

# RASMART

## WHITE PAPER

Ver 1.0

## 内容

### 1. ラストマルトシステムの特性

### 2. イントロダクション

#### 2.1 問題と解決策

#### 2.2 プロジェクトにおけるブロックチェインの役割

### 3. 定義

#### 3.1 ネットワークノード

#### 3.2 最後に保存されたブロック

### 4. ネットワークのコンセンサス

#### 4.1 コンセンサスタイプの比較

#### 4.2 主なネットワークノードのコンセプト

#### 4.3 ネットワークノードの機器

#### 4.4 コンセンサスを得ること

#### 4.5 元帳の形成と開始

#### 4.6 元帳で記録されていないトランザクション

### 5. トランザクション処理

#### 5.1 トランザクション

#### 5.2 コンセンサスの形成

#### 5.3 トランザクション処理

#### 5.4 元帳の構造

#### 5.5 ラスマルトの元帳の構造

#### 5.6 ブロックのサイズ

#### 5.7 パートナーのサーチ

#### 5.8 データ転送チャネル

#### 5.9 システム内のアクション

#### 5.10 確認へのトランザクションの追加

#### 5.11 トランザクションのコスト

### 6. ラスマルトの構造

#### 6.1 ネットワークの構造

#### 6.2 ネットワークノードの特定とアドレス指定

#### 6.3 ネットワークの構成

#### 6.4 DHT

#### 6.5 PEX

#### 6.6 トラフィック

#### 6.7 元帳の構造

#### 6.8 リポジトリとハッシュ木

#### 6.9 トランザクションチェーンとブロックチェーン

#### 6.10 均一性についてネットワーク同期化

## 7. データベースエンティティ

- 7.1 ウオレット
- 7.2 トランザクション
- 7.3 ランダムなリエンティティ
- 7.4 スマート契約
- 7.5 トランザクション
- 7.6 「フォーク」と言う問題の解決
- 7.7 意思決定ブロック
- 7.8 コンセンサスの実施
- 7.9 意思決定ブロックの並列化
- 7.10 トランザクショングラフの分割
- 7.11 料金
- 7.12 料金計算方法

## 8. スマート契約

- 8.1 Luaインタプリタ
- 8.2 コストのエクスプレッション
- 8.3 データソース
- 8.4 API

## 9. セキュリティと可能な脅威

- 9.1 仲介者の攻撃
- 9.2 5 1 %攻撃
- 9.3 RSA暗号鍵のハッキング
- 9.4 シビル攻撃
- 9.5 二重支払い問題
- 9.6 トラフィックの捕捉と偽装
- 9.7 ローカルリポジトリの変更

## 10. 実施計画

- 10.1 プロゼクト実施技術計画
- 10.2 ICO
- 10.3 ラスマルト暗号通貨

## 1. システムの特性

ラスマルトと言うプラットホームがブロックテインのテクノロジーに基づいています。 ラスマルトと言うプラットホームが、分散型元帳、暗号化通貨、自己実行スマート契約に基づく金融サービスの適用性を強化しています。

今日は、ブロックテインと言うテクノロジーが急速に発展していますので、たくさんあらゆる要件が現れています。ブロックテインベースのプラットホームであるラスマルトが、マーケットの要件を満たす一連の改善と拡張方法を提供しています。

**1.トランザクションの速度。** 一秒でシステム内で実行されたすべてのトランザクションの数を示す特性

**2.使用の柔軟性。** ラスマルトは、サードパーティサービスとの統合向け柔軟で適応性のある**API**を提供します。

**3.安全。** **0.2秒**のブロック時間とコンセンサスの構成のおかげで、さまざまな攻撃や脆弱性が遮断する可能性が高くなります。

**4.自分のスマートな契約。** ラスマルトというシステムのスマートな契約は、時の試練を経た**LUA**というプログラミング言語の使用のおかげで高速です。その利点の中には高速実行とは多くの演算リソースを必要としない小さなインタプリタです。その結果、大規模なトランザクションキャパシティーと高速実行+様々な機能を持つ定期的に更新された**SDK**という特性を持つ。

## 2. イントロダクション

最初のブロックチェーンプラットフォームであるビットコインは、監督機関なしに、アセット交換が行われる参加者間の新しい分散化されたインターラクション方法の発展のために提供されたプラットホームです。 したがって、ビットコインはユーザーの中で間もなく人気を集めました。 ビットコインネットワークでは、唯一なオペレーション方法が使用されています。仲介なしでユーザー間のアセット交換です。

エーサリアムはスマートな契約に基づいて動作する分散化されたオンラインブロックチェーンベースのサービスをほとんどすべて作成する可能性を与えるプラットフォームです。

しかし、最初のブロックチェーンベースのプラットフォームに関するユーザーの認識はあまり高くなく、熱狂的なファンの数が少ないしかいませんでした。 さらに、トランザクションの速度が低かったから、多くのユーザーを引き付けることができませんでした。 後で出現したすべてのブロックチェーンプラットフォームは ビットコインのプラットホームに基づかれた。

今日、参加者間の分散化されたインターラクションに対抗しようとするのは金融業界だけです。しかし、技術的および組織的な意見から見れば、一般の銀行よりも分散化された金融サービスシステムの作成する方が簡単です。 すべての必要な経験と知識はもう獲得されました。

したがって、分散化された元帳に基づく最大限に快適な分散型金融サービスシステムを作成するには、次のことが必要です。

- 1. 最小限のオペレーションコストと共に、高速データ処理（毎秒数十万トランザクション）。**
- 2. 分権化された金融サービスを提供するために、すべての参加者と要素を統合することができる、最大限にユーザーフレンドリーな統一されたシステムを作成する必要があります：**ラスマルトプラットフォームは、不換紙幣決済センター、ユーザーパーソナライゼーション、クレジットビューロー、暗号通貨キャッシング・アウトサービスなどを提供するプラットホームです。

ラスマルトは、金融サービスのすべての参加者を統合し、分散された元帳コンセプトに基づいて超高速かつ安全なトランザクションを実行できる統一された分散型テクノロジープラットフォームです。 特殊なシステムと自己実行スマートな契約により、独自のソリューションを作成し、さまざまな金融商品間のインターラクションを確立することができます。 ラスマルトは、技術的（低速）および 財務制限のために、分散型元帳の使用不可能金融およびその他の分野において、ブロックチェーン・プロジェクトの使用を可能にします。

## 2.1 問題と解決

ネットワーク遅れは、多くのブロックチェーンプロジェクトが直面する主な問題の1つです。 ビットコインのトランザクションの平均時間は10分以上です。 同時に、銀行トランザクションは瞬間に実行され、1秒以上かかることはありません。

### 技術問題

技術問題の中には次の問題があります:

