スワップ：
イーサリアム・トークンの
トレーディングのための
ピアツーピア・プロトコル

Michael Oved、Don Mosites
2017年6月21日

team@swap.tech

要約

我々は、イーサリアム ブロックチェーンでのイーサリアム・トークン（ERC20）のトレーディングのためのピアツーピア方式を提示する。最初に、ブロックチェーン オーダーブック（注文控え帳）の限度の概要を述べ、ピアツーピアトークンのトレーディングにおける強力な代替手段：オフチェーン ネゴシエーションおよびオンチェーン セツルメントを提案する。次に、当事者が、他の人々にトークンを取引する意図があることを合図することが可能なプロトコルについて説明する。一旦接続されると、取引相手同士の間で自由に価格について連絡を取り合い、注文を送信する。このプロセス中、当事者は、正確性を検証するために独立したサードパーティーであるオラクルから価格を以来する場合がある。最後に、イーサリアム ブロックチェーンで注文に応じるためのイーサリアム スマートコントラクトを提示する。

1) はじめに

過去12 か月にわたるイーサリアムでのデジタル資産の数は、ますます多くの使用事例がスマートコントラクトとして実行されるので、急激に増大している。この傾向は、将来まで続くというのが我々の主張である。その理由は、ユーザーが、彼らのトークン化されたポートフォリオを使用事例の間で移する、またはリバランス（調整）するので、この拡大が資産の内外へのスワップの需要を増大させると確信することによる。ブロックチェーン・オーダーブックに基づいた取引は、固有の制限がないわけではなく、その多くは本論文で概説する設計決定によって軽減される可能性がある。我々は、そのような制限を伴わずにイーサリアム・エコシステムを推進させるために、資産流動性のロックを解除しイーサリアム・エコシステムを自由にする一組のプロトコルを指定することによって、ブロックチェーン・オーダーブックの代替手段を提供することを追及する。

オーダーブック

オーダーブックは、所定の取引可能な資産の供給と需要に適合する高度に自動化できる方法を実現する。従来型では、これらは集中化され、注文の実行と組み合わされ、真実の中心点で注文が作成され、実行され、取り消されることを可能にする。非集中化の精神で、オーダーブックは、ブロックチェーンのために再設計された。しかし、ブロックチェーンでのオーダーブックの展開には、いくつかの制約が存在する。

ブロックチェーン・オーダーブックは、拡大縮小しない。ブロックチェーンでコードを実行するとコストが生じ、そのために自動化された注文-取り消し-注文のサイクルはすぐに高額になり、高性能の、自動化できるマッチングシステムとしてのオーダーブックの強みは損なわれる。実際に、そのマッチングアルゴリズムがブロックチェーンで実行されている場合、発注している当事者には、オーダーブックのサイズによって大幅に増加する実行コストが生じる。

ブロックチェーン・オーダーブックは公開されている。ブロックチェーンで注文を作成する取引は、マイナー（採掘者）によって処理されるので、それらのマイナーは、ブックに計上される前に注文を内々に知ることになる。これによって、本来の注文に重大な影響を与えるおそれのあるフロント・ラニング（先回り売買）の機会が発生する。さらに、その注文は公式に発表されるので、その注文価格は、誰にとっても同一であり、サプライヤーから個別対応する流動性を奪う。

ブロックチェーン・オーダーブックは不公正である。物理的に分散システムは、本質的に、それらのノードの間の待機時間を受けてしまう。マイナーは、地理的に分散されているので、高度な当事者は、注文を同一場所に設定し、注文を見つけ、ブロックチェーンの待機時間を上回ることができる可能性があり、他の当事者より前に注文情報によって効果的に活動する可能性がある。この情報の非対称性は、あまり慣れていない当事者がエコシステムに参加する勇気を全く損ねているおそれがある。

ピアツーピア（P2P）

それとは代わって、ピアツーピアトレーディングは、個々の当事者が、お互いに直接取引することを可能にする。日々我々が行う取引のほとんどは、ピアツーピアである。カフェでコーヒーを買い、eBayで靴を売り、Amazonでキャットフードを購入する。これらは、人々または企業の間の内密な取引なので、各当事者は、彼らが取引する相手を知っており最終的に選択する相手である。

ピアツーピア 取引は拡大縮小する。注文は、個々の当事者間で送信され、唯一のものであり、完全に満足させる保証のない公的取引での注文とは対照的に行われる。これにより、オーダーブックでの取り消しは通常の出来事となり、一方でピアツーピアでの注文は、すでに関心を示している当事者に提供されるので、多くの場合満足される。さらに、ピアツーピアでの供給と需要のマッチングは、オンチェーン、オフチェーンに関わらず、高価なアルゴリズムのマッチメークとは対照的に軽量のピアの発見を通して解決されることができる。

