Swap：　乙太坊代幣交易對等協定

Michael Oved, Don Mosites
2017年6月21日

team@swap.tech

摘要

我們提出了一種ERC20代幣在乙太坊塊鏈上的對等交易方法。首先，我們概述了塊鏈訂單簿的局限性，為對等代幣交易提出了一種強大的替代方案：鏈外協商和鏈內結算。然後，我們說明了一種各方可以向他人發出代幣交易意圖的協定。交易對手之間一旦建立了連接，就可以自由地協商價格和傳遞訂單。在此過程中，各方可以從獨立的協力廠商資料庫獲取價格以驗證其準確性。最後，我們提出了一種在乙太坊塊鏈上完成訂單的乙太坊智慧合約。

1) 簡介

在過去的十二個月，由於越來越多的用例是以智慧合約的方式實施的，乙太坊上的數位資產數量急劇增加。我們認為這種趨勢在未來會保持下去；因此我們認為，隨著用戶在用例之間切換或重新調整其代幣投資組合，這種增長趨勢將增加資產轉入和轉出的需求。基於塊鏈訂單簿的交易有其固有的局限，其中許多局限可以通過本文所述的設計決策來緩解。我們希望通過描述一套能夠釋放資產流動性和使乙太坊生態系統突破這些局限自由發展的協定，為塊鏈訂單簿提供一種替代方案。

訂單簿

訂單簿能夠以高度自動化的方式來對指定可交易資產的供需進行匹配。傳統的訂單簿是集中式的，並且與訂單執行相結合，以集中式的真實來源實現訂單的創建、執行和取消。雖然塊鏈訂單簿已按照去中心化的精神對塊鏈進行了重新設計，但在塊鏈上部署訂單簿仍存在著若干限制。

塊鏈訂單簿不能擴展。在塊鏈上執行代碼會產生成本，因此自動化的訂單-取消-訂單週期很快變得昂貴，削弱了訂單簿作為高性能自動化匹配系統的優勢。實際上，如果匹配演算法在塊鏈上運行，則下訂單的一方會產生執行成本，該成本隨著訂單的大小而大幅增加。

塊鏈訂單簿是公開的。因為在塊鏈上創建訂單的交易是由礦工處理的，所以這些礦工在訂單發佈到訂單簿之前就能知曉訂單。這就產生了可嚴重影響原始訂單的搶先交易的問題。此外，由於訂單是公開發佈的，訂單價格對所有人都是一樣的，這讓供應商無法調整流動性。

塊鏈訂單簿是不公平的。物理上的分散式系統在其節點之間不可避免地會有延遲。由於礦工分佈在不同的地理位置，水準高的礦工能夠協同定位，檢測訂單，超越塊鏈延遲，有效地先于其他礦工對訂單資訊採取行動。這種資訊不對稱極有可能打擊相對弱勢的礦工，可能會使他們退出該生態系統。

對等（P2P）

作為一種替代方案，對等交易可使各方直接相互交易。我們每天所做的大部分交易是對等的：在咖啡館購買咖啡，在eBay上賣鞋子，或者在Amazon上購買貓食。因為這些是個人或企業之間的私人交易，所以每一方知道並最終選擇與誰交易。

對等交易可以擴展。對等交易訂單在各方之間傳遞,一次性決定.而公共交易所中的訂單不能保證完全成交。所以訂單簿中會經常發生訂單取消的情況。而對等訂單是提供給已表達了交易興趣的各方，所以成交機會較大。此外，對等供需匹配可以通過羽量級對等點發現來解決，而非不論鏈內還是鏈外的昂貴匹配演算法。

對等交易是私密的。一旦交易雙方發現並選擇相互交易，就不需要第三方（協力廠商）進行協商。雙方之間的溝通在協商期間仍然是私密的，消除了他方對訂單請求行為採取行動的機會。訂單只有在提交以用於成交時，才會成為公開信息。

對等交易是公平的。由於訂單是直接在雙方之間創建和傳遞的，所以外部參與者不能獲取優勢。只要他们與多個獨立方合作，參與者就可以獲得與交易所達成的價格相當或更好的價格。此外，這些定價訂單可以大膽地執行，而不用擔心被自動化的低延遲交易策略所利用。

由於塊鏈訂單簿在可擴展性、隱私性和公平性方面的局限性,必須找到替代方案。今天的乙太坊生態系統需要一個開放的對等解決方案來進行資產交換。

Swap簡介

Swap是一種用於乙太坊塊鏈代幣交易的協定，它可以實現一個真正的對等生態系統。以下所述的規範是可以擴展的，支持高效地實現交易對等點發現和協商。這些協定旨在成為資產交易生態系統的基礎，加速乙太坊生態系統的發展。通過發佈本文和引發各方討論，我們希望徵求生態系統利益相關者的意見，從而制定出高品質的協議以實現各種實際應用。