**キャパシティ** : 一定量のデータを処理するのに必要な時間です。  
**トランザクションの速度**

**レイテンシー** : 予想された時間に対するシステムの応答の時間

**保存されたデータ量 – ネットワークで保存されたデータ量**  
(エーサリアムは常に数十ギガバイトを得ています)

**コスト高** – トランザクション実行の主なパラメタです。特に 主な二つのトランザクションです。

1. グローバルネットワークで様々なモノを統合させるモノモノのインターネット内のオペレーション
2. 日常生活におけるコーヒーの購入、スーパー・マーケットでの食料の購入、あらゆる種類のマイクロローンなどのトランザクションのコストが低いマイクロペイメント。

上記の問題と制限は、金融分野を含めて多くの分野でのブロックチェーン技術の発展を防いでいます。

ブロックチェーンベースのテクノロジは、以下の特性を持つランダムなデータベースの作成可能性を与えます。

**アベイラビリティ** : 分散型データベースは、すべてのネットワークノードに同時に保存されます。 つまり、ネットワークと接続できない場合、すべての情報にアクセスできるのは1つのノードだけです。

**ユニフォーミティー** : ブロックテーブルは偽装を不可能にし、偽装の試みと場所を素早く検出させます。 .

**機密性**。 非対称のデータ暗号化のおかげで、情報の受信者だけがそれにアクセスすることができます。 電子署名は、情報が実際に送信者に属することを保証します。

**信頼性**。 各ノードにデータベースの保存のおかげで、障害時にすべての情報を回復できます。 ネットワークは、損傷を受けたノードが1つであっても、ネットワークが完全に回復できます。

**スケーラビリティ**。 ノードの追加を通じて、ネットワークのサイズを素早く簡単に増やすことができます。 ネットワークに接続するには、記録して、現時点で存在するノードに接続するのが十分です。

**インターフェース。** APIのおかげで、各ソフトウェアを分散型データベースに統合できますし、そのソフトウェアがその中のデータを受信して保存するようになる。

上記の特性のおかげで、データベースの適用範囲が金融業界を超えていきます。これは、保護された情報の交換、電子文書フロー、会計オートメーション、およびさまざまなデータ収集の普遍的な方法です。そのアプリケーションは、多くの参加者とプロセス自動化のための情報保存の最高のセキュリティを必要とするあらゆる業界で人気を集める予定です。

## 2.2 プロジェクトでのブロックチェーンの役割

ブロックチェーンは、接続された情報ブロックのチェーンです。そのような各ブロックは、チェーンの前のブロックの暗号化ハッシュ関数を含みます。二つのブロックだけがある場合（例えば、チェーン内の最初と最後）それらの間のすべてのブロックが100%に検証されているに違いない。

これに基づくデータベースは、新しいバリューを記録し、残りのバリューを変更せずに、変更の連続した履歴を形成しますし、これによって、情報を追加する可能性を与えます。情報の削除が不可能です。このように、データベース内のアクティビティが検出され、その信頼性がチェックされます。これは、元帳を開くには最適です。

しかし、オープンなデータベースさえも、機密情報を保存する機能を提供します。受信者の公開キーによる非対称暗号化は、プライベートキーの所有者だけがそれにアクセスできるようにします。RSA暗号化アルゴリズムのおかげで、キーの数が十分です。ネットワークの参加者の残りの部分は情報の追加の事実しか見ることができませんが、情報にアクセスすることはできません。

プラットフォームは、分散型元帳の保管、保護、および同期化のための技術的メカニズムをユーザーに提供します。情報保管の構造と形式の制限はありません。RSAは、これを、元帳に追加されたデータの接続と信頼性に関する情報だけで補完します。

## 3. 定義

**システム** - トランザクションの処理、保存、転送、スマートな契約の実行、条件の確認を行う分散化されたネットワークノードの組み合わせです。サードパーティシステムからの要求を処理し、要求に応じて情報を提供します。

**2. ネットワークノード**とは、ネットワーククライアントのフルバージョンがインストールされ、統合されたシステムに接続され、トランザクションの実行、データの保存と転送するコンピュータです。

**3. 元帳**とは、ネットワークノードで保存された全ての確認されたトランザクションです

**4. トランザクション**とは、特定のスマートコントラクト方法の実行要求です。これがブロックチェーン内で記録されます。

**5. スマートな契約**とは、インタラクションの条件を確認して、テックするコンピュータプロトコルです。ほとんどの場合では、スマートな契約は信頼できる関係の論理をエミュレートします。分権化がスマートな契約の重要な特徴です。

**6. スマートコントラクト方法**とは、スマート契約のオペレーションのコンピュテーションと受信されたデータの元帳での記録を担当するプログラムコード

**7. 契約の当事者** - システムのユーザおよびネットワークの参加者

### 3.1 ネットワークノード

自由アクセスとノードコネクションに基づく分散化されたインディペンデントネットワークを作るには、目的に応じて複数のノードタイプを同時に使用します。

1. 通常のノードはトランザクションの妥当性のチェックプロセスで参加しますが、信頼性が低いものです。、その他にネットワークにおけるノードの役割の次の選択ラウンドの時に、このノードが現在の処理のためのノードまたは信頼できるノードになるための キャンディデートです。
2. 信頼できるノードがトランザクションのチェックプロセスで参加しますし、重点的な信頼係数を持っています。
3. 主なネットワークノードは、通常のノードから送信されたトランザクションをアキュミュレイトして、信頼できるノードの中に配信します。
4. 記録ネットワークノードは、ラウンド中に選択された信頼できるノードの1つであります。これはロックを形成して、それをデータベースで記録して、ネットワークのすべての参加者に信送します。その後で、ネットワークのハッシュを取得して新しいラウンドを形成します。

### 3.2 最後保存されたブロック

GBL（主なブロック元帳）は、システムのすべてのノードでのブロックの元帳全体の完全に同期した状態です。

元帳ブロックの内容は、これの中に保存された情報の単位です。この情報の中は前のブロックのハッシュコードと元帳に関連するデータです。ブロックが別のノードから受信される後で、ブロック番号に応じて主な元帳に入れられます。このように、ネットワーク中のスペースを節約できます。

同期化の時に、ブロックの数だけがチェックされています。あるノードにブロックがない場合、そのブロックがダウンロードされ、ハードドライブに保存されます。

このアプローチのおかげで、システムのすべてのユニットは最新の情報のみを含んでいます。一番大きな元帳がLLと名付けられました。LLは、コンセンサスが見つかった後、元帳の形成を担当するノードによって自動的に作成されます。ブロックがシステム内のすべてのノードに送信されていますので、ノードを通じてすべての元帳が最新の情報を受信します。すべてのネットワークノードは相互に接続されており、それらの間で連続的かつ絶え間ない情報交換が行われます（トランザクションとブロック）。したがって、トランザクションのセットが後で元帳に追加されるブロックから形成されています。この間、各サーバーは、他のサーバーのための候補の提案セットを形成します。確認後、元帳に追加するかどうかという決定がされています。

