Equipe

	Language de base préféré	Autres languages	Refactor / Rewrite (NXT)	Fuseau horaire	Temps de travail estimé (h/s)
Jaguar0625	*	C++, C#, JavaScript, Java	Rewrite	UTC-05:00	10-20
BloodyRookie	Java	JavaScript, C++	Refactor	UTC+01:00	10-20
Thies1965	Java	JavaScript, C, (Objective-C)	Refactor	UTC+01:00	8
borzalom	Java	php, C++, JS, ASM	Rewrite	UTC+01:00	10-20
gimre	Java	C++, python, lua (php if must)	Rewrite	UTC+01:00	10-20
Makoto (誠)	Python	Java, Lua	Rewrite	UTC+08:00	8
UtopianFuture	N/A	N/A	N/A	UTC-05:00	N/A
minusbalancer	*	C#, C++	Rewrite	UTC+01:00	10-20

Tableau 1: Equipe de base

Il nous est également possible de déléguer une partie du travail de développement à une entreprise de sous-traitance n'ayant pas d'expérience dans le domaine des crypto-monnaies mais ayant une grande connaissance de Java.

La plupart des membres de l'équipe de base et de l'entreprise de sous-traitance utilisant Java, il est ainsi logique de développer NEM en Java.

Code de base

Le répertoire github NEM est disponible à l'adresse suivante : https://github.com/NxtEM/nem.

Afin de ne pas reproduire l'erreur qu'a fait NXT, le code source de NEM devrait être préalablement testé dans le but de garantir son fonctionnement et de permettre à la communauté de l'appréhender.

Plan de haut niveau

Noyau de la Blockchain

NEM produira 1 bloc par minute, soient 1440 blocs par jour.

Blocs

Structure des blocs NEM.

Champ	Description	Taille
Taille	Nombre d'octets du bloc	4 octets
Version	Version du bloc	4 octets
Bloc précédent	Hachage du bloc précédent	32 octets
Hachage du bloc	Hachage du bloc en cours	32 octets
Horodatage	Moment (temps UNIX) où le bloc a	4 octets

	été crée	
ID de forge	ID du compte qui a forgé le bloc	52 octets
Preuve du forgeron	Preuve que le bloc est signé par la clef privée du forgeron	*
Nombre de transactions	Nombre de transactions durant le bloc	4 octets
Transactions	Transactions	*

Tableau 2: Structure des blocs

Types de transactions

Types de transactions admises par NEM

Champ	Description	Valeur
Transfert	Transfert NEM	0x1001
Nouvel actif	Délivrance d'un nouvel actif NEM	0x2001
Demande d'actif	Placer une demande d'actif	0x2002
Offre d'actif	Placer une offre d'actif	0x2004
Envoi de message	Envoyer un message sécurisé	0x4001
Transaction instantanée	Envoyer un hachage instantané	0x8001

Tableau 3: Types de transactions

Transaction

Structure des transactions de base.

Champ	Description	Taille
Taille	Nombre d'octets de la transaction	4 octets
Version	Version de la transaction	4 octets
Type	Type de transaction	4 octets
Frais	Frais de la transaction	8 octets
Date d'expiration	Moment (temps UNIX) où la transaction devrait expirer si elle n'est pas éxécutée	4 octets
Expéditeur	ID de l'expéditeur de la transaction	52 octets
Preuve de l'expéditeur	Preuve que la transaction est signé par la clef privée de son expéditeur	*
Destinataire	ID du destinataire de la transaction	52 octets
Données variables	Dépend du type de transaction	*

Tableau 4: Structure des transactions

Données variables des transactions de transfert

Données variables lors d'un transfert.

Champ	Description	Taille
Montant	Montant de la transaction	8 octets
Message	[Optionel] Message associé à la transaction	*

Tableau 5: Données variables lors d'un transfert

Données variables des transactions d'actifs

Données variables lors d'un échange d'actifs.

Champ	Description	Taille
Montant	Montant de la transaction	8 octets
Prix d'exercice	Prix d'exercice des actifs (0 pour le placement d'un ordre sur le marché)	8 octets
Nom de l'actif	Nom de l'actif	*
Description de l'actif	[Optionel] Description de l'actif	*

Tableau 6: Données variables des transactions d'actifs

Données variables des transactions instantannées

Données variables lors d'une transaction instantanée.

