Swap: Peer-to-Peer 방식의 이더리움 토큰 교환 프로토콜

Michael Oved, Don Mosites
6 월 21 일, 2017 년

team@swap.tech

초록

우리는 이더리움 블록체인 상에서 ERC20 토큰을 P2P 방식으로 교환하는 방법을 소개합니다. 이 백서에서는 먼저, 블록체인 오더북의 한계를 짚어보고 그 대안으로 오프체인에서 협상을 한 후 온체인에서 정산을 하는 P2P 방식의 토큰 교환 방식을 제안합니다. 그 뒤에 참여자들이 토큰 교환에 대한 의사를 전달할 수 있는 프로토콜을 서술 합니다. 이 프로토콜에 연결된 참여자들은 자유롭게 가격을 협상하고 거래를 체결할 수 있습니다. 이 과정에서 거래 참여자들은 가격 적정성과 정확성을 검증하기 위해 제 3 자 오라클에게 가격 문의를 할 수도 있습니다. 마지막으로, 이더리움 블록체인 상에서 거래를 체결할 수 있는 이더리움 스마트 컨트랙트에 대해 논의합니다.

1) 소개

스마트 컨트랙트 응용 사례들이 늘어나면서 이더리움 체인 상에 올려진 디지털 자산의 개수는 지난 12 개월 간 기하급수적으로 증가했습니다. 우리들은 이러한 트렌드가 미래에도 계속될 것이라 예상합니다. 뿐만 아니라, 이러한 급격한 성장과 함께 사용자들은 자신들의 포트폴리오를 재조정하거나 새로운 서비스를 이용하기 위해 이런 디지털 자산을 서로 교환하려는 수요도 같이 증가할 것입니다. 블록체인 오더북을 기반으로 하는 거래소들은 내재적인 한계를 지니고 있습니다. 그리고 그러한 한계들은 본 백서에서 소개하는 디자인을 사용한다면 대부분 극복할 수 있습니다. 우리는 블록체인 오더북의 대안이 될 수 있는 프로토콜을 설계하여 자산을 유동적으로 만들어주고 이더리움 생태계가 한계를 극복하고 발전할 수 있도록 만들어주고자 합니다.

오더북

오더북은 자동적으로 특정 자산의 수요와 공급을 연결시켜주는 방식입니다. 전통적으로 오더북은 중앙화로 운영되었고 주문이 생성되고, 체결되고, 취소되는 주문 집행(order execution) 기능도 단일 지점에서 제공합니다. 탈중앙화를 근간으로 하는 블록체인 상에서든 오더북이 새롭게 설계되어야 합니다. 하지만, 블록체인 상에서 오더북을 운영하는 것은 몇 가지 제약을 지니고 있습니다.

블록체인 오더북은 확장성이 떨어집니다. 블록체인 상에서 코드를 실행하는 것은 비용이 들기 때문에 자동적으로 주문-취소-주문 사이클을 운영하게 되면 코드를 실행하는 비용 때문에 자동적으로 주문을 매칭해주는 시스템으로써의 장점이 사라집니다. 주문간의 매칭 알고리즘이 블록체인 상에서 실행된다면 주문을 내는 사람의 입장에서는 오더북의 크기가 커지면 커질수록 주문을 내기 위한 비용은 더 커지는 사태에 직면하게 됩니다.

블록체인 오더북은 공개적입니다. 주문을 생성하는 트렌젝션이 채굴자들에 의해 처리되기 때문에 오더북에 입력되기 전에 채굴자들은 해당 주문을 미리 알게 됩니다. 이로 인해 선행매매(front-running) 문제가 발생하여 원래 주문에 실질적으로 영향을 줄 수도 있습니다.

이에 더해, 모든 주문이 공개되어 있기 때문에 자산의 가격은 모두에게 동일하게 적용되고, 이로 인해 자산 공급자들은 유동성을 조정할 수단을 잃게 됩니다.

블록체인 오더북은 불공정합니다. 물리적으로 분산화 되어 있는 시스템은 노드 간의 지연 속도(latency)가 발생합니다. 채굴자들 또한 지역적으로 퍼져있기 때문에 인근에 있는 참여자들끼리 주문에 대한 정보를 먼저 접하고 지연 속도로 인해 다른 참여자들이 주문에 대한 정보를 접하기 전에 선제적으로 대응을 할 수도 있습니다. 이런 정보의 불균형으로 인해 생태계에 새로 진입하고자 하는 참여자들에게 진입 장벽이 높아질 수도 있습니다.