その他に、元帳のデータが多数のサーバー（別個のシステムノード）に複数回保存されます。すべての情報の安全ノレーベルがとてもたかいです。システム内のノード数が多ければ多いほど良いです。

### 4. ネットワークのコンセンサス

ラスマルトのコンセンサスは、グループの意思決定の方法であり、その目的は、ネットワークの各ノードにふさわしい最終的な決定の作成にあります。

## 4.1 コンセンサスタイプの比較

コンセンサスタイプを比較するには、分散化されたラスマルト元帳の原理をせつめいしなければなりません：

### 元帳のアベイラビリティ。

すべてのノードがデータを元帳に記録し、いつでもそれを使いとることができます。

### 変更可能

ネットワーク内のすべてのノードが変更を加えることができます。

### 合意

システム内のすべてのノードが相互対応しています。一つだけ同一の元帳のバージョンを使うと意味します。

### デエビージョントレランス

1つまたは複数のノードが障害を受ける場合でも、システムのオペレーションへののがありません。

## 4.2 主ネットワークノードと記録ネットワークノードの理念

ネットワーク内のすべてのノードが分散化されており、他のノードに対して利点はありません。

必要なレベルのデータストレージセキュリティを確保し、さらなる情報の転送とトランザクション処理の速度を上げるには、ラスマルトプラットフォームは独自構造のコンバインドプロトコルを使用します。

記録ノードは、形成されたトランザクションブロックにサインし、それを元帳に保存し、その後、形成されたブロックをシステム内のすべての参加者に送信します。システム内の各参加者は、受信したブロックを自分の元帳に記録し、記録されたブロックのハッシュを記録ノードに送り返します。記録ノードは、メインノードと信頼できるノードの新しいリストを作成し、作成されたリストをネットワーク内のすべての参加者に送信して、ラスマルトプラットフォームの新しいラウンドを形成します。

## 4.3 ネットワークノードの機器

弊社の目的は、最速トランザクション処理を特徴とするプラットフォームを作成することにあります。それで、強力なサーバーとインターネットの高いキャパシティーによって、最良の状態でネットワークノードを維持するという報酬刺激を使用することを提案します。

報酬として、各ノードの所有者は、処理された元帳のトランザクションのために収集されたコンミッショングーの一部であるラスマルト暗号通貨の報酬を受けるようになります。残り部分は信頼できるノード（BFTコンセンサスの解決に参加した）の間で配布される予定です。報酬のパーセンテージが、独立したネットワークノード間の連合投票によって、ICOの後で3年以内にレート形成のシステムに設定され、変更される可能性があります。

したがって、弊社は、元帳の処理の目的で、サーバー所有者のより強力な機器の使用を促します。データプロセッシングの速度によって、サーバー所有者の報酬が違います。

## 4.4 コンセンサスを得ること

主なシステムノードはトランザクションのアキュムレータと、信頼できるシステムノードへのトランザクションの送信者の役割を果たしています。信頼できるノードはトランザクションを検証し、形成されたトランザクションブロックにサインする記録ノードを選択して、それを元帳に保存します。その後で、形成されたブロックをシステム内のすべての参加者に送信します。受け取った情報の処理とそれに基づく最新元帳の形成は、コンセンサスの達成です。そして、最新の元帳の形成の結果はコンセンサスのソリューションである

このプロセスはいくつかの段階に分けることができる:

- 主なノードのサーチ
- 信頼できるノードのリストの形成
- トランザクションリストの受け入れ、後で元帳に加えるための候補者リストの形成
- 候補リストの処理（ノード間の投票）
- 未確認トランザクションと候補リストの未確認ノードの削除
- 元帳に追加するための確認トランザクションのリストの形成
- トランザクションを含むブロックのハッシュコードの元帳へのトランザクションの追加
- ネットワークに参加しているすべてのノードにトランザクションのRブロックを送信します。ブロックが受けた後、すべてのデバイスの元帳に自動的に追加されます。

#### 4.5 元帳の形成と開始

以下にそのプロセスを記述します:

- 最後の参加者がトランザクションを形成します。
- すべての条件が満たされた場合、ユーザーはプラットフォームの特別なソフトウェアで必要な方法を使ってトランザクションを開始します。
- バリデータのコアは、同期化のすべてのステージと、元帳バージョンの不変更状態を監視します。
- コンセンサスの達成時に、すべてのトランザクションが統一されたブロックに収集されます。
- 各ブロックは、マークとハッシュコードに変換されたノード識別子からなる独自の番号を受けます。次に、ブロックがコンセンサス獲得モジュールに置かれます。
- 候補リストの作成の後、トランザクションとブロックのハッシュが元帳に記録されます。ソースの信頼性を常にチェックすることができます。
- 上記されたハッシュはユナイテッドブロックサインのタイプであり、さらに、トランザクションのブロックの作成に関する詳しい情報を持っています。
- 連合アルゴリズムのおかげでコンセンサスが獲得された後、ブロック内のトランザクションはバリデータのコアに転送され、さらに元帳に記録されます。

#### 4.6 元帳で記録されていないトランザクション

リストが含まれていないすべてのトランザクションが廃棄されます。

このイベントに関する情報は、トランザクションの開始者に瞬間的に示されています。

元帳に含まれていないトランザクションが廃棄されています。

## 5. トランザクションの処理

### 5.1 トランザクション

トランザクションは、スマート契約の形成を必要としなく、契約方法とアセットの直接転送に関する情報を持つシステムの最小ユニットです。さらに、結果はピアリングネットワークで発表されています。

### 5.2 コンセンサスの形成

弊社のシステムでは、コンセンサスソリューション形成の連合モデルが、信頼できるノードバリデータの投票の方法によって使用されます。

さらに、コンセンサスソリューション形成アルゴリズムが有限状態機械アライバルのアルゴリズムを使用します。コンセンサスこそがサイクルで動作し、各サイクルで、トランザクションはネットワークから抽出され、プールに入れられます。次に、すべてのトランザクションがレスポンスを受ける目的で信頼できるノードに送信されます。レスポンスが受け取られた場合、バリデーターを元帳に追加するため、トランザクションへのリクエストが送信されます。コンセンサスの達成と転送の合法性の完全な確認後で、トランザクションはさらに、元帳へ追加するために確認のステージに送信されています。

### 5.3 トランザクション処理

システムの分散状態を達成するために、各サーバーは元帳用の大きなデポジトリを持つだけでなく、すべてのトランザクションの本格的な生産能力の高いプロセッサーを持たなければなりません。弊社のシステムが『システムズコア』と言う概念に基づかれています。つまり、システム内の他の参加者の操作性を必要としなく、割り当てられたタスクを実行するデータプロセッサです。インプットでは、各コアが結果（ポジティブとかネガティブとかエラーなど）を出します。