Champ	Description	Taille
Hachage	Hachage de la dernière transaction instantannée	*

Tableau 7: Données variables des transactions instantannées

Compte

Structure des comptes NEM qui sera sérialisée en blocs de transactions instantannées.

Champ	Description	Taille
Taille	Nombre d'octets de la structure	4 octets
Version	Version du compte	4 octets
ID	ID du compte	52 octets
Balance	Solde (confirmé) du compte	8 octets
Balance non confirmée	Solde (non confirmé) du compte	8 octets
Etiquette	Etiquette (optionelle) du compte	*

Tableau 8: Structure du compte

Notes techniques

- Toutes les structures comprendront des champs dédiés à la taille et la version afin de permettre une extension future. Les caractéristiques de taille variable auront une longueur prédéfinie.
- Les frais requis pour une transaction seront générésen conformité à une formule les définissant, il n'est donc pas nécessaire de gonfler la blockchain avec des frais inutiles.
- Les champs de preuve sont similaires aux champs de scripts de signature de Bitcoin (https://en.bitcoin.it/wiki/Transaction#Data) en ce qu'elles sont composés d'une signature et d'une clef publique.
- De la même manière que pour Bitcoin, les champs de hachage seront doublement hachés, à la seule différence que NEM utilisera le hachage SHA-3 d'un hachage SHA-256 au lieu de deux

hachages SHA-256 dans le cas de Bitcoin. Cette méthode plus sécurisée demanderait à trouver deux vulnérabilités dans deux fonctions de hachages différentes pour être compromise.

Numéro de comptes

Les numéros de compte NEM seront générés aléatoirement de la même manière que les numéros de compte Bitcoin. L'algorithme ECDSA sera utilisé pour générer les pairs clef publique / clef privée. Une clef publique ECSDA donnée, la procédure suivante la convertira en numéro de compte :

1. Effectuer le hachage SHA-3 de la clef publique
2. Effectuer le hachage RIPEMD-160 du résultat de #1
3. Ajouter les octets de la version devant le hachage RIPEMD-160
4. Effectuer le hachage SHA-3 du résultat de #3
5. Effectuer le hachage SHA-3 du résultat de #4
6. Créer la somme de contrôle de l'adresse à partir des quatre premiers octets de #5
7. Enchaîner #3 et #6
8. Encoder #7 en Base32

Notez que la procédure est plus ou moins identique à celle suivant laquelle les adresses Bitcoin sont générées (https://en.bitcoin.it/wiki/Technical_background_of_Bitcoin_addresses.) La seule différence est la dernière étape à laquelle l'adresse binaire est convertie en une chaîne. Bitcoin utilise un encondage en Base58 alors que NEM utilise un encodage en Base32.

L'encodage en Base32 permettra la création d'adresses insensibles à la casse, ce qui évitera aux débutants de se tromper. Le décodage et l'encodage en Base32 sont également plus rapides que les opérations en Base58, au détriment de chaînes d'adresses légèrement plus grandes.

Chaque adresse dans la blockchain aura un champ d'étiquette de chaîne optionnel. Les propriétaires d'un compte peuvent utiliser ce champ pour y associer une étiquette (non unique.) Cela devrait en outre contribuer à atténuer le risque d'envoyer de l'argent à un mauvais compte.

Divisibilité

Le montant de NEM sera stocké dans des nombres entiers de 64 bits signés (qui peuvent stocker au maximum une valeur de 2^63 - 1) en tant que micro-NEM. Par exemple, 1 NEM sera stocké comme 1 000 000 (10^6) micro-NEM, ce qui permettra donc à NEM d'être divisible en micro-NEM.

Structure des frais

Les frais de transactions minimum requis par NEM sont spécifiés dans le tableau suivant :

Action	**Frais**
Transaction < 1000 NEM	1 NEM
Transaction >= 1000 NEM	0.10% des NEM transférés
Emission d'actifs	1000 NEM
Message sécurisé < 1000B	5 NEM
Message sécurisé >= 1000B	0.50% des octets transférés

Tableau 9: Frais minimaux

L'expéditeur peut choisir de payer des frais de transaction plus importants afin que la transaction soit traitée prioritairement aux autres.