Peer-to-peer (P2P)

이에 대한 대안으로, P2P 교환 방식은 참여자들끼리 직접 거래를 할 수 있도록 해줍니다. 우리가 카페에서 커피를 사거나, eBay 에서 물건을 사거나, 아마존에서 고양이 사료를 사는 등 일상 생활에서 겪는 대부분의 거래가 이런 직접 거래입니다. 이런 거래들은 개인간 혹은 회사간의 거래이기 때문에 내가 누구와 거래를 할 것인지 분명하게 알고 또 선택할 수 있습니다.

P2P 방식의 거래는 확장성이 있습니다. 주문은 참여자들 간에서만 전송되고 원하는 수량만큼 한번만 거래하고 끝이기 때문에 공개 거래소와 같이 주문 수량이 다 채워질까 걱정하지 않아도 됩니다. 거래소에서는 주문이 완전히 체결되지 않을 수도 있기 때문에 주문 취소도 빈번하게 일어나지만 P2P 거래 방식에서는 이미 해당 주문의 수량만큼 관심을 표한 사람들이 연결되기 때문에 주문을 낸 수량만큼 거래가 체결될 가능성이 굉장히 높습니다. 추가적으로, 비용이 많이 드는 알고리즘 연결 방식과는 다르게 P2P 방식으로는 거래 상대방을 찾는 방식이 온체인이든 오프체인이든 단순하게 이루어집니다.

P2P 방식 교환은 공개되지 않습니다. 두 참여자가 거래를 하기로 동의를 하게 되면 제 3 자가 개입할 필요가 없습니다. 협상의 과정에서 커뮤니케이션은 참여자 간에만 비공개로 이루어지기 때문에 주문 정보에 따라 다른 참여자가 이득을 취할 기회를 제거합니다. 주문이 체결되고 나서야 주문에 대한 정보가 공개 됩니다.

P2P 방식 거래는 공정합니다. 주문이 생성되어 거래 참여자들끼리만 전송하기 때문에 다른 사람이 이득을 취할 수 없습니다. 다수의 거래자가 참여한다면 참여자들은 공개 거래소와 동등하거나 더 좋은 가격으로 거래할 수 있습니다. 추가적으로, 주문을 내면서 가격을 결정할 때 자동화된 low-latency 거래자에게 피해를 볼 걱정할 필요가 없습니다.

블록체인 오더북의 확장성, 비공개성, 그리고 공정성에 대한 한계 때문에 대안에 대한 필요성이 부각되고 있습니다. 오늘 날의 이더리움 생태계는 오픈된 P2P 자산 거래 솔루션이 필요합니다.

Swap 을 소개합니다

Swap 은 이더리움 블록체인 상에서 토큰을 거래할 수 있는 진정한 P2P 생태계를 이루는 프로토콜 입니다. 아래에서는 효율적으로 거래 상대방을 발견하고 협상을 할 수 있도록 지원해주는 방식에 대해 기술하였습니다. 이러한 프로토콜은 자산 교환 생태계의 기반이 되고 이더리움 생태계의 성장을 돕도록 설계되었습니다. 본 백서를 발간하고 나서 실제 다양한 응용 애플리케이션에서 사용될 수 있는 프로토콜을 만들어나가고자 하는 생태계 참여자들의 의견을 받으면서 논의를 계속해 나가고 싶습니다.

2) 피어 프로토콜 (Peer Protocol)

적합한 거래 참여자들끼리 몇 개의 메시지만 주고 받을 수 있게 해주면 거래들이 빠르고, 공정하고, 비공개적으로 협상될 수 있습니다. 이 백서에서 메이커는 주문을 제공하는 개인 혹은 단체이고, 테이커는 주문을 체결하는 개인 또는 단체를 칭합니다. 모든 당사자들이 피어(peer)의 역할을 하기 때문에 메이커나 테이커의 역할을 언제든지 담당할 수 있습니다. 아래에 기술된 토큰은 ERC20 토큰이며 ERC20 표준을 따르는 토큰들은 이 프로토콜을 활용하여 교환될 수 있습니다.

핵심 프로토콜은 아래의 그림과 같이 진행됩니다. 메이커와 테이커는 오프 체인에서 협상을 합니다. 아래의 Contract 는 테이커가 블록체인 상에서 주문을 체결할 때 호출하는 이더리움 스마트 컨트랙트를 칭합니다.