主なデータセットの他に、レスポンスコードが常にシステムのコアに含まれます。このような構造は、互いに独立して動作しなければならない各プロセスの最高速度のために必要です。

### 5.4 元帳記録の構造

システム生産性の必要なレベルを達成するために、以前のブロックのハッシュコードと実行されたトランザクションのようなハッシュ木の作成を必要なく、元帳のデータベースを使用することをお勧めします。

ハッシュ木は、チェーンの独特的な識別子の受信とブロックの正しいシーケンスの回復の目的で全体データをチェックするために適用されるハッシングのタイプです。受け取った情報は、Leaf Tiger Hashの助けを借りて個々のハッシングを受ける別々のブロックに分割されます。次に、ハッシュの各ペアからInternal Tiger Hashesが計算されます。ハッシュがペアを持たない場合、これが変更されずに新しいチェーンに転送されます。このプロセッサは、1つだけのハッシュを取り出すまで行われています。

ハッシュ木は、これに基づいて結成された元帳のオペレーションを大幅に遅くします。トランザクションの速度は非常に遅くなり、確かに快適ではありません。さらに、低い最適化レベルのため、計算リソースが過負荷になる。高速を維持するためには、多数の高価なノードが必要になります。このようにデータリポジトリを使用するのが非合理的です。

## 5.5 ラスマルトの元帳の構造

弊社はハッシュ木の代わりにラスマルトシステムのトランザクションの元帳を使用することをお勧めします。

ラスマルト元帳の各レコードは、トランザクションブロックのハッシュコードで構成されます。このトランザクションを元帳の候補者リストに追加することが承認されます。さらに、ノード識別子が、形成時間と共にそのようなレコードに含まれることになります。元帳レコードには、トランザクションの方向、イニシャルおよびエンドアカウント、引き出しユニット数、入金タイプおよび入金されるユニット数があります。このようなアプローチは、トランザクション処理を大幅に高速化し、元帳への違法な変更の複雑さのレベルを高め、元帳のレコードを変更する能力を排除する能力を与えます。すなわち、情報処理が、最も高速かつ正確な手段を使って行われます。

## 5.6 ブロックのサイズ

主なノードと信頼できるノードを検索するサイクルは時間の単位であり、サイクルの長さはネットワーク自体の複雑さに基づいて計算されます。単位時間当たりのトランザクションのn個が、前の処理サイクルの終了から次の処理サイクルの開始までの時点から形成され、さらなる処理のために転送されます。ネットワークから選択されたトランザクションのn個はブロックに入られています。そのサイズは、トランザクションの総数によって違います。

## 5.7 パートナーのサーチ

ラスマルトピアリングシステムはノードがユーザアカウントとシステムで実行されるトランザクションのエッジであるグラフとして提示されます。このエッジがアカウントの二つのノードを接続します。そして、すべてのエッジが終わりと始まりを持つので、それに基づく有向グラフを構成する可能性が現れます。

恒等式として以下の条件を取りましょう：

- 実行された各トランザクションには、送信者と受信者の両方がいます
- 有向エッジのおかげで各アカウントのノードがいつも相互的に接続されます
- 各ノードは限られた有向エッジの数を持ちます

上記によって、グラフにはチェーン結成の目的でトランザクション実行のために必要な条件を満たすために、探索経路が含まれていると結論づけることができます。

単純なパスはリピートノードを持たない有向グラフのルートです。そして、グラフ自体が未知であるため、グラフ理論によれば、未知の有向グラフでルートを作成しなければなりません。このグラフのクラスでは、トラバージングの長さが $\Theta(nm)$ であり、Nがノードの数であり、Mがエッジの数であることが知られています。すべてのグラフについて、 $O(nm)$ の長さのトラバージングがあり、最小限可能トラバージングの長さが $\Omega(nm)$ であるグラフも存在する。

未知のグラフのトラバーシングは、そのトポロジーが未知であり、グラフに沿って移動すればするほど理解することができます。各ノードから出るアローが丸見えです。開始から終了までノードをフィローすることでわかつることができます。実際、これは迷路の内にあって、外出する計画を持たないロボットによる迷路のトラバージングシチュエーションと同じぐらいです。状態の数が制限されている場合、ロボットは有限状態マシンになります。このようなロボットは、テープがグラフに置き換えられ、その四角形がノードに接続されるだけでなく、グラフのエッジに直接接続される、いわゆるチューリングマシンと全く同じです。

## 5.8 データ転送チャネル

十分なネットワークセキュリティレベルを確保するために、データは暗号化された形で検証ノード間で転送されるし、ノード間の各コネクションがネットワークライブラリに基づく低レベルコネクションによって確保されます。データの転送中にエラーまたは障害が行ったら、フローは自動的に中止され、エラーの記録が最初にロギングシステムに追加されて、ログファイルに追加されます。すべてのデータは型付きヴァリアブルによって転送されます。すべてのヴァリアブルデータは、RS4と言う対称アルゴリズムによって暗号化されます。アルゴリズムのワークが共通の秘密鍵で実行され、鍵自体はノード間の接続中に暗号化された形で転送されます。

## 5.9 システム内のアクション

システム内のアクションは、あるアカウントから別のアカウントへの最も単純な支払いとかアセット転送のトランザクション、またはシステム内のコンセンサスをさらに見つけるためのバリデーターへのデータ転送です。

一つの同一の識別子を有する一つのブロック内のトランザクションの重複事を防ぐために、システム内で次の決定が行われます。システムの唯一の有効かつ正しい識別子は、最初に処理のために到着したトランザクションである。バリデーターのシステムには、あるアカウントのトランザクションがすでに実行されているという記録がありますので、コンセンサスを得る可能性が現れます。このように、二重支払い問題を解決できます。

トランザクションが実行され、バリデーターが確認の情報を受ける後、元帳の状態の変更に関するデータがリスト内のすべてのノードに自動的に送信され、その結果に元帳がシンクロナイズを始まります。いつもアクティブなトランザクションの元帳を持つために、すべてのノードの元帳の中では、着信トランザクションのシンクロナイズを実行する必要があります。

ノード。そして、このアクションを成功させるためには、シンクロナイズのために別のポートを使用する必要があります（そのような可能性がある場合）。ポートローディングの効果的な分配のおかげで、バリデータのコアに入ってくる情報の処理速度が大幅に向上します。シンクロナイズフローは連続的かつ周期的であります。そのほかに、オペレーティング メモリ分配およびCPUローディングの優先順位が平均以下です。したがって、接続されているアカウントとともに、最大1000のオペレーションがメモリに保護されます。これのおかげで、計算プロセスを最適化し、したがってリクエストへの対応速度を最適化する可能性が現れます。