Le coût des frais NEM s'additionne. A titre d'exemple, une transaction de 2000 NEM associée à un message sécurisé de 100 octets requerra des frais minimaux de 7 NEM (2 NEM pour le transfert et 5 NEM pour le message.)

Forgeage

NEM récompensera tous les participants au maintien du réseau. Une partie de la vision d'équité de NEM est d'inciter les titulaires de compte à créer des noeuds. Cela permet de protéger le réseau contre d'éventuelles attaques de type DOS.

Dans le cas de NXT, les stakeholders ont peu d'intérêt à éxécuter plusieurs noeuds étant donné que la puissance de forgeage dépend uniquement du nombre de pièces possédées, ce qui influence et réduit le nombre de noeuds en ligne.

Afin d'empêcher la manipulation de la puissance de forge, les contraintes suivantes seront imposées aux forgerons :

- Un ordinateur ne sera autorisé qu'à forger avec un seul compte ouvert à un moment donné. Cela empêche à un ordinateur sur lequel plusieurs comptes seraient ouverts simultanément d'avoir une puissance de forge plus importante qu'un autre ordinateur sur lequel un seul compte est ouvert.
- Les pièces déposées ne seront pris en considération dans la puissance de forge du compte qu'après 1440 blocs. Cela empêche de frauder à quiconque souhaiterait augmenter sa puissance de forgeage simplement en déplaçant ses pièces de compte en compte.
- Seuls les comptes contenant plus de 1 000 NEM seront autorisés à forgés.

A FAIRE: Il y a actuellement deux possibilités différentes pour le système de forge : (1) système hybride PoS / PoN (2) PoI. Nous prendrons bien entendu en compte l'avis de la communauté avant de choisir.

Proposition 1 - Système hybride PoS / PoN

NEM récompenserait les forgerons selon l'échelle suivante :

Balance du compte	**Puissance de forgeage équivalente**
]1K, 10K]	10K
]10K, 40M[	Amount
]40M, *[	40M

Tableau 10: Puissance de forgeage

Tous les comptes ayant entre 1K et 10K NEM auront une puissance de forgeage équivalent à 10K. Au dessus de 10K, la puissance de forge augmentera proportionnellement au nombre de pièces possédées jusqu'au plafond de 40K. Au dessus de ce plafond, la puissance de forgeage restera constante.

Proposition 2 - PoI

NEM récompenserait les forgerons les plus utiles au réseau. Le système PoS standard récompense les comptes avec le plus de pièces mais n'incite que peu à soutenir le réseau. Le système PoN récompense tous les noeuds connectés de façon égale même si certains sont plus importants que d'autres. Ni l'un ni l'autre n'encourage la fluctuation des pièces.

Afin de contourner le problème, NEM propose un nouvel algorithme de forgeage nommé PoI (Proof-if-Importance) qui récompense les noeuds en fonction de leur importance pour le réseau. Chaque noeud aura un poids correspondant au nombre de jours depuis lequel chaque pièce est contenue dans le compte. La puissance de forge est un facteur du poids du noeud ainsi que du poids des transferts qui le lient aux autres noeuds. Les transactions avec des noeuds plus importants seront plus lourdes que les transactions avec des noeuds moins importants.

Stockage de la blockchain

NEM intégrera une blockchain consolidée afin d'accélerer la synchronisation de la blockchain initiale. Pour ce faire, le réseau NEM présentera deux types de noeuds : les noeuds de serveur et les noeuds normaux.

Les noeuds de serveur stockeront un historique complet de la blockchain. La blockchain complète servira de registre public, ce qui permettra ainsi de conserver les fonctionnalités d'explorateur. Les noeuds de serveur recenseront tous les noeuds normaux et fourniront un aperçu de la blockchain.

Les noeuds normaux seront connectés au réseau et demanderont un aperçu de la blockchain aux noeuds de serveur. Ces aperçus contiendront entre autre les informations concernant la balance de chaque compte ainsi que les ordres en cours, ou plus généralement les données de tous les blocs confirmés Après avoir téléchargé l'aperçu, le client devra synchroniser tous les blocs crées entre sa création et le moment de se connecter au réseau. Ce dernier pourra alors intéragir avec le réseau NEM.

Cela devrait minimiser le temps nécessaire à un client ouvert depuis peu pour synchroniser la totalité de la blockchain et intéragir avec le réseau.