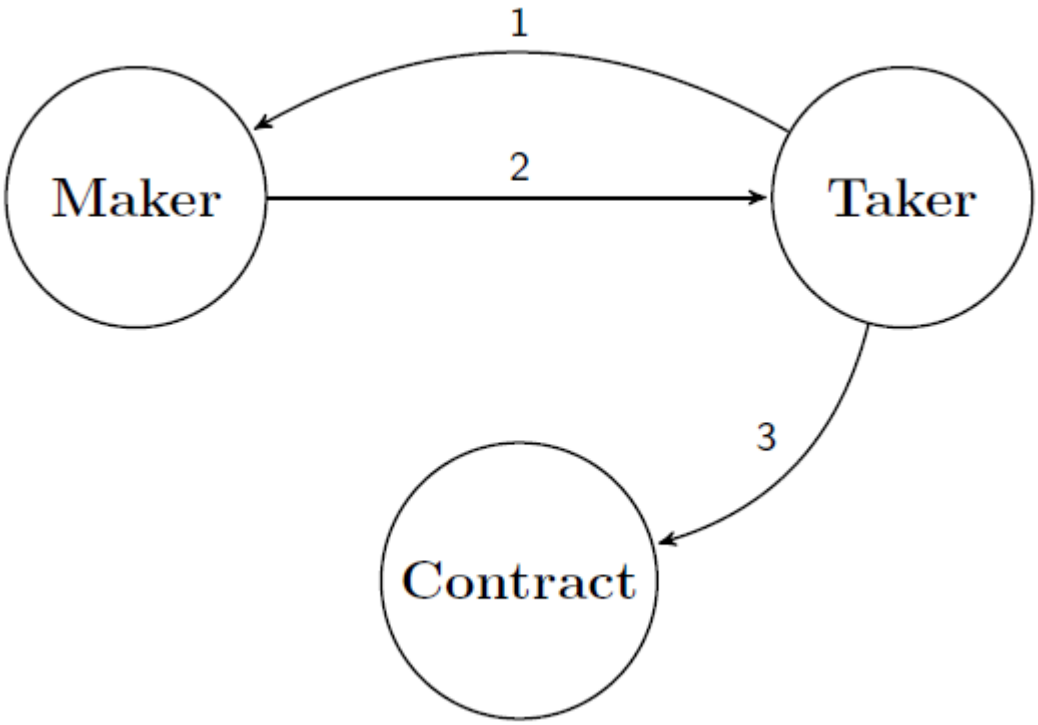

그림 1: 주문 요청, 제공, 체결 과정

1. 테이커가 메이커 대상으로 `getOrder` 를 호출한다
2. 메이커가 답변으로 주문을 제시한다
3. 테이커가 Contract 대상으로 `fillOrder(order)` 를 호출한다

2.1) 주문 API

아래에 나오는 API 들은 전송으로부터 자유로운(transport-agnostic) 원격 프로시저 호출(Remote Procedure Calls -RPC)이며, Peer 와 서비스간 커뮤니케이션에 사용됩니다. 예제에서는 토큰의 티커를 주소 대신 사용하지만, 실제 호출은 ERC20 호환 토큰의 주소를 필요로 합니다. 더 기술적인 세부사항들은 다른 문서에서 다루어지는 관계로, 아래의 호출 서명은 간단한 논의 목적으로만 기술하였습니다.

주문 API 는 오프체인으로 운영되며 당사자 간의 비동기 호출을 거래 과정에서 특정화합니다. 사용자들은 요청-제공 사이클을 동기화 가능한 요청-응답 사이클로 사용하도록 선택할 수 있습니다. 메이커가 주문을 지정해 놓으면, 테이커가 이후에 주문을 스마트 컨트랙트로 제출하여 완료할 수 있습니다.

getOrder(makerAmount, makerToken, takerToken, takerAddress)

테이커가 호출. 토큰 교환을 위하여 메이커에게 거래를 요청.

예: *"BAT 로 10 개의 GNO 를 구매하고 싶습니다."*

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

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

테이커가 호출. 토큰 교환을 위하여 메이커에게 거래를 요청.

예: *"10 GNO 를 5 BAT0 에 판매하고 싶습니다."*

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

2.2) 견적 API

견적은 참여자들간 가격 정보를 보여주기 위하여 사용되며 실제 주문에 사용되지는 않습니다. 서로의 조건이 충족된다면 견적은 나중에 주문으로 변경할 수 있습니다.

getQuote(makerAmount, makerToken, takerTokens)

테이커가 호출, 특정 토큰의 견적을 메이커에게 요청.

예: *"10 GNO 를 BAT 으로 사려면 얼마를 지불해야 하나요?"*

```
getQuote(10, GNO, [BAT])
```

provideQuote(makerAmount, makerToken, takerAmounts)

메이커가 호출, 테이커가 요청한 토큰의 견적을 제공.