ピアツーピア取引は非公開である。一旦、二人の当事者が見つかり、お互いに取引するために選択されると、交渉するためのサードパーティーは必要ない。これらの当事者間の連絡は、交渉の間中内密に保たれ、他の者が注文要請行動を行う機会を奪う。注文が満足されるように送信されるときのみ、それは公知の事実になる。

ピアツーピアトレーディングは公正である。注文は、2つの当事者間で直接作成され送信されるので、外部の参加者に利点がある可能性はない。参加者は、複数の独立した当事者と連携している限り、交換を実現しようとする価格に相当する、またはそれより良い価格を得ることができる。さらに、それらの価格設定の注文は、自動化された、低待機時間取引戦略によって有利な立場を取られる心配なく、積極的に行うことができる。

ブロックチェーン・オーダーブックによって課される、拡張性、プライバシー、および公正さの制約は、代替手段を必要としてきた。今日、イーサリアム・エコシステムには、資産交換のためのオープンなピアツーピアソリューションが必要である。

スワップの紹介

スワップは、イーサリアム・ブロックチェーンでのトークンの取引のための真のピアツーピアエコシステムを促進するためのプロトコルである。以下に示すのは、効率的なカウンター・パーティーの発見および交渉をサポートする拡張可能な仕様である。これらのプロトコルは、資産取引エコシステムのための基礎となり、イーサリアム・エコシステムの成長を促進するように意図されている。本論文を公開し、考察をオープンにすることによって、我々は、多種多様な現実世界での適用を可能にする質の高いプロトコルを作成する目的で、エコシステムのステークホールダーからの意見を求める。

2) ピアのプロトコル

カウンター・パーティー（取引先）同士の間の数回のメッセージのみによって、取引は迅速、公正、かつ内密に取り決められることができる。本論文の目的において、メーカーは注文を出す当事者であり、テイカーは注文に応じる当事者である。各当事者はピアであるため、一方当事者はいつでもメーカーまたはテイカーの役割を引き受ける可能性がある。以下の仕様書は、イーサリアム・トークン（ERC20）に準拠しており、その基準を実装するいかなるトークンもこのプロトコルを使って取引することができる。

中心的なプロトコルは、以下のダイアグラムに配列されている。メーカーおよびテイカーがオフチェーンで取引交渉を行なう。以下のコントラクトは、イーサリアム スマート コントラクトであり、それによってブロックチェーンで注文に応じる準備が整うとテイカーは発信する。

グラフィックス

English	Your language
Maker	メーカー
Taker	テイカー
Contract	コントラクト

図 1：注文の要請、提示、および注文に応じる

1. テイカーは、メーカーにgetOrderを発信する。
2. メーカーは、注文によって返信する。
3. テイカーは、コントラクトにfillOrder（order）を発信する。

2.1) API（アプリケーションプログラミングインタフェース）の注文

以下のAPIは、ピアとサービスとの間での通信に使われるトランスポート-アグノスティック リモート プロシージャ コール（RPC）である。例ではアドレスの代わりにトークンのティッカーを使用するが、実際の発信にはERC20準拠トークンのアドレスが必要である。以下の発信の署名は、更に踏み込んだ技術的詳細を別の文書で公開するための検討目的としてのものである。

注文APIは、オフチェーンで、取引交渉の間、カウンター・パーティーの間で行なわれた非同期発信を

指定する。 作成者は、要請提示サイクルを同期要求応答として保存することを選択することができる。注文は、メーカーによって署名されているので、テイカーは、その後、その注文に応じるためにそれをスマートコントラクトに送信することができる。

getOrder(makerAmount, makerToken, takerToken, takerAddress)

テイカーから、メーカーに、トークンを取引するための注文を要求する発信があった。

例：*「私は、BATを使って10 GNO買いたい。」*

getOrder（10、GNO、BAT、<takerAddress>）

provideOrder(makerAddress, makerAmount, makerToken, takerAddress, takerAmount, takerToken, expiration, nonce, signature)

メーカーから、テイカーに、実行のための署名入りの注文を提供する発信があった。

例：*「私は、5 BATで10GNOをあなたに売ります。」*

provideOrder（<makerAddress>, 10, GNO, <takerAddress>, 5, BAT,
<expiration>、<nonce>、<signature>)

2.2) APIを見積もる

見積もりは、当事者間で価格情報を提示するためのもので実行可能ではない。見積もりは、その後、両方のカウンター・パーティーについて条件が一致する場合、注文に変えることができる。

getQuote(makerAmount, makerToken, takerTokens)

テイカーから、メーカーに、指定したトークンでの見積もりを要求する発信があった。

例：*「BATを使って10GNOを買うにはいくらかかりますか？」*

getQuote（10、GNO、[BAT]）

provide Quote(makerAmount, makerToken, taker Amounts)