2) 對等協議

由於交易對手之間傳遞的消息量較少，交易可以快速、公正和私密地進行協商。在本文中，“製造方”是提供訂單的一方，而“接受方”是完成訂單的一方。因為各方為對等方，所以任何一方都可以隨時成為“製造方”或“接受方”。以下規範中的代幣符合ERC20標準，實現了該標準的任何代幣均可使用此協定進行交易。

核心協議序列如下圖所示。“製造方”和“接受方”在鏈外進行交易協商。下面的合約是一個乙太坊智慧合約，當“接受方”準備好在塊鏈上完成訂單時調用此合約。

1

圖片

English	Your language
Maker	“製造方”
Taker	“接受方”
Contract	合約

圖 1：請求、提供和完成訂單

1. “接受方”對“製造方”調用getOrder。
2. “製造方”回應一個訂單。
3. “接受方”對合約調用fillOrder(order)。

2.1) 訂單API

以下API是在對手和服務之間進行通信的與傳輸無關的遠端程序呼叫（RPC）。示例中使用的是代幣代碼而不是位址，但實際調用需要符合ERC20標準的代幣的位址。下面的調用簽名僅用於討論，因為進一步的技術細節將在其他文檔中另外發佈。

訂單API在鏈外，它規定了對手之間在交易協商

期間的非同步調用。開發者可以選擇將請求-提供週期作為同步請求-回應處理。由於訂單是由“製造方”簽署的，因此“接受方”可以稍後將其提交給要完成的智慧合約。

getOrder(makerAmount, makerToken, takerToken, takerAddress)

由“接受方”向“製造方”調用，用於請求交易代幣的訂單。

舉例：*“我想用BAT買入10個GNO。”*

getOrder(10, GNO, BAT, <takerAddress>)

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

由“製造方”向“接受方”調用，用於提供一個簽名合約以供執行。

舉例：*“我以5個BAT的價格賣給你10個GNO。”*

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

2.2) 報價API

報價用於表示各方之間的價格資訊，不可執行。如果稍後兩個交易對手的條件都得到滿足，報價就可以變成訂單。

getQuote(makerAmount, makerToken, takerTokens)

由“接受方”向“製造方”調用，請求以特定代幣報價。

舉例：*“買入10個GNO需要多少BAT？”*

getQuote(10, GNO, [BAT])

provideQuote(makerAmount, makerToken, takerAmounts)

由“接受方”向“製造方”調用，提供用“接受方”代幣表示的報價。

舉例：*“買入10個GNO你要花費5個BAT。”*

provideQuote(10, GNO, *{*BAT: 5*}*)

3) 索引子協議

索引子是一種鏈外服務，它根據交易意圖聚合和匹配交易對手：希望購買或出售代幣的潛在“製造方”和“接受方”。索引子的鏈外服務將這些意向聚合到交易中，並根據購買或出售特定代幣的意圖說明匹配交易對手。許多潛在的“製造方”都可以表示交易意圖，所以當“接受方”要求索引子尋找合適的交易對手時，可能會有多個結果。一旦“接受方”找到一個願意交易的“製造方”，他們就接著使用上述的對等協議進行協商。一旦“製造方”和“接受方”達成協議，訂單將在智慧合約中成交。

“製造方”、“接受方”和索引子之間的互動如下圖所示。“製造方”、“接受方”和索引子都不在塊鏈上操作，並通過他們喜歡的任何消息媒體進行通信。

圖片

English	Your language
Indexer	索引子
Maker	“製造方”
Taker	“接受方”
Contract	合約

圖 2：尋找對手並進行交易

1. “製造方”向索引子調用addIntent。
2. “接受方”向索引子調用findIntent。
3. 索引子向“接受方”調用foundIntent(maker)。

4. “接受方”向“製造方”調用getOrder。
5. “製造方”回應一個訂單。
6. “接受方”對合約調用fillOrder(order)。

多個“製造方”、一個“接受方”和一個索引子之間的交互如下圖所示。每個“製造方”都獨立地宣示他們的意圖。“接受方”要求找到有特定意圖的“製造方”，索引子返回一個乙太坊地址清單和詳細資訊。

圖片

English	Your language
Indexer	索引子
Maker	“製造方”
Taker	“接受方”

圖 3：“製造方”調用addIntent，“接受方”向索引子調用findIntent

1. 多個“製造方”向索引子調用addIntent。
2. “接受方”向索引子調用findIntent。
3. 索引子向“接受方”調用foundIntent(maker)。

一旦“接受方”找到合適的“製造方”，他們可使用訂單API來請求每個“製造方”的訂單並對它們進行比較。如果“接受方”已經決定成交某個特定訂單，他們將對智慧合同調用fillOrder。

圖片

English	Your language
Maker	“製造方”
Taker	“接受方”
Contract	合約

圖 4：“接受方”對“製造方”調用getOrder，“接受方”對合約調用fillOrder

4. “接受方”對多個“製造方”調用getOrder。
5. 多個“製造方”回應訂單。
6. “接受方”選擇一個訂單並對合約調用fillOrder(order)。