예: *"10 GNO 를 BAT 으로 사려면 얼마를 지불해야 하나요?"*

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

3) 인덱서 프로토콜 (Indexer Protocol)

인덱서는 거래를 하고 싶어하는 사람들의 토큰 구매, 혹은 판매 의사를 기반으로 peer 들을 모으고 연결해주는 오프체인 서비스입니다. 많은 메이커가 될 유저들이 거래 의사를 담아 보내면, 테이커가 인덱서에게 적합한 거래상대를 찾아달라고 요청할 때 복수의 가능한 결과들을 응답합니다. 테이커가 거래하고 싶은 메이커를 찾고 나면 위 Peer 프로토콜을 이용하여 협상을 진행하게 됩니다. 메이커와 테이커 사이에 합의가 일어나면, 주문이 스마트 컨트랙트로 올라가게 됩니다. 메이커, 테이커와 인덱서의 상호작용은 아래 그림에 묘사되어 있습니다. 메이커, 테이커와 인덱서는 모두 블록체인 밖에서 상호작용하며, 선호하는 어떠한 메시지 교환 매개체를 통하더라도 통신할 수 있습니다.

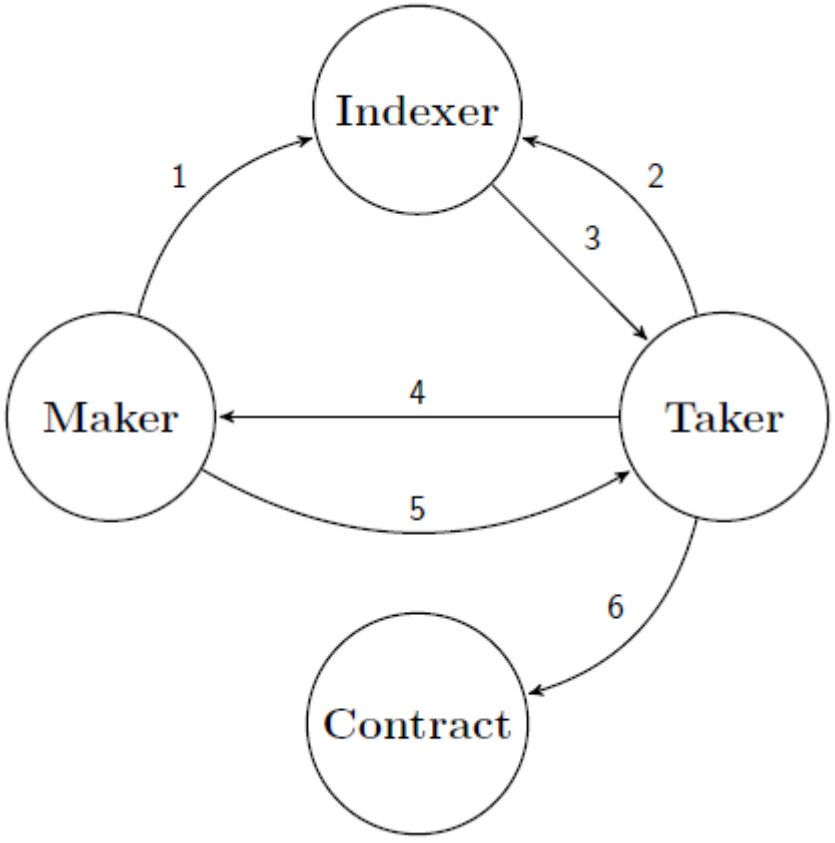

그림 2: 거래상대를 찾아 거래

1.메이커가 `addIntent` 를 인덱서에게 요청

2.테이커가 `findIntent` 를 인덱서에게 요청

3.인덱서가 `foundIntent`(메이커의 의사)를 테이커에게 요청

4.테이커가 메이커에게 `getOrder` 를 요청

5.메이커가 주문에 답변

6.테이커가 `fillOrder`(주문 완료)를 컨트랙트에 요청

다수의 메이커와, 테이커와, 인덱서와의 상호작용이 아래 그림에 묘사되어 있습니다. 각각의 메이커들은 각자 그들의 거래 의사 및 거래 조건을 밝힙니다. 테이커가 특정 조건을 요구하는 메이커를 찾아달라고 요청하면, 인덱서는 조건에 맞는 이더리움 주소 리스트와 상세 내역을 전달합니다.

