plate-forme d'application la plus populaire. Nous fournirons des outils de développement Web faciles à utiliser, tels que JS-bibliothèques et JSON-API pour les fonctionnalités principales de æternity.

C. Modules Open Source

Pour être facilement réutilisé pour le consortium de Blockchaine privé et d'autres cas d'utilisation, le logiciel sera écrit dans des modules sous licence MIT, tels qu'un module de consensus adapté à des besoins spécifiques.

D. Usabilité et conception UX

L'interaction humaine sans friction sera l'objet de nos efforts de développement. Plus précisément, nous nous assurons que celui qui contrôle l'identité, les clés et les transactions est clairement établi. De plus, offrir un accès facile via les passerelles Web constituera un axe central du développement futur. La participation à des marchés de prédiction via une interface mobile de type Tinder (glisser à gauche / droite) et des portefeuilles Web simples qui peuvent être facilement intégrés dans un site Web à travers un iframe seront la nouvelle norme.

V. DISCUSSION

Nous avons expliqué comment créer un système de transfert de valeur fondamentalement plus efficace. Le système décrit est en fait une machine oracle globale qui peut être utilisée pour fournir des services décisionnels à l'échelle mondiale. En particulier, toutes les applications proposées dans la section III peuvent être construites facilement et efficacement au-dessus d'æternity.

[3]https://github.com/aeternity/chalang/blob/master/opcodes.md
[4]https://github.com/aeternity/chalang/tree/master/examples

Cependant, notre approche comporte à la fois des limites fondamentales et des possibilités d'amélioration. Celle-ci sont discutés ici bas.

A. Limitations et compromis

Bien que nous croyions que les compromis réalisés dans notre architecture sont raisonnables compte tenu de l'augmentation de performance qui en résulte dans d'autres domaines, æternity n'est pas une solution attrayante pour les applications décentralisées. Il devrait plutôt être considéré comme un complément synergique aux technologies existantes. Il existe plusieurs réserves dont il faut tenir compte.

A.1) Etat à la chaîne : en dépit d'un grand nombre d'avantages, le manque d'état programmable d' æternity ne permet pas d'utiliser des applications nécessitant un état personnalisé. Par exemple, cela inclut les DAO tels qu'ils sont généralement conçus, les systèmes de noms personnalisés et les sous-courants qui ne sont pas liés à la valeur d'un élément sous-jacent.

A.2) Problème d'option gratuit : Si Alice et Bob ont un canal et Alice signe un contrat, elle lui donne essentiellement une option gratuite quand elle l'envoie : Bob peut choisir de signer et de retourner (c.-à-d. Activer) le contrat à n'importe quel Temps dans le futur. Souvent, ce n'est pas ce qui est prévu. Pour éviter ce problème, les contrats de canal ne sont pas immédiatement activés avec le montant total. Ils sont répartis dans le temps ou dans l'espace. Les deux participants s'inscriraient au contrat en petits intervalles, de sorte qu'aucun utilisateur n'offre une option gratuite importante à l'autre.

Par exemple, si les parties souhaitent parier 100 æon, elles peuvent s'inscrire à elle en 1000 étapes, chacune augmentant le pari de 0,1 æon. Cela nécessiterait environ 1000 messages à passer, 500 dans chaque direction, ce qui est assez économique car le contrat n'est jamais soumis à la Blockchain. Comme un autre exemple, si l'on voulait créer un actif financier qui durerait 100 jours, on peut s'inscrire en 2400 étapes d'une heure chacune. Cela nécessiterait environ 2400 messages à passer,1200 dans chaque direction.

A.3) Pertes de liquidité et topologies des chaînes d'états: Lors de la composition de canaux utilisant des hashlocks comme démontré dans la section II-B.1, tous les intermédiaires doivent verrouiller au moins deux fois plus d'æon que transmis par eux. Par exemple, si Alice et Carol veulent transiter à travers Bob, Bob agira comme Carol en interagissant avec Alice et vice versa.