3.1) 索引子API

索引子API管理在對手之間傳遞的交易意圖。以下是對手和一個索引子之間的調用。

addIntent(makerToken, takerTokens)

添加一個購買或出售一定數量代幣的意圖。

舉例：*“我想用GNO購買BAT。”*

addIntent(GNO, [BAT])

removeIntent(makerToken, takerTokens)

刪除代幣交易的意圖。

舉例：*“我不再想用GNO購買BAT了。”*

removeIntent(GNO, [BAT])

getIntent(makerAddress)

列出與一個位址相關的有效意圖。

舉例：*“列出[makerAddress]想要交易的代幣。”*

getIntent(<makerAddress>)

findIntent(makerToken, takerToken)

尋找願意交易特定代幣的人。

舉例：*“找出願意用GNO購買BAT的人。”*

findIntent(GNO, BAT)

foundIntent(makerAddress, intentList)

索引子找到了願意交易的人。

舉例：*“已找到正在出售10個GNO以換取BAT的人。”*

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

4) Oracle協議

Oracle是一種向“製造方”和“接受方”提供定價資訊的鏈外服務。在訂單交付給“接受方”前對訂單進行定價時，“製造方”可向Oracle瞭解公平價格的建議。同樣地，“接受方”收到訂單後可要求Oracle核查訂單的價格，以驗證其是否公平。Oracle提供這種定價資訊，以說明“製造方”和“接受方”制定更明智的定價決策，以促進交易協調的順利進行。

圖片

English	Your language
Oracle	Oracle
Maker	“製造方”
Taker	“接受方”
Contract	合約

圖 5：“製造方”提供訂單前查詢Oracle

1. “接受方”對“製造方”調用getOrder。
2. “製造方”向Oracle調用getPrice。
3. Oracle向“製造方”返回價格。
4. “製造方”在分析價格資訊後提供訂單。
5. “接受方”對合約調用fillOrder(order)。

“接受方”收到訂單時與Oracle之間的交互與此非常相近。

圖片

English	Your language
Oracle	Oracle
Maker	“製造方”
Taker	“接受方”
Contract	合約

圖 6：“接受方”完成訂單前查詢Oracle

1. “接受方”對“製造方”調用getOrder。
2. “製造方”回應一個訂單。
3. “接受方”向Oracle調用getPrice。
4. Oracle向“接受方”返回價格。
5. “接受方”在分析價格資訊後對合約調用fillOrder(order)。

4.1) Oracle API

“製造方”和“接受方”使用Oracle API確定訂單價格。價格僅為建議，是不可執行的。

getPrice(makerToken, takerToken)

“接受方”或“製造方”向Oracle調用以獲取價格。舉

例：*“GNO與BAT的當前兌換價格是多少？”*

getPrice(GNO, BAT)

providePrice(makerToken, takerToken, price)

由Oracle調用，向“製造方”或“接受方”提供價格。

舉例：*“GNO與BAT的當前兌換價格是0.5。”*

providePrice(GNO, BAT, 0.5)

5) 智能合約

完成或取消訂單的乙太坊智慧合約。

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

“接受方”調用代幣的原子交換。該合約確保資訊發送者與接受者匹配，並確保時間沒有超過“過期時間”指定的期限。要完成訂單，交易對手必須已經調用了特定代幣的許可權，至少允許合約提取指定數量。對於代幣交換，合約向各代幣調用transferFrom。在該函數成功完成時，會向塊鏈廣播一個“完成”事件。

舉例：*“我想以10個BAT換5個GNO以完成此訂單。”*

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

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

取消訂單的資訊已發給“接受方”但還沒有完成。由訂單的“製造方”調用。在合約中將此訂單標識為已完成，因此後續完成訂單的嘗試會失敗。在該函數成功完成時，會向塊鏈廣播一個“取消”事件。

舉例：*“我想取消10個BAT換5個GNO的訂單。”*

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

5.1) 乙太幣訂單

智慧合約支援用乙太幣（ETH）換代幣。如果合約包括一個空的takerToken位址(0x0)，智慧合約會檢查與該函式呼叫一起發送的乙太幣值，並代表“接受方”將該值傳遞給“製造方”。

6) 總結

Swap協定可以滿足乙太坊網路上去中心化資產交換的不斷增長的需求。基於塊鏈的訂單簿雖有創新且符合我們生態系統的精神，但我們認為它的一些限制會讓它們最終難以與現有的集中式解決方案競爭。Swap提供了一種既可以去中心化又不受這些限制影響的方法。

參與者通過實施該協議，能夠以可擴展、私密和公平的方式獲得流動性，而不必犧牲獲取優勢定價的機會。Swap協議和API是可擴展的，我們也鼓勵社區與我們一起構建應用程式。我們歡迎您提出回饋意見，並期待與您攜手共同推動乙太坊社區的發展。