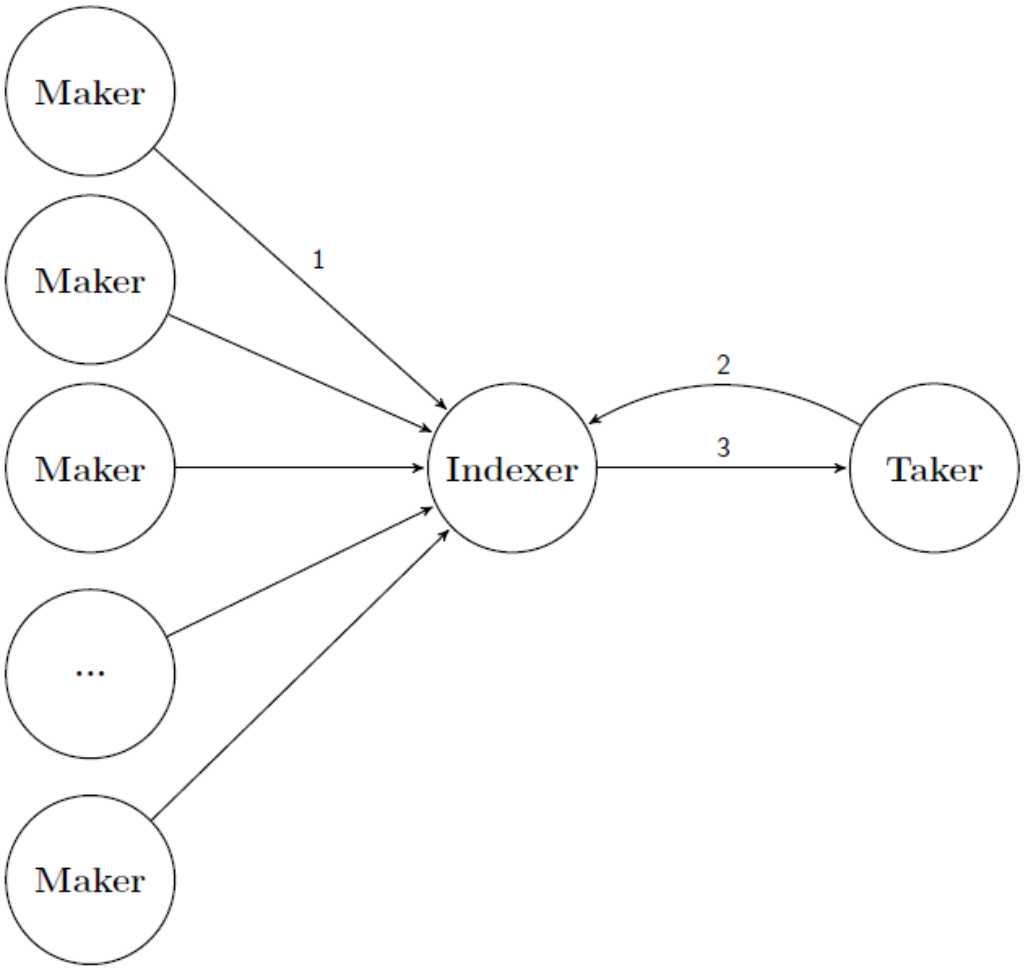

그림 3: 메이커의 `addIntent` 호출 및 테이커의 `findIntent` 호출

1. 다수의 메이커들이 인덱서에게 `addIntent` 를 호출합니다.
2. 테이커는 인덱서에게 `findIntent` 를 호출합니다.
3. 인덱서가 `foundIntent`(메이커의 의사)를 테이커에게 호출합니다.

테이커가 메이커들을 찾고 나면, 주문 API 를 사용하여 메이커들을 비교 검토하고 주문을 낼 수 있습니다. 테이커가 주문을 결정하면, 테이커는 `fillOrder` 를 호출하여 주문을 스마트 컨트랙트에 기록합니다.