Étant donné que cela coûte cher pour Bob, il serait très probablement rémunéré à titre de compensation. Si Alice et Carol s'attendent à mener de nombreuses transactions entre elles, ceci peut être évité en créant un nouveau canal et en transférant sans faille les contrats actifs vers le nouveau canal en utilisant un hashlock.

Pourtant, en gardant un canal supplémentaire ouvert, on s'attend à ce que la liquidité soit négative. Passer par les intermédiaires devrait être souhaitable dans de nombreux cas, surtout dans les cas où les parties ne s'attendent pas à échanger beaucoup dans le futur. Ainsi, une topologie économique de canaux où certains utilisateurs gagnent de l'argent en jouant le rôle d'intermédiaires, et en transmettant sans faille des transactions entre d'autres utilisateurs devrait apparaître.

Il convient de noter que cela ne constitue pas un seul point d'échec (single point of failure), car nous ne donnons pas notre confiance à ces émetteurs de transactions. Si un émetteur est hors-ligne avant que le secret d'un hashlock a été révélé, la transaction ne se déroule pas. Si elle ne se déconnecte plus tard, le seul effet "négatif" possible est que l'émetteur ne peut pas réclamer son æon (sa commission).

B. Travaux futurs

Il existe plusieurs façons possibles d'améliorer l'architecture actuelle.

B.1) Langue de Contrat Fonctionnel:

Une direction future raisonnable serait d'expérimenter avec des langages de haut niveau qui adhèrent plus étroitement au paradigme fonctionnel. Garder la trace d'un empilement implicite est généralement susceptible d'erreurs et peut-être ne convient pas à une langue de haut niveau qui serait adaptée aux développeurs. Cela devrait être assez simple étant donné que les programmes sont déjà des fonctions pures (modulo certaines variables d'environnement) et simplifieront grandement le développement et la vérification formelle des contrats. Si cela est fait, il pourrait également être judicieux de réviser la Machine Virtuelle pour être étroitement couplé à la nouvelle langue, afin de rendre la compilation moins propice aux erreurs et moins dépendante de la confiance en les développeurs. Idéalement, la traduction du langage de surface au code Machine Virtuelle serait simplement une transcription directe de la recherche évaluée par des pairs, mais des concessions pragmatiques devront probablement être faites.

B.2) Chaînes multipartites:

Actuellement, tous les canaux sur æternity sont limités à deux parties. Alors que les canaux multipartites peuvent être réalisés de facto par le biais d'un hash, cela peut être coûteux. Par conséquent, nous prévoyons étudier la possibilité d'ajouter un support pour les canaux n-parties, avec un mécanisme de règlement m-de-n.

GLOSSAIRE

Blockchain : Une base de données distribuée et inviolable avec accès métré. La base de données est définie par une liste croissante de blocs liés au hash et peut avoir n'importe quelles les règles et paramètres à ajouter.

æon : Un æon représente une unité de compte et un droit d'accès à la Blockchain æternity. Il est transférable.

Transaction : Un message d'un utilisateur vers la Blockchain. C'est ainsi que les utilisateurs peuvent utiliser leur devise « æon » pour accéder à la Blockchain.

Canaux d'Etas : Une relation entre deux utilisateurs enregistrés sur la Blockchain. Ils permettent aux utilisateurs d'envoyer des æons et de créer des contrats intelligents sans confiance entre eux. Ceux-ci sont appliqués et réglés par la Blockchain**.**

Hash : Un hash prend comme entrée un binaire de n'importe quelle taille et donne une sortie de taille fixe. La même entrée donne toujours la même sortie. Compte tenu d'une sortie, il est impossible de calculer l'entrée.

Hashlocking : C'est ainsi que nous connectons des paires de canaux pour créer des contrats intelligents qui impliquent plus de 2 personnes. Un secret est référencé par son hash. Lorsque le secret est révélé, il peut mettre à jour plusieurs chaînes en même temps.