## 5.10 確認へのトランザクションの追加

元帳へのトランザクションの追加が、バリデーターの後だけで可能です。特に、コンセンサスが見つかった時、元帳のトランザクションのホワイトリストが作成された時です。セキュリティを強化するために、第三者システムからのコールはブロックされます。

入力パラメータは、実行されたトランザクションに特徴を与えるオブジェクトです。結果のパラメータである `ResultValue <0` - 実行プロセスがエラーで中止され、結果バリューである `-possible error code / 0 <ResultValue` - 実行プロセスがエラーを面しておらず、結果が元帳のレコード数になります。

入力パラメータは、実行されたトランザクションのユニークなマーク、送信側および受信側についてのデータ、バリューのコレスペンドンス、転送されるアセットの量、転送されるシステム情報の量などを含むオブジェクトです。

## 5.11 トランザクションのコスト

システムではラスマルトと言う通貨が次のために使用されます：

- ・システムの使用料を支払うため
- ・システム内で通貨交換を行うため
- ・システム内でアセットを交換するため
- ・スマートな契約とのプロセスオペレーションを生成するため
- ・第三者からデータを購入するため
- ・システム内のサービスの使用料を支払うため

システム負荷のレベルによっては、トランザクションのコストが変わることがある。 ネットワークの負荷を管理するためのレバーとして、金融方法を使用することをお勧めします。

システム運営の最初の3年間は、トランザクションのコストが、実行されたオペレーションのタイプとトランザクションの数によって異なる予定です。 将来には、上記の要因に基づいて、完全自動モードでトランザクションのコストを形成するアルゴリズムを作成する予定です。

## 6. ラスマルトの構造

概念的に言えば、ラスマルトはエンドユーザー（ウォレット、モニター、オラクル）用に作成された一連のセカンダリモジュールとノード（システムの主要要素）で構成され、ノードによって提供されるAPIで動作します。

### 6.1 ネットワークの構造

ネットワークは、UDPプロトコルベースのメッセージを交換できるノードで構成されます（OSIモデルの第4レベル）。 トポロジー的に言えば、ネットワークはオーバーレイ（Ethernet上）であります。 この中はユニークなノード識別子が使用され、ルーティングがノード識別子から構成されて有向グラフと共に実行されます。 実際のネットワークインフラストラクチャ上のオーバーレイネットワークの投影とそのルーティングは、ノードのソフトウェアの転送レベルによって完全に実施されます。

### 6.2 ネットワークノードの特定とアドレス指定

受信者のパブリックキーを使う暗号化と送信者のプライベートキーでサインするために、ネットワーク内の各参加者は非対称暗号キーを使用します。 ノードの識別およびそのネットワークのラフィックが、パブリックキーのハッシュファンクションに基づいて実行されます。

Diffy-Hellmanプロトコルによる暗号化と電子署名のために、RSA（Rivest、Shamit と Adleman）アルゴリズムを、1024ビット鍵長とかECDHE（楕円曲線ディフィー・ヘルマン鍵共有）を持つ素数の分野で使用できます。 第二のケースでは、比較暗号精度のために、256ビット長の鍵が十分です

### 6.3 ネットワークの構成

ネットワークは、双方向サイクリックグラフである。 ノード識別子は、各ノードが「左側に隣人」および「右側に隣人」を有するように、丸の中でリンクされています。

メッセージを作成したノードが、すべての隣人ノードにパックを送信します。 彼らは隣人の間でパックの受取人を探します。 彼らがそれを見つけたら、メッセージを直接送信します。 もしそうでなければ、左側からパックを受け取ったノードが、右側のすべての隣人にそれを送ります。 右側からそれを受け取ったノードが、左側の隣人にそれを送ります。

トラフィック過度が、UDPの損失、メッセージ途中のノードのフォールト、およびパック偽装の試みを防止します。

この方式は、すべてのノードがいつでもUDPパックを受信できる場合だけにのみ機能を果たします。そのための主な要件は、「グレイアドレス」を持つ階層的にレイヤードとルータを持たないフラットなネットワーク、プライベートネットワークです。

NATの外部のノードは、リクエストに応答したUDPトラフィックだけを受信することができます。NATの外部のノードは、ARPルータがテーブルに記録する存在時間である期間内のみに、リクエストの送信者からUDPトラフィックを受信することができます。

ノードがそのような条件を考慮して行動を適応させる場合、ネットワークに参加することができます。入トラフィックと出トラフィックの分析を通じて、ノードがどのような行動モデルに従うべきかと理解することができます。

その分析の基準は、ノードが前にメッセージを送信したノードからメッセージを受けたのかと言ふことにあります。 そうではないと、ノードは、維持する必要のある「開いた窓」と言う期間に情報を受け取っています。

## 6.4 DHT

リングと言うオーガニゼイションは、順序付けられた方法で "隣人"を見つけ、それらの間に新しいノードを動的に含めることを可能にするハッシュディストリビューションの平等を使用します。

トレントネットワークで使用されるDHTアルゴリズムは、「メトリック」という概念を運営します。「メトリック」という概念のおかげで 異なるデータセンターや異なる大陸にあるような方法で近隣選択メカニズムを使用することができます。 これはシビルア攻撃からの保護です。 このように意思決定ブロックの全ラウンドを制御できるようになります。

逆に、ノードの応答の速度を考慮に入れると、メトリックは相互作用の速度の最適化ができます。 これに従って、近いノードによるネットワーク速度を向上させることができます。

実際、セキュリティの高いレベルとトランザクションの高速の中庸を見つけるべきです。

## 6.5 PEX

ピアクスチェンジメカニズムは、新しいノードのコネクションのためにネットワークを再構成することを可能にする。

有向グラフで構成されているため、ノードはその場所を見つけますし、その隣人がUDPの「波長さ」が同じいままであるように、隣人のサブセットを再分配します。

## 6.6 トラフィック

ネットワークはUDPレベルで動作します。 ノードによって作成されたパックは、送信者、受信者、およびコマンドのランダムなペイロードがあります。 ブロードキャストメッセージを作成するには、存在しない受信者のアドレスを設定する必要があります (0など) 。

ノードはネットワークでの登録、元帳同期化に必要な情報のリクエスト、トランザクションの作成、意思決定ブロックに参加するためのリクエストの送信、などの参加に必要なコマンドをすることができます。

メッセージを作成するノードは、これを短命と言うサインで署名します。 このようなシグネチャの特質は、TTLと言う存在時間 を持って、よりコンパクトであり、UDPパック内の多くのスペースを使用していないという事実にあります。 短いTTLを考慮に入れた後、セキュリティ問題ではありませんが、そのようなパックのチェックと処理を高速化され、エイリアンと言うパックの不正な統合は不可能になります。