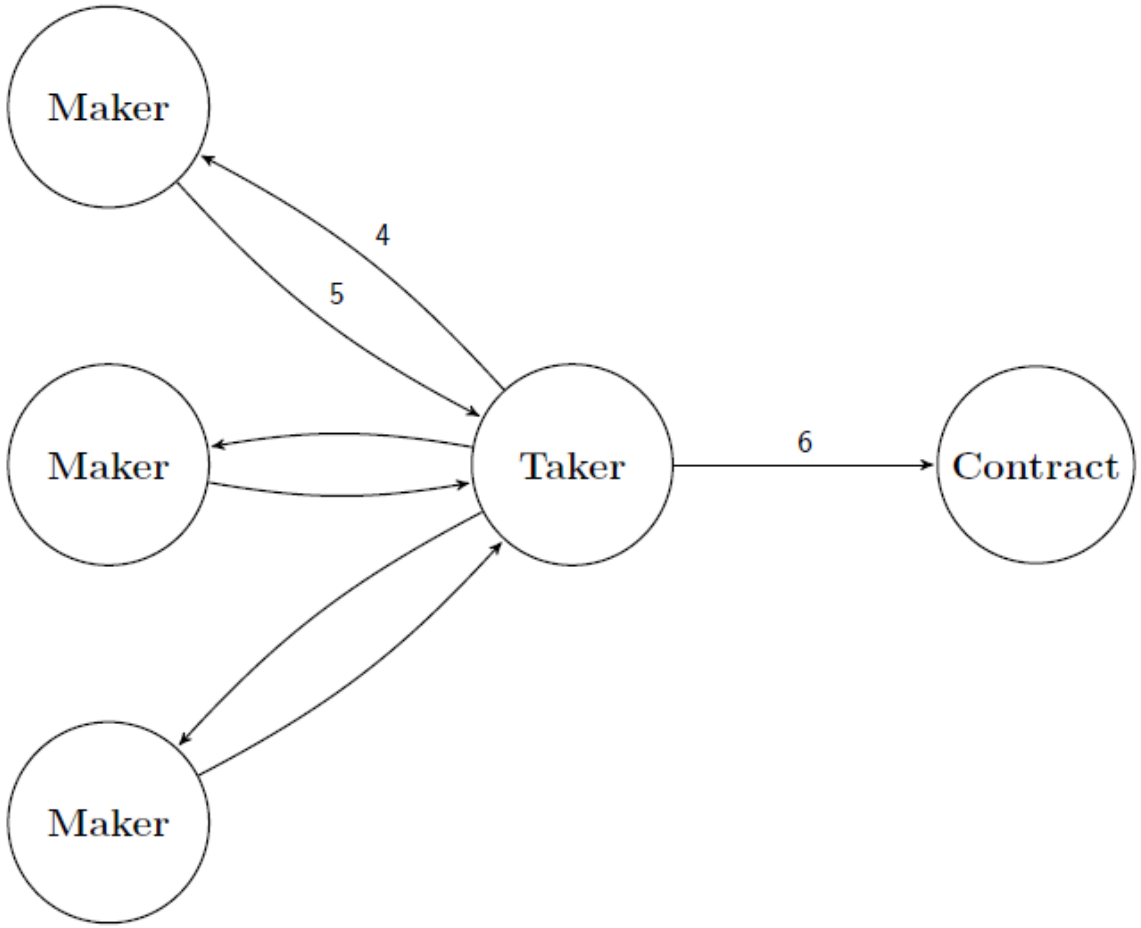

그림 4: 테이커의 `getOrder` 호출 및 테이커의 `fillOrder` 컨트랙트 호출

4. 테이커는 `GetOrder` 를 다수의 메이커에게 호출합니다.
5. 메이커는 자신들의 주문내역을 응답합니다.
6. 테이커는 주문을 선택하여 `fillOrder`(주문 완료)를 컨트랙트에 호출합니다.

3.1) 인덱서 API (Indexer API)

인덱서 API 는 매수자, 매도자가 거는 거래 요청(intent)을 관리합니다. 아래 명령문은 거래 당사자(peers)와 인덱서 간에 호출됩니다.

addIntent(makerToken, takerTokens)

토큰 매수/매도 요청 접수

예: *"GNO 를 BAT 로 거래하고 싶습니다"*

```
addIntent(GNO, [BAT])
```

removeIntent(makerToken, takerTokens)

토큰 거래 요청(intent) 삭제

예: *"더 이상 GNO 를 BAT 으로 거래할 생각이 없습니다."*

```
removeIntent(GNO, [BAT])
```

getIntent(makerAddress)

특정 사용자(makerAddress)의 거래 요청(intent) 내역 호출

예: *"특정 사용자[makerAddress]가 거래하고자 하는 토큰 내역을 출력하겠습니다"*

```
getIntent(<makerAddress>)
```

findIntent(makerToken, takerToken)

특정 토큰을 거래하고자 하는 모든 사용자 검색

예: "GNO 를 BAT 와 거래하고자 하는 사용자를 검색하겠습니다"

```
findIntent(GNO, BAT)
```

foundIntent(makerAddress, intentList)

인덱서가 거래 요청 내역이 있는 사용자를 찾았을 경우

예: *"10 GNO 를 BAT 로 거래하고자 하는 사용자가 검색되었습니다"*

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

4) 오라클 프로토콜 (Oracle Protocol)

오라클은 메이커와 테이커에 거래가격 정보를 전달하는 오프체인(off-chain) 서비스입니다. 테이커에게 매도/매수 희망 가격을 전달하기 전에, 메이커는 오라클을 통해 적정 거래가를 추천받을 수 있습니다. 마찬가지로, 테이커 역시 거래 요청을 받은 뒤 오라클을 통해 매도 가격이 적절한지 확인할 수 있습니다. 오라클은 이러한 거래가격 정보를 제공해 테이커와 메이커 모두 가격에 대한 충분한 정보를 얻은 후 거래 결정을 내리고, 차질 없이 거래가 진행되도록 합니다.