テイカーから、メーカーに、テイカーのトークンでの見積もりを提示する発信があった。

例：*「10GNOのためには、あなたに5 BATかかります。」*

provide Quote（10、GNO、*{*BAT: 5*}*)

3) インデクサーのプロトコル

インデクサーは、今後のメーカーかテイカーかを問わず、トークンを購入または販売することを望む、ピアをその取引の意図に基づいて集めマッチさせるオフチェーンサービスである。インデクサーは、特定のトークンを購入または販売する意図に基づいて、取引する意図のあるピアを集めピアがマッチするのを助けるオフチェーンサービスである。多くの見込まれるメーカーは、取引する意図を合図することができ、テイカーが適切な取引相手を見つけることをインデクサーに要請すると、複数の検索結果が存在する可能性がある。一旦テイカーが、取引したいと思うメーカーを見つけると、彼らは上記のピアのプロトコルを使った交渉に進む。一旦メーカーとテイカーとの間で合意に達したら、その注文はスマートコントラクトで応じられる。

メーカー、テイカー、およびインデクサーの相互作用は、以下の図表で説明されている。メーカー、テイカー、およびインデクサーは、すべてブロックチェーンから離れて動作し、好適なメッセージング媒体によって通信する。

グラフィックス

English	Your language
Indexer	インデクサー
Maker	メーカー
Taker	テイカー
Contract	コントラクト

図 2：カウンター・パーティーを見つけ、取引を行う

1. メーカーは、インデクサーにadd Intentを発信する。
2. テイカーは、インデクサーにfindIntentを発信する。
3. インデクサーは、テイカーにfoundIntent(maker) を発信する。

4. テイカーは、メーカーにgetOrderを発信する。
5. メーカーは、注文によって返信する。
6. テイカーは、コントラクトにfillOrder（order）を発信する。

いくつかのメーカー、テイカー、およびインデクサーの相互作用は、以下の図表で説明されている。各メーカーは、独立してそれらの意図を通知する。テイカーは、特定の意図のあるメーカーを見つけるように要請し、インデクサーはイーサリアムのアドレスと詳細情報のリストを返す。

グラフィックス

English	Your language
Indexer	インデクサー
Maker	メーカー
Taker	テイカー

図 3：メーカーは、インデクサーにaddIntentを発信し、テイカーは、findIntent を発信する。

1. いくつかのメーカーは、インデクサーにaddIntentを発信する。
2. テイカーは、インデクサーにfindIntentを発信する。
3. インデクサーは、テイカーにfoundIntent(maker) を発信する。

一旦、テイカーが適切なメーカーを見つけると、彼らは、互いに対して各自を比較検討するために、それぞれのメーカーからの注文をリクエストする注文APIを使用することができる。テイカーが提示された注文に応じると決めた場合、彼らはスマートコントラクトにfillOrderの発信を行う。

グラフィックス

English	Your language
Maker	メーカー
Taker	テイカー
Contract	コントラクト

図 4 : テイカーは、メーカーにgetOrderを発信し、テイカーは、コントラクトにfillOrderを発信する。

4. テイカーは、いくつかのメーカーにgetOrderを発信する。
5. いくつかのメーカーは、注文によって返信する。
6. テイカーは、注文を選択し、コントラクトにfillOrder（order）を発信する。

3.1) インデクサーのAPI

インデクサーのAPIは取引する意図を管理し、それはピアの間に合図される。以下の発信は、ピアおよびインデクサーの間で行われる。

addIntent(makerToken, takerTokens)

いくらかの金額のトークンを購入または販売する意図を追加する。

例：*「私は、GNOをBATのために取引したいと思っています。」*

addIntent（GNO、[BAT]）

removeIntent(makerToken,　takerTokens)

トークンを取引する意図を削除する。

例：*「私は、もはや、GNOをBATで取引することに関心はありません。」*

removeIntent（GNO、[BAT]）

get Intent(makerAddress)

アドレスに関連付けられている有効な意図をリストアップする。

例：*「[メーカーのアドレス] が取引を望んでいるトークンをリストアップします。」*

get Intent(<makerAddress>)

findIntent(makerToken, takerToken)

特定のトークンを取引する意図のある他のユーザーを見つける。

例：*「BATでGNO を取引する他のユーザーを見つける。」*

findIntent(GNO, BAT)

foundIntent(makerAddress,　intent List)

インデクサーは、取引する意図のある他のユーザーを見つける。

例：*「BATで10GNO販売している他のユーザーを見つける。」*

foundIntent(<makerAddress>, [*{*makerAmount: 10, makerToken: GNO, takerTokens: [BAT]*}*)