短命と言うサインのチェックはネットワークレベルで行われ、拒否されたパックは最初のリダイレクトより遠くに行かず、ノードソフトウェアの上位レベルでは考慮されません。

大きなパックがいくつかの小さなパックに分割されます。これが大きなメッセージとして送信され、受信ノードのみで収集されます。 これは、キャプチャから保護するためのエクストラ手段です。

## 6.7 元帳の構造

元帳はランダムデータのベースであり、そのサンプルは各ノードにストアされ、複製されます。このデータの上では、情報の完全性と真実性と最新性を担うエクストラ構造があります。技術的に言えば、データベースは、ハッシュがランダムバリューのキーである「キーバリュー」と言うリポジトリです。GoogleによるLevelDBというリポジトリがデータベースのローカルリポジトリであります。

## 6.8 リポジトリとハッシュ木

二分木が、データのレコードの上に構成されます。このデータの各ノードが前のレベルのハッシュの対応ユニオンのハッシュと等しいです。すべての二分木がチェック可能を持ちます。許可されていない変更が行われた場合、ハッシュは変更され、他のノードで同じハッシュに対応しなくなります。これはデータベースの全体がないと言う兆候です。これはハッシュ木のモデルifikーションです。

## 6.9 トランザクションマークとブロックマーク

トランザクションチェーンは、データベースで行った変更を記述し、データベースの現在の状態に関する情報を提供するレコードのリストです。レコードは、ブロックの作成者の電子署名で署名されたブロックで結合されます。そのほかに、このレコードには一つの接続リスト（いわゆるブロックチェーン）を形成する前のブロックのハッシュも含まれます。

## 6.10 同期化と完全性のコントロール

データベースはキーバリューセットであるので、ベースの同期化はノード間の単一レコードのエクスチェンジです。トランザクションチェーンまたはハッシュ木が「同期」を停止するたびに、変更または削除されたレコードの場所が見つけられ、ネットワークからリクエストされます。これにより、最新の変更で更新するだけでなく、ネットワークにノードの長期間不存在か、またはローカルデータベースに不可逆的なデータ損失が発生した場合でも、ベースの状態の初期同期と復旧が保証されます。全体の制御は、ノードの開始時に毎回実行されます。ベースのローカルコピーの計算状態がネットワークから受信された元帳の最新の状態と同じではない場合にも行われる。各違反が同期サイクルが開始させます。

# 7. データベースの完全性

## 7.1 ウォレット

ウォレットは、ネットワーク内の参加者の個人アカウントです。参加者は、アセットを送受信するために、ローンチされたノードと電子サインキーのペアが必要です。ウォレットの識別子（およびノードの識別子）は公開鍵のハッシュです。

## 7.2 トランザクション

転送をすると、ネットワークの参加者はトランザクションを実行して、プライベート鍵で署名して、ネットワークに送信します。金融トランザクションは、送信者、受信者、転送された金額、オペレーションの通貨、ランダムなエクストラデータに関する情報を持つエンティティです。それらのトランザクションはブロックのチェーンに入り、データベースの正確性のコントロールメカニズムの一部です。

### **7.3 ランダムなエンティティ**

転送に関する情報に加えて、元帳にはランダムな情報を含むことができます。ただし、直接保存されていません。そのような各エンティティは、スマートな契約であるオーナーを持ちます。スマートな契約が元帳のデータを作成、変更、または削除することができます。

### **7.4 スマートな契約**

スマート契約はストアドプロシージャとしてデータベースにストアされます。識別子を通じてコネクトできるプログラムコードです。スマートな契約の識別子は、ソースコードのハッシュです。

### **7.5 トランザクション**

参加者によって作成されたトランザクションはまだブロックチェーンの一部ではなく、トランザクションチェーンには含まれません。その前に、その正しさをチェックする必要があります。そのような確認は約2つのステージから成っています。送信者は送信する予定のアセットを持つアセットを超える必要があります。送信者はプライベート鍵でトランザクションを署名する必要があります。チェックに失敗したトランザクションはすべて拒否されますが、破損したトランザクションはすべて、バリデータによって署名されてブロックに統合され、ブロックチェーンに含まれます。

### **7.6 「フォーク」と言う問題の解決**

ブロックチェーンでは "フォーク" に弱いです。単一の接続のリストが存在する場合、ツリーも存在する可能性があります。複数のブロックが1つの「親」ブロックを参照するのが普通です。これを次のブロックへのリンクで「閉じる」ことができません。理由はブロックが元帳に記録された後に変更される不可能ですので、2つの接続のリストで行われているということになります。他のブロックチェーンと同様に、私たちはこの問題をいつでもネットワーク上で利用できる記録ブロックを1つだけ持つことで解決します。バリデーターがバリフィケーションのラウンドの時で決定されます。

### **7.7 意思決定ブロック**

トランザクションのチェックおよび新しいブロックの保存は、意思決定ブロックによって実行されます。これは、互いに独立して検査を行い、決定結果を互いに交換するノードのサブセットです。信頼できない環境（およびエンドユーザーによって制御されるノードを持つ分散ネットワークは、デフォルトでは信頼できません）では、このようなメカニズムにより、妥協ノードを検出し、その存在に関係なく正しい結果を受け取ることができます。この方法の基礎は、半正しいデータに基づいて正しい決定をする可能を与えるビザンチンのゼネラルの問題の解決です。

### **7.8 コンセンサスの実施**

コンセンサスは、すべてのノードが受信したトランザクションの有効性に関する判断です。トランザクションに関する決定がされた後、ブロックを記録するノードが選択され、その後、意思決定ブロックが新たなノードによって再び形成され、これがすべて繰り返されます。このように、「フォーク」からの新しい保護手段が導入されます。一つだけのノードが新しいブロックをネットワークで記録します。

### **7.9 意思決定ブロックの並列化**

意思決定ブロックは一つ以上にすることができます。収集されたすべてのトランザクションがサブセットに分割されている場合、すべての準備ワークを同時に実行することができますし、サイクルごとに意思決定権が記録ノードから別の記録ノードに転送されます。このスキームでは、記録が連続的であり、ネットワークのトランザクションキャパシティが増加するであろう。

## 7.10 トランザクショングラフの分割

ノードが送信者でありおよび受信者であり、それらの間のコネクションがトランザクションの量に等しいコストを有する接続されたグラフの形式でのトランザクションのセットの表現は、グラフを最適にサブセットに分割する可能性を与えます。

このためには、各ノードの結果バランスがネガティブであるグラフが各サブセットに形成される必要があります。このタスクは、ナップサック問題の解決です。このファンクションの目的は、グラフの一般的な非負性のルールを破るトランザクションの最小限数を破棄するように、準備されたトランザクションのリストの形成にあります。このラウンドで破棄されたトランザクションが、エクストラ確認ために次のラウンドに引き渡されます。