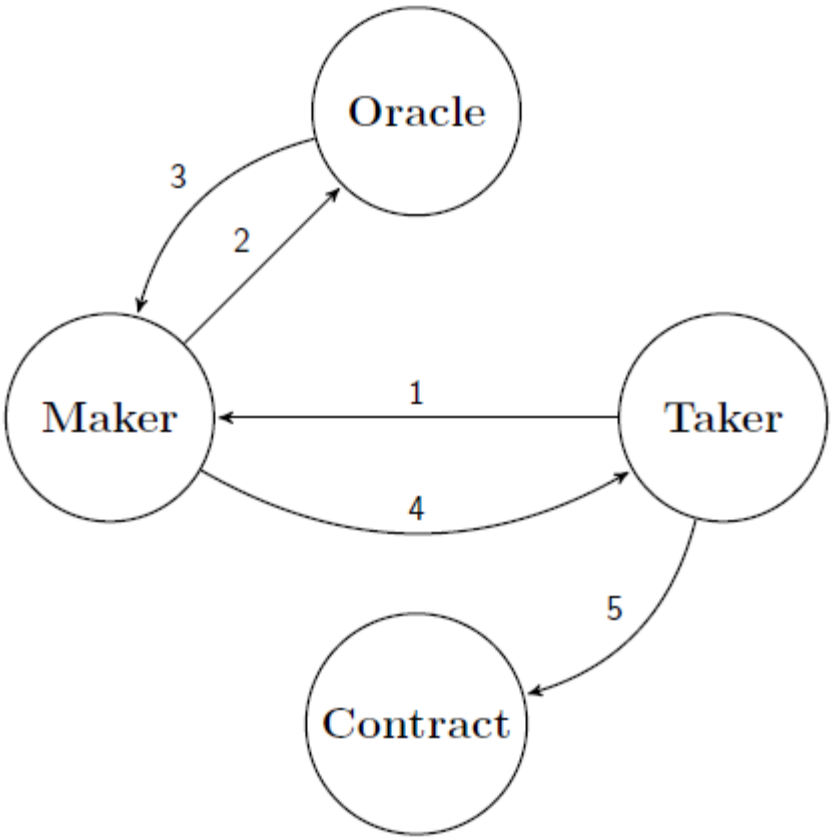

그림 5: 거래 전, 메이커가 오라클을 통해 적정 거래 가격을 조회하는 과정

1. 테이커는 `getOrder` 명령을 통해 메이커를 호출
2. 메이커는 `getPrice` 명령을 통해 오라클에게 가격 정보 요청
3. 오라클은 메이커에게 적정 거래 가격을 반환
4. 가격 정보를 분석한 후, 메이커는 거래 요청을 전송
5. 테이커는 `fillOrder(order)` 명령을 호출

테이커가 거래 요청을 받은 후에 이와 유사한 과정(적정 가격을 주고 받는)이 테이커와 오라클 간에도 진행됩니다.