Un aperçu sera généré quotidiennement. Lorsqu'un nouvel aperçu est confirmé par la blockchain, une transaction sans frais dédiée est alors envoyée à un compte spécial. Les nouveaux clients peuvent valider l'aperçu le plus récent en utilisant le hachage de la dernière transaction envoyée au compte spécial. Les aperçus seront validés après 1440 confirmations.

API

- account/new
  - Créer un nouveau compte
  - Input: {}
  - Output: { accountId }
- account/unlock
  - Débloquer le compte
  - Input: { proofOfAccount }
  - Output: { balance }
- account/lock
  - Bloquer le compte
  - Input: { proofOfAccount }
  - Output: {}
- account/setLabel
  - Définir l'étiquette du compte
  - Input: { proofOfAccount, label }
  - Output: {}

- account/list
    - Lister tous les comptes
    - Input: {}
    - Output: { arrayOf { accountId, label, balance } }
- account/details
    - Lister les détails et les informations du compte
    - Input: { accountId }
    - Output: { balance, unconfirmedBalance, feesEarned, feesPaid, isForging }
- account/transactions
    - Lister toutes les transactions associées au compte
    - Input: { accountId }
    - Output: { arrayOf { transactionId } }
- transaction/new
    - Emettre une nouvelle transaction
    - Input: { <transaction-data> }
    - Output: {}
- block/details
    - Lister les détails d'un bloc
    - Input: { blockId }
    - Output: { <block-details> }
- block/listIds
    - Lister les 1440 ID des blocs suivant un bloc spécifié
    - Input: { blockId }
    - Output: { arrayOf { blockId } }
- block/listMilestoneIds
    - Lister les ID de tous les blocs importants après le bloc spécifié
    - Input: { blockId }
    - Output: { arrayOf { blockId } }
- block/list
    - Lister tous les blocs après le bloc spécifié
    - Input: { blockId }
    - Output: { arrayOf { <block-data> } }
- peer/new
    - Créer une nouvelle pair à partir du noeud
    - Input: {}
    - Output: {}
- peer/list
    - Lister les informations de toutes les pairs
    - Input: {}
    - Output: { arrayOf { address } }
- node/status
    - Lister les informations concernant le noeud
    - Input: {}
    - Output: { <node-details> }
- node/summary

- Afficher la blockchain consolidée
- Input: {}
- Output: { <snapshot information> }

A FAIRE : Cette liste d'API n'est qu'un projet. Plus de détails sont à venir et plusieurs modifications seront faites durant la phase de développement.

Client

minusbalancer a accepté de créer un client officiel pour NEM basé sur le client NXT posté ici : https://bitbucket.org/minusbalancer/dotnxt-client/wiki/Home. Le client sera écrit en C# et supportera la forge.

A FAIRE : Cette section est toujours en cours de discussion et de conception. Plusieurs mises à jour seront apportées.

Explorateur de la blockchain

NEM a établi un partenariat avec abuelau, l'auteur de l'explorateur de la blockchain NXT MyNxt.info (http://www.mynxt.info/blockexplorer/.) abuelau travaillera avec l'équipe de base de NEM pour créer un explorateur de la blockchain NEM similaire à MyNxt.info.

Pièces colorées

NEM supportera un échange d'actifs décentralisé, l'émission d'actifs ainsi que la possibilité de placer des ordres d'offre et de demande. Les frais d'émission et d'échange n'ont pas encore été établis.

A FAIRE : UP travaille sur un catalogue recensent tous les cas possibles pour le support des pièces colorées. Si toutefois le plan ne supporte pas certains cas d'utilisation critique qui pourraient survenir, nous mettrons à jour la blockchain pour les supporter.

Messagerie sécurisée

Le réseau NEM comportera une messagerie cryptée sécurisée entre les titulaires de comptes à l'aide d'un échange de clefs publiques et privées. Les titulaires de compte souhaitant envoyer un message sécrisé signeront ce dernier avec leur clef privée et le crypteront avec la clef publique du destinataire. Seul ce dernier aura la possibilité de décrypter le message en utilisant sa clef privée.

Bien que la blockchain et la structure des frais de NEM supportent des messages de taille arbitraire, la première version de NEM la limitera à 1KB en raison des ressources trop importantes nécessaires au delà.

Historique des révisions

Date	Description
02/09/2014	Version initiale

Tableau 11: Historique des révisions