Gouvernance : Un processus bien défini de prise de décisions pour le(s) protocole(s) futur(s) de la Blockchain.

Oracle : Un mécanisme qui fournit à la Blockchain les faits sur le monde réel dans lequel nous vivons. En utilisant les utilisateurs d'oracle, on peut prévoir le résultat des événements, à l'extérieur du système Blockchain.
Valeur-titulaire : Un utilisateur possédant un æon ou un dérivatif financier dans le système.
Validateur : Un validateur est un utilisateur qui participe au mécanisme du consensus. Dans le cas d'æternity, chaque détenteur de valeur peut participer.

REMERCIMENTS

Merci à Vlad, Matt, Paul, Dirk, Martin, Alistair, Devon et Ben pour la correction des épreuves. Merci à ceux-ci et beaucoup d'autres personnes pour des discussions approfondies.

REFERENCES

[1] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008. [Online]. Available: https : / / bitcoin.org/bitcoin.pdf.

[2] V. Buterin, “Ethereum: A next-generation smart contract and decentralized application platform,” 2014 [Online]. Available: https : / / github . com / ethereum/wiki/wiki/White-Paper.

[3] P. Sztorc, “Market empiricism,” [Online]. Available: http://bitcoinhivemind.com/papers/1_ Purpose.pdf.

[4] M. Liston and M. Ko¨ ppelmann, “A visit to the ora- cle,” 2016. [Online]. Available: https : / / blog . gnosis.pm.

[5] C. Detrio, “Smart markets for smart contracts,” 2015. [Online]. Available: http://cdetr.io/smart- markets/.

[6] Namecoin wiki, 2016. [Online]. Available: https : //wiki.namecoin.org/index.php?title= Welcome.

[7] P. Snow, B. Deery, J. Lu, et al., “Factom: Business processes secured by immutable audit trails on the blockchain,” 2014. [Online]. Available: http : / / bravenewcoin.com/assets/Whitepapers/ Factom-Whitepaper.pdf.

[8] J. Peterson and J. Krug, “Augur: A decentralized, open-source platform for prediction markets,” 2014 [Online]. Available: http : / / bravenewcoin . com / assets / Whitepapers / Augur - A - Decentralized - Open - Source - Platform - for-Prediction-Markets.pdf.

[9] A. Swartz, “Squaring the triangle: Secure, decentralized, human-readable names,” 2011. [Online]. Available: http : / / www . aaronsw . com / weblog / squarezooko.

[10] T. Hvitved, “A Survey of Formal Languages for Contracts,” in Formal Languages and Analysis of Contract-Oriented Software, 2010, pp. 29–32. [On- line]. Available: http : / / www . diku . dk / hjemmesider / ansatte / hvitved / publications/hvitved10flacosb.pdf.

[11] R. C. Merkle, “Protocols for public key cryptosystems,” in IEEE Symposium on Security and Privacy, 1980.

[12] V. Buterin, “Proof of stake: How I learned to love weak subjectivity,” 2014. [Online]. Available: https : / / blog . ethereum . org / 2014 / 11 /

[13] 25 / proof - stake - learned - love - weak - subjectivity/. “Schema.org schemas,” 2016. [Online]. Available:

[14] http://schema.org/docs/schemas.html. “Atomic-cross-chain-trading,” 2016. [Online]. Avail- able: https : / / en . bitcoin . it / wiki / Atomic%5C_cross-chain%5C_trading.

[15] “Interledger,” 2016. [Online]. Available: https:// interledger.org/.

[16] K. J. Arrow, R. Forsythe, M. Gorham, et al., “The promise of prediction markets,” Science, 320 2008 [Online]. Available: http : / / mason . gmu . edu / ˜rhanson/PromisePredMkt.pdf.

[17] Z. Hess, “Chalang,” 2016. [Online]. Available: https://github.com/aeternity/chalang.