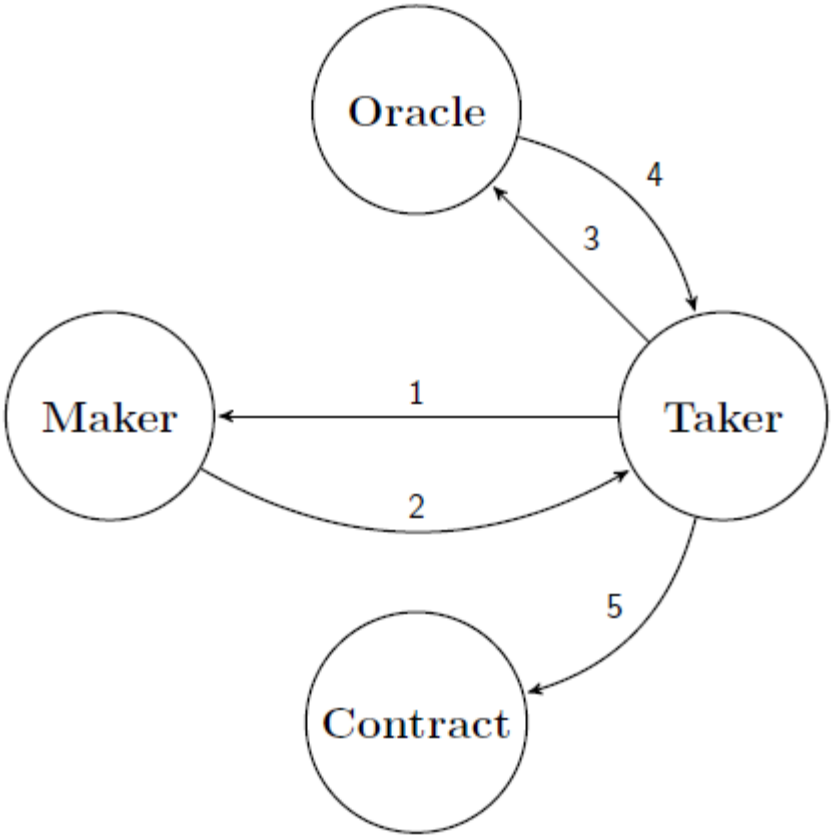

그림 6: 테이커가 거래 요청을 넣기 전 오라클을 통해 적정 가격을 조회하는 과정

1. 테이커가 `getOrder` 명령을 통해 메이커를 호출
2. 메이커가 거래 요청에 응답
3. 테이커가 `getPrice` 명령을 통해 오라클에게 가격 정보 요청
4. 오라클이 적정 가격 반환
5. 가격 정보 분석 후, 테이커가 `fillOrder(order)` 명령을 호출

4.1) Oracle API

메이커와 테이커는 오라클 API 를 활용해 거래 가격을 산정할 수 있습니다. (오라클이 제시한) 추천 가격은 참고용이며, 실제 거래가 이뤄지지 않을 수 있습니다.

getPrice(makerToken, takerToken)

테이커나 메이커가 오라클을 통해 가격을 조회하는 명령문

예: *"GNO-BAT 간 현재 거래 가격을 조회하겠습니다"*

```
getPrice(GNO, BAT)
```

providePrice(makerToken, takerToken, price)

오라클이 메이커나 테이커에게 가격을 반환하기 위해 호출하는 명령문

예: *"GNO-BAT 간 현재 거래 가격은 0.5 입니다"*

```
providePrice(GNO, BAT, 0.5)
```

5) 스마트 컨트랙트 (Smart Contract)

거래 주문을 넣거나 취소하기 위한 이더리움 스마트 컨트랙트

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

테이커는 거래를 위해 토큰 간 아토믹스왑(Atomic swap, 거래소를 거치지 않고 이뤄지는 p2p 거래)을 호출합니다. 컨트랙트를 통해 명령 호출자와 테이커가 일치하는지와 주문 유효 시간이 지나지 않았나를 확인합니다. 거래를 확정짓기 위해 거래 당사자들은 명시된 토큰을 승인해 컨트랙트가 명시된 액수만큼 출금할 수 있도록 해야 합니다. 토큰의 이체를 위해, 컨트랙트는 각각의 토큰에 `transferFrom` 명령을 호출합니다. 해당 명령문(`fillOrder`)이 성공적으로 실행되면 거래 완료 내역이 블록체인에 올라가게 됩니다.

예: *"5 GNO 를 10BAT 로 최종 거래가 확정되었습니다"*

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

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

테이커와 거래가 진행 중이지만 확정되지 않은 주문은 취소될 수 있습니다. 이 명령문은 메이커가 호출하게 되며, 해당 거래를 확정 상태로 바꿔 다른 거래 요청이 들어오지 않도록 합니다. 해당 명령문(`cancelOrder`)이 성공적으로 실행되면 거래 취소 내역이 블록체인에 올라갑니다.

예: *5 GNO 를 10 BAT 로 교환하고자 했던 거래를 취소하겠습니다"*

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

5.1) 이더리움 거래 (Ether Orders)

스마트 컨트랙트를 통해 이더리움(ETH)으로 토큰을 거래할 수도 있습니다. 만약 주문 내역의 `takerToken` 주소값이 비어있다면(0x0), 스마트 컨트랙트는 명령문 호출과 함께 전달된 이더의 가치를 평가하며 테이커를 대신해 메이커로 전달합니다.

6) 요약

Swap 프로토콜은 이더리움 네트워크 기반 탈중앙화 거래소에 대한 늘어나는 수요를 충족시킬 수 있습니다. 블록체인 기반의 오더북은 새로운 개념이며 우리 생태계에도 포함되지만, 몇몇 한계점으로 인해 기존 중앙화 솔루션과 경쟁하기 어려울 것으로 판단됩니다. Swap 은 이러한 한계점에 영향을 받지 않으며, 탈중앙화된 방법을 제공합니다.

Swap 프로토콜을 통해 사용자들은 더 나은 거래 조건을 포기할 필요 없이 확장성을 가지고(scalable) 폐쇄적(private)이며 공정한 방식으로 유동성에 접근할 수 있습니다. 해당 프로토콜과 API 는 확장이 가능하며, 우리는 커뮤니티와 함께 어플리케이션을 붙여나가고자 합니다. 피드백을 언제나 환영하며 당신과 함께 이더리움 커뮤니티를 발전시킬 날 만을 기다리겠습니다.