4) オラクルプロトコル

オラクルは、メーカーとテイカーに価格情報を提供するオフチェーンサービスである。テイカーに配信する前に注文の価格設定をするときに、メーカーは、それが適正な価格提案と見なされるものかをオラクルに尋ねることができる。同様に、注文を受取ったとき、テイカーは、注文の価格が適正であることを確かめるためにオラクルに尋ねることができる。オラクルは、メーカーとテイカーの両方がより知識に基づいた価格決定を行うのに役立つように、そして取引交渉のプロセスをスムーズにするために価格情報を提供する。

グラフィックス

English	Your language
Oracle	オラクル
Maker	メーカー
Taker	テイカー
Contract	コントラクト

図 5：メーカーは、注文を提示する前にオラクルに問い合わせる

1. テイカーは、メーカーにgetOrderを発信する。
2. メーカーは、オラクルにget Priceを発信する。
3. オラクルは、メーカーに価格を返す。
4. 価格情報を分析した後に、メーカーは注文を提示する。
5. テイカーは、コントラクトにfillOrder（order）を発信する。

テイカーが注文を受け取るときには、テイカーとオラクルとの間に同じ相互作用が起こる。

グラフィックス

English	Your language
Oracle	オラクル
Maker	メーカー
Taker	テイカー
Contract	コントラクト

図 6：テイカーは、注文に応じる前にオラクルに問い合わせる

1. テイカーは、メーカーにgetOrderを発信する。
2. メーカーは、注文によって返信する。
3. テイカーは、オラクルにgetPriceを発信する。
4. オラクルは、テイカーに価格を返す。
5. 価格情報を分析した後、テイカーは、コントラクトにfillOrder（order）を発信する。

4.1) オラクルのAPI

オラクルのAPIは、注文価格を判断するためにメーカーとテイカーによって使用される。
価格は、
提案であって実行可能形式ではない。

getPrice(makerToken, takerToken)

テイカーまたはメーカーから、オラクルに、価格を得

るための発信があった。例：*「BATでのGNO の現在の*

価格はいくらになっていますか？」

getPrice(GNO, BAT)

provide Price(makerToken, takerToken, price)

オラクルから、メーカーまたはテイカーに、価格を提示する発信があった。

例：*「BAT でのGNOの現在の価格は、0.5です。」*

provide Price(GNO, BAT, 0.5)

5) スマートコントラクト

注文に応じるまたは取り消すためのイーサリアム スマート コントラクト。

fillOrder(makerAddress, makerAmount, makerToken, takerAddress, takerAmount, takerToken, expiration, nonce, signature)

トークンのアトミック・スワップがテイカーによって発信された。コントラクトは、メッセージの差出人がテイカーに一致していることを確認し、期限に表示されている時刻が過ぎていないことを確認する。注文に応じるために、ピアは、特定のトークンに、コントラクトが少なくとも指定された金額を引き出すことを許可する承認をすでに発信していなければならない。トークンの取引については、コントラクトはそれぞれのトークンに、transferFrom を発信する。この機能が正常に完了したときに、完了されたイベントはブロックチェーンにブロードキャストされる。

例： *「私は、10 BATで5 GNOの注文に応じることを希望します。」*

fillOrder([maker], 5, GNO, [taker], 10, BAT, [expiration], [signature])

cancelOrder(makerAddress, makerAmount, makerToken, takerAddress, takerAmount, takerToken, expiration, nonce, signature)

すでにテイカーに伝えられているがまだ応じられていない注文の取り消し。注文のメーカーから発信された。その注文は、コントラクトにすでに応じられたものとしてマークされるので、後続する注文に応じるための試みは機能しない。この機能が正常に完了したときに、取り消されたイベントはブロックチェーンにブロードキャストされる。

例：「私は、10 BATで5 GNO のこの注文を取り消すことを希望します。」

cancelOrder([maker], 5, GNO, [taker], 10, BAT, [expiration], [signature])

5.1) Etherの注文

スマートコントラクトは、トークンのためのEther（ETH）の取引に対応している。注文に無効なtakerTokenアドレス（0 x 0） が含まれている場合、スマートコントラクトは、関数呼び出しによって送信されたEtherの値を確認し、テイカーに代わってメーカーにそれを転送する。

6) 要約

スワッププロトコルは、イーサリアムネットワークで、資産交換の分散化のために需要を拡大するのに役立つ。ブロックチェーンをベースにしたオーダーブックは、我々のエコシステムの気風の中では目新しく確かである一方、それらが現在利用できる集中化されたソリューションと競争するのは最終的に困難になると確信する制限がある。スワップは、分散化され、かつこれらの制限によって影響されない方法を提供する。

このプロトコルを実装することによって、参入者は、莫大な価格設定を利用する被害を受けることなく、スケーラブルで、内密な、かつ公正な方法で、流動性を利用する権を取得する。このプロトコルおよびAPIは、拡張可能であり、我々とともにアプリケーションを築き上げるようコミュニティーに奨励する。我々は、フィードバックを歓迎し、皆さんとともにイーサリアムコミュニティーを前進させることを期待するものである。