## 7.11 料金

料金は、ネットワークの作業のメンテナンス向け、ネットワーク内の参加者が自分の計算リソースを費やすためのモチベーション方法です。

弊社は、ノードが強力な機器で動作し、高質のインターネットチャネルを持つことに興味があります。意志決定ブロックでは、より迅速に対応できるノードの料金の受取の可能性がもっと高いです。

## 7.12 料金計算方法

下の表には、料金のサイズの理由のリストがあります（送信者のトランザクションの実行料）

### 料金計算方法がラウンドによって違います

#### I. ラウンドでのトランザクション数

ラウンドでのトランザクション数が多い程、料金が少なくなる

ラウンドでのトランザクションの最小量は1です

- ・ノミナルバリューは、一つのラウンドで**10000**のトランザクションです
- ・トランザクション数の上限は、**65536**です

#### II. ラウンドでの信頼できるノード数

ラウンドでの信頼できるノード数が多い程、料金が高くなる

ノミナルバリューは、一つのラウンドで**101**のトランザクションです

#### III. トランザクションのサイズ

トランザクションが占めるディスクスペースが多い程、料金が高くなる

トランザクションのサイズがある数から他の数までです

ノミナルバリューの制限はありません。

#### IV. 送信者のトランザクション活動

送信者が実行するトランザクション数が多い程、料金が高くなる

そんなトランザクション活動の下限は0です。

ノミナルバリューは**5000～10000**です。

このようなトランザクション活動の上限は、**1310720**トランザクション/秒です。

## 8. スマートな契約

スマートな契約は、受信者がその識別子であるトランザクションの作成の時に実行されたプログラムコードです。スマートコントラクトの識別子は、ソースコードのハッシュです。

スマートな契約の発行は、ソースコードを含むトランザクションの作成です。 ラスマルトのコントラクトがデータベースに追加され、トランザクション受信者としての識別子がネットワークに入るときにトランザクションが実行されます。 実行された後、スマートな契約は、データベース内のいくつかのエンティティを作成、変更、または削除し、コール間で状態を維持します。

そのほかに、スマートな契約はブロックチェーンから情報をリードし、トランザクションを作成することができます。 その一例は、スポーツイベントの賭け金を収集するスマートな契約が、イベントに関する情報がネットワークに現れた後、形成された「銀行」全体の分配に比例して報酬を分配します。 決定のために必要な情報は、別のスマート契約によってシステムに追加されます。 したがって、通貨レート、スポーツイベントの結果、公証の事実などの世界の各事実がスマートな契約の実体になる可能性があります。

スマートコントラクトは、Luaプログラミング言語で書かれており、分離された仮想マシン内のすべての機能を使用します。 Luaの標準ライブラリはディスエーブルであり、スマートな契約は外部世界とコンタクトできません。

すべての必要な機能を、特別に書かれたSDKの枠組みの中のみでスマートな契約が受け取ります。

受信者がスマート契約の識別子であるトランザクションを受信すると、意思決定ブロックの参加者は、そのコードをエグゼキュートします。この完了までに、すべてのヴァリアブルおよび作成されたエンティティを含む仮想マシンの状態のハッシュをエクスチエンジします。スマート契約実行の結果が意思決定ブロックの各参加者の中で同じである場合、トランザクションがブロックチェーンに保存され、変更されたすべての元帳エンティティがネットワーク内で同期されます。

スマートコントラクトはアルゴリズム的に制限されていないため、ネットワーク障害が発生するリスクがあります。これを防止するために、スマートな契約が厳密に制限された実行時間が与えられ、その後でトランザクションが停止され、トランザクションが無効となります。そのようなトランザクションの料金は、とにかく請求され、意思決定ブロックの参加者間で分配されます。

スマートな契約が変更できるエンティティを、単独で開始する必要があります。それ以外のエンティティをすべてリードできますが、記録できません。

スマートな契約にはインプットやアウトプットがなく、ネットワーキングエフェクトを満たすことができません。元帳のローカルコピーからすべてのエンティティをリードだけでなく、独自のエンティティを作成、変更、および削除することだけができます。

### 8.1 Luaインタプリタ

Luaインタープリタは、プログラムが実行される分離された仮想レジストリマシンです。 インタープリタの中では、周辺オペレーティングシステムのリソースを利用できないため、すべてのライブラリ（入出力を含む）を確実に提供する必要があります。 この方法では、仮想マシンは確かに特別に書かれたSDKにしかアクセスできません。例えば、ファイルの記録やネットワークコネクションを確立するためです。

## 8.2 コストのエクスプレッション

ラスマルト暗号通貨は、1つの契約ユニットのコストであり、さらにコンセンサスを達成するために2つの絶対的に異なるユニットを比較する可能性を与えます。ラスマルトの暗号通貨は、実際に、アセット間の転送の伝導のコネクターの役割を果たします。 そのようなアプローチが可能になった理由はすべてのアセットがラスマルトの暗号通貨に流動できる事実にあります。

## 8.3 データソース

正確で完全なオペレーション、最大限の正確なチェック、エクストラ明確化する情報の提供のために、ラスマルトシステムはサードパーティのデータサプライヤを使用します。 これは、契約の当事者に関する情報がコンフィデンシャルです。 例えば、ローンの決定をするために信用情報を受け取ることです。

システムは、インテグレーションバスを呼び出すことができます。 これは、サードパーティの情報システムと連携する機能を提供します。 上記のバスは、他の参加者に関する情報の提供という形で、自ら第三者のシステムにリクエストを生成するが、これは無料ではない。 ラスマルト暗号通貨は、支払い手段として使用されます。

このようなリクエストが、情報システムによって提供されたアドレスに、暗号化された形のみで送信されます。 決定を下すのに必要な最低限の情報を含む応答が結果として考えられます。

## 8.4 API

各ノードは、ネットワーク内のリクエストを実行するためのREST APIです。 そのようなリクエストが、トランザクション作成、スマートコントラクトの発行、トランザクションリストのリード、および元帳からのバリューの受信のため使われます。 APIは、ネットワークと相互作用することを目的とするソフトウェア、および独自のローカルノードを持つ必要がある分散化された元帳に対して、ローカルのみのリクエストを受け入れます。 転送伝導のための「テーブル」と言うアプリであるウォレットは、同じワークステーションにあり、そのすぐ隣にインストールされたAPIノードを通じてネットワークとのサードパーティのソフトウェアインターフェイクションの一例です。

## 9. セキュリティと可能な脅威

### 9.1 仲介者の攻撃

送信者を認証するトランザクションの電子サインによって解決されます。

### 9.2 51攻撃

ネットワークが成長するにつれて、確認ラウンドに入る確率がとても低い。

### 9.3 RSA暗号鍵のハッキング

2048ビット（256バイト）のサイズのRSA鍵の信頼性は数学的に証明されました。 ショアアルゴリズムによる攻撃は、量子コンピュータが存在しないため不可能です。

### 9.4 シビル攻撃

妥協ノードによる100%周囲の場合のみに可能です。 正しいノードが少なくとも一つであつたら攻撃が不可能です。 このプロセスはまた、信頼できる検証ノードを選択するという点でエンタロピー要素によって複雑になります。

## 9.5 二重支払い問題

ブロック記録ノードが少なくとも1つであれば、ネットワークがフォークから保護されます。そのほかに、二重支払い問題を排除されます。

## 9.6 トラフィックの捕捉と偽装

同じものが複数回コピーされ、異なる方向に送信されます。トラフィックをブロックするには、環境全体を制御する必要があります。インターネットの場合、それは起こり難いです。短命のパックを認証するのに十分な軽量の電子サインがトラフィックの捕捉と偽装を防止します。

## 9.7 ローカルリポジトリの変更

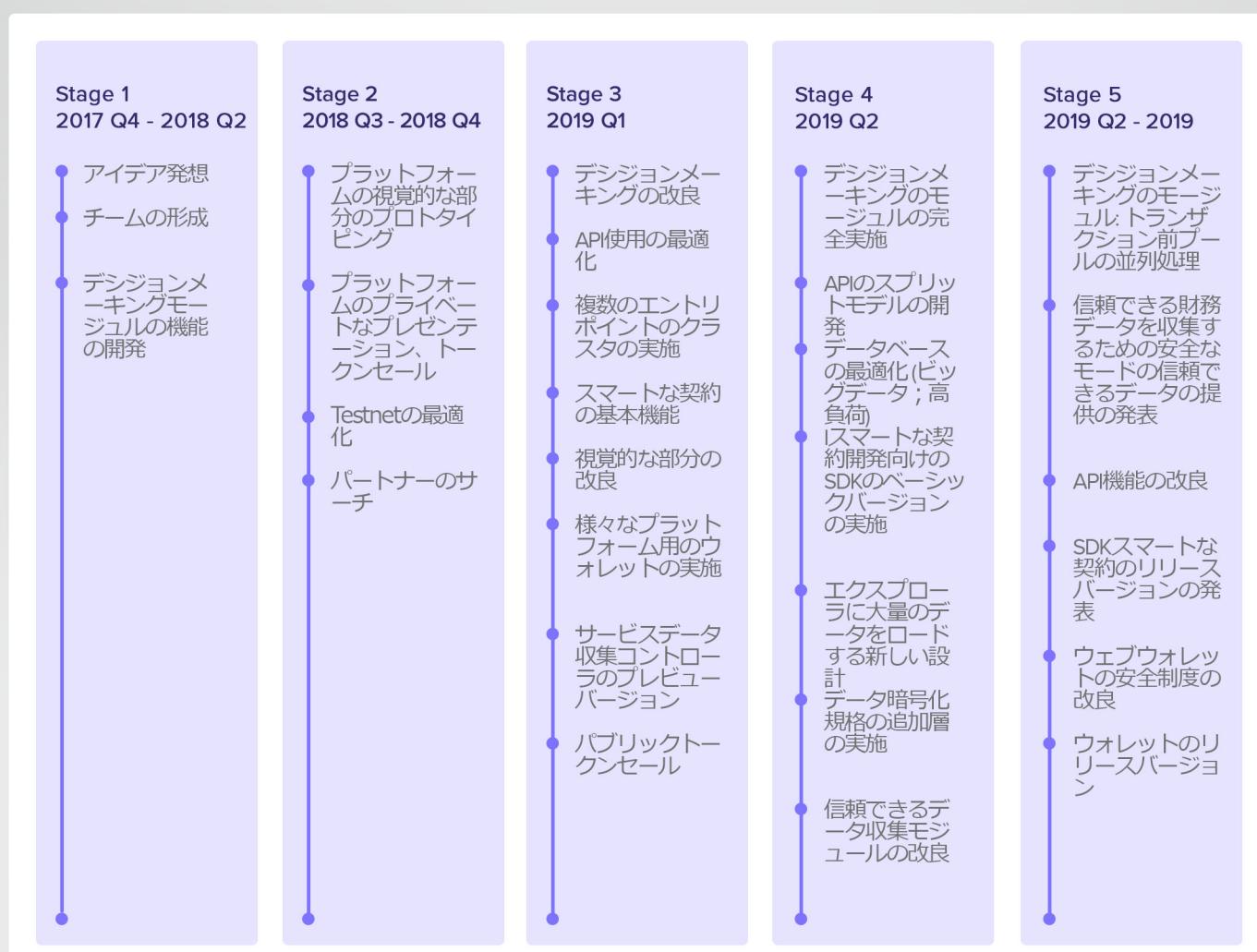
ローカルリポジトリに妨害があれば、ベースの状態の再計算につながります（Merkle Treeのルートハッシュと最新のブロック）。データが正しくない場合、ネットワークは、復元する前に危険化されたノードとのあらゆるインタラクションを中止します。

# 10. 実施計画

## 10.1 プロゼクト実施技術計画

### 10.2 ICO

### ロードマップ



# ラスマルトイニシャルコイン オファーリング

## トークン化

トークン: RAS

トークンのタイプ: ユーティリティ

## トークンフィーチャー

トークンの価格: \$ 0,23

ハードキャップ: \$ 50 000 000

エミッション: 500 000 000 RAS

最小限購入トランザクション: \$

受け入れる通貨: ETH

## パブリックセール

500 000 000 RAS - トークンの初期発行量

225 000 000 RAS - ICO

67 000 000 RAS - マーケティングとアドバイザー

90 000 000 RAS - プラットフォームの維持

59 000 000 RAS - リザーブファンド

59 000 000 RAS - チームとパートナー

## ICO スケジュール

ステージ	長さ, 時間	最低発注数量, USD	ボーナス, %*
プライベートセール	60	10 000	30
プレセール	20	5 000	20
主なセールステージ	10	500	5-10

\* – 大規模購入をしたら、すべてのステージでボーナスを受けます。トークンをチームのパーセントと弊社のファンドから取り出す予定です。

### 10.3 ラスマルト暗号通貨

システムのオペレーティングバージョンの開始後、5億RASの一定のトークン量が発行される予定です。さらに、ICO中に発行される予定ERC20トークンにRASをエクスチェンジ することができます。 エクスチェンジを、固定相場(ERC20の1トークン= 1 RAS)で実行することができる。

ICOの後、トランザクション処理の速度を毎秒400000まで向上させる予定です。

現在のスマートな契約メカニズムは、シンプルだがかなり生産的に高いLuaプログラミング言語に基づいています。そのセキュリティレベルはとても高いです。チームは、金融オペレーションのための言語セキュリティの徹底的な検査と、その仮想マシンの保護方法の調査を計画しています。

各ノードの最速ルートの検索の高速化させる第2レベルのメカニズムの開発でネットワークの速度を大幅に向上させることを計